JTLS-GO

Version Description Document

August 2020

DEPARTMENT OF DEFENSE
JOINT STAFF J7
116 LAKE VIEW PARKWAY
SUFFOLK, VA 23435-2697

JOINT THEATER LEVEL SIMULATION - GLOBAL OPERATIONS
(JTLS-GO 5.1.9.0)

[Blank Page]

ABSTRACT

The Joint Theater Level Simulation - Global Operations (JTLS-GO®) is an interactive, computer-based, multi-sided wargaming system that models combined joint and coalition resource air, land, naval, and Non-Governmental Organization (NGO) environments.

This *JTLS-GO Version Description Document (VDD)* describes the new features of the Version 5.1.9.0 delivery of the configuration-managed JTLS-GO software suite.

JTLS-GO 5.1.9.0 is a Maintenance release of the JTLS-GO 5.1 series that includes an updated wespac51 demonstration database as well as updated repository data held in the repository51 database. There are no major Engineering Change Proposals (ECPs) included with this release, but there are a few minor ECPs that required no new data or data format changes, which are summarized in Chapter 2. Code modifications that represent corrections to known Software Trouble Reports (STRs) are described in Chapter 3. Remaining and outstanding STRs are described in Chapter 4.

This publication is updated and revised as required for each Major or Maintenance version release of the JTLS-GO model. Corrections, additions, or recommendations for improvement must reference specific sections, pages, and paragraphs with appropriate justification and be forwarded to:

JTLS-GO Development Team Leader
ROLANDS & ASSOCIATES Corporation
120 Del Rey Gardens Drive
Del Rey Oaks, California 93940 USA
jtlsdev@rolands.com

[Blank Page]

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 SCOPE

This *JTLS-GO Version Description Document* (VDD) describes Version 5.1.9.0 of the configuration managed Joint Theater Level Simulation - Global Operations (JTLS-GO®) software suite. JTLS-GO 5.1.9.0 is a Maintenance release for the JTLS-GO 5.1 series.

JTLS-GO 5.1.9.0 includes the entire JTLS-GO suite of software, a repository of engineering level data, and a realistic demonstration scenario based on the Western Pacific theater of operations, called “wespac51”. No database format modifications have been made for this release, but a static data error for allowable OTH-Gold Ship Types was corrected in Version 5.1.1.0. This does require a user to execute a correction procedure for all of their Version 5.1.0.0 scenarios loaded in Oracle prior to the release of Version 5.1.1.0. Information on this procedure can be found on Page 1-11.

Descriptions of minor Engineering Change Proposals (ECPs) implemented for this release are provided in Chapter 2.0. Explanations of all Software Trouble Reports (STRs) corrected in this release are provided in Chapter 3.0. Outstanding STRs are provided in Chapter 4.0. Changes made to the JTLS-GO 5.1 engineering data repository are provided in APPENDIX C.

JTLS-GO 5.1.9.0 executes on the Red Hat Enterprise Linux Version 7.6 64-bit operating system. The Web-Hosted Interface Program (WHIP®) user workstation interface can be executed on any operating system from any Java-compatible Web browser.

1.2 INVENTORY OF MATERIALS

This section lists documents and software that are relevant to JTLS-GO. All JTLS-GO documents included in this delivery are provided in Portable Document Format (PDF) within a documents subdirectory.

1.2.1 Obsolete/Outdated Documents

No documents have been deleted or become outdated as a result of this release.

1.2.2 Unchanged Documents

- *JTLS-GO Analyst Guide* (JTLS-GO Document 01, Version 5.1.3.0)
- *JTLS-GO Configuration Management Plan (*JTLS-GO Document 03, Version 5.1.2.0)
- *JTLS-GO Controller Guide* (JTLS-GO Document 04, Version 5.1.8.0)
- *JTLS-GO DDS User Guide* (JTLS-GO Document 06, Version 5.1.7.0)

- *JTLS-GO Director Guide* (JTLS-GO Document 07, Version 5.1.2.0)
- *JTLS-GO Executive Overview* (JTLS-GO Document 08, Version 5.1.3.0)
- *JTLS-GO Installation Manual* (JTLS-GO Document 09, Version 5.1.6.0)
- *JTLS-GO WHIP Training Manual* (JTLS-GO Document 10, Version 5.1.6.0)
- *JTLS-GO Player Guide* (JTLS-GO Document 12, Version 5.1.3.0)
- *JTLS-GO Repository Description* (JTLS-GO Document 14, Version 5.1.2.0)
- *JTLS-GO Software Maintenance Manual* (JTLS-GO Document 15, Version 5.1.2.0)
- *JTLS-GO Technical Coordinator Guide* (JTLS-GO Document 16, Version 5.1.5.0)
- *JTLS-GO Entity Level Server User Guide* (JTLS-GO Document 19, Version 5.1.2.0)
- *JTLS-GO Federation User Guide* (JTLS-GO Document 20, Version 5.1.8.0)
- *JTLS-GO C4I Interface Manual* (JTLS-GO Document 21, Version 5.1.8.0)
- *JTLS-GO Air Services User Guide (*JTLS-GO Document 24, Version 5.1.2.0)

1.2.3 Updated Documents

- *JTLS-GO Data Requirements Manual* (JTLS-GO Document 05, Version 5.1.9.0)
- *JTLS-GO Version Description Document* (JTLS-GO Document 17, Version 5.1.9.0)

1.2.4 New Documents

No new documents are delivered with JTLS-GO 5.1.9.0.

1.2.5 Delivered Software Components

JTLS-GO 5.1.9.0 may be delivered either on a CD or as a set of compressed TAR files to be downloaded. Either method includes the complete suite of software executable code and command procedures. The following software components are included with this release:

- Combat Events Program (CEP)
- Scenario Initialization Program (SIP)
- Interface Configuration Program (ICP)
- Reformat Spreadsheet Program (RSP)

- JTLS Symbols Application (JSYMS)
- Database Development System (DDS)

  Database Configuration Program (DCP)

  DDS Client User Interface (DDSC)
- ATO Translator Service (ATOT)
- ATO Generator Service (ATOG)
- ATO Retrieval Program (ATORET)
- JTLS Convert Location Program (JCONVERT)
- Count Critical Order Program (CCO)
- JTLS HLA Interface Program (JHIP)
- After Action Review Client (AARC)
- Scenario Data Client (SDC)
- Order Entry Client (OEC)
- Order Verification Tool (OVT)
- Modernized Integrated Database (MIDBTool)
- JTLS Object Distribution Authority (JODA)
- Web Services Manager (WSM)
- Web-Hosted Interface Program (WHIP) and its component programs:

  Apache Server (APACHE)

  JTLS XML Serial Repository (JXSR)

  Order Management Authority (OMA)

  Synchronized Authentication and Preferences Service (SYNAPSE)

  XML Message Service (XMS)

  Total Recall Interactive Playback Program (TRIPP)

When operating the TRIPP capability in current JTLS-GO releases, users are not prevented from logging into an actively running TRIPP making a connection to the same Replay JXSR. A TRIPP, as documented, requires its own Replay JXSR to control and perform the replay of the recorded simulation events.

This situation as a consequence can have more than one user concurrently control the playback of the game, as the shared Replay JXSR will honor each of the user's playback requests and will then change what each connected user sees on their TRIPP instance.

This issue has been addressed in JTLS-GO 6.0 by only permitting one login per TRIPP user instance, in the same manner users are prevented from logging into the same WHIP instance. This correction could not be implemented in JTLS-GO 5.1 due to Cybersecurity restrictions on major interface changes.

Organizations should develop their own procedures to ensure that only one user logins to a given TRIPP at a time.

- Entity Level Server (ELS)
- JTLS Operational Interface (JOI) for both OTH-Gold and Link 16 generation
- Tactical Electronic Intelligence (TACELINT) Message Service
- KML Operational Interface (KOI)
- JTLS Transaction Interface Program (JTOI)
- JTLS Interface Network Navigator (JINN)
- JTLS Order of Battle Editor (JOBE)
- JTLS Geographic Information System (GIS) Terrain Building Program

Instructions for installing JTLS-GO 5.1.9.0 are provided in the *JTLS-GO Installation Manual*. Installing a previous version of JTLS-GO prior to installing JTLS-GO 5.1.9.0 is not necessary. No other upgrade beyond installation of the compressed TAR files (or CD) is required. The software provided with this delivery is a complete release that includes all files and code required to execute JTLS-GO.

1.2.6 Released Databases

This release includes the following sample unclassified databases:

- The scenario “repository51” serves as a repository of engineering level data. Although not useful as a scenario, it does follow all of the database requirements for a scenario, and should be loaded into your Oracle scenario table-space. With JTLS-GO 5.1.9.0, it is possible to access and copy records from the repository51 database into your own developed scenarios.

- The scenario “wespac51”, which is based on the Western Pacific theater of operations and is suitable for training and demonstrations.

1.3 INTERFACE COMPATIBILITY

1.3.1 Support Software

JTLS-GO 5.1.9.0 requires the following versions of support software, including operating systems, compilers, scripting utilities, database tools, transfer protocols, and display managers:

- Operating system for the model: Red Hat Linux Enterprise Edition Version 7.6 (ES), 64-bit architecture.

  In previous version of JTLS-GO 5.1, the JTLS-GO Engineering Team suggested that Security Enabled (SE) Linux be disabled on the JTLS-GO servers. Further testing has indicated that this restriction is no longer a problem. Each organization site should review their internal system requirements to determine whether to enable this capability on their JTLS-GO servers.

- JTLS-GO 5.1 has been tested with the following versions of Linux 7:

  Red Hat Linux 7.6 - This operating system license must be purchased, but it has been approved by the Defense Information Systems Agency (DISA) for use by U.S. Government Agencies.

  Oracle Linux 7.6 - This operating system is free to download, use, and distribute, and is provided in a variety of installation and deployment methods. It has been approved by DISA for use by U.S. Government Agencies.

  CentOS Linux 7.6 - A free version of Linux 7 that has **not** been approved by DISA for use by U.S. Government Agencies.

- There are no restrictions on the operating system used for client workstations, except that the operating system must have a Java-enabled web browser. JTLS-GO 5.1.9.0 has been tested on the following operating systems:

  Red Hat Linux Enterprise Edition Version 7.6.

  CentOS Linux Version 7.6 and 7.8.

Windows 7 and Windows 10, which can be used only if the workstation is an external HTTP client of the simulation network.

- The JTLS-GO 5.1 series no longer uses Oracle Java, and has moved to the latest version of OpenJDK 8, which is OpenJDK 8 Version 262. We no longer deliver the Java Runtime Environment (JRE) within the JTLS-GO delivered software TAR files. Each user organization must obtain the latest version of the OpenJDK Red Hat Package Manager (RPM) and install the RPM on the servers used by JTLS-GO.

- JTLS-GO uses IcedTea to provide the OpenJDK web start capability that implements the web-enabled JTLS-GO functionality. The current version of JTLS-GO supports IcedTea version 1.8.3.

  Red Hat Linux version 7.7 continues to distribute IcedTea version 1.7.1. There are available RPM packages for IcedTea 1.8.3. JTLS-GO users must explicitly install the request RPM on the JTLS-GO servers and client workstations. This provides JTLS-GO 5.1.9.0 the capability to fully use either unsecured internal connections (http:) or secure connections (https:).

- JTLS-GO database tools require use of a certified Oracle database server and the full Oracle Client installation for runtime requirements. Additional installation details can be found in Section 1.5.2 of this chapter.

- Windows software, X11R5 server, Motif 1.2 Library, Motif Window Manager: These items are included as part of the supported versions of Red Hat Linux ES.

- TCP/IP is required for inter-process communication between the JODA data server and all user interface service programs. The version of TCP/IP included with the supported versions of Red Hat Linux ES is sufficient.

- The Perl script language is used by the JTLS-GO system and game setup scripts. The version of Perl included with the supported versions of Red Hat Linux ES is sufficient. The Perl program is typically located in the /usr/bin directory. If Perl is installed in a another location, a link should be created from the /usr/bin directory to this program.

- SIMSCRIPT II.5 (SIMSCRIPT to C) translator/compiler: SIMSCRIPT is required for recompiling JTLS-GO code. It is not necessary to have a SIMSCRIPT compiler to execute JTLS-GO, because all JTLS-GO software executables are statically linked with the SIMSCRIPT libraries. The compiler is needed only by a U.S. Government Agency that can obtain source code and plans to re-compile JTLS-GO SIMSCRIPT code. To obtain a SIMSCRIPT compiler, contact CACI Inc.

- ANSI C Compiler: It is not necessary to use a C compiler to execute JTLS-GO. This compiler is needed only by a U.S. Government Agency that can obtain source code and plans to re-compile any of the JTLS-GO component programs. The C Compiler version delivered with the supported versions of Red Hat Linux ES is sufficient.

- C++ Compiler: It is not necessary to use a C++ compiler to execute JTLS-GO. This compiler is needed only by U.S. Government Agency that can obtain source code and plans to re-compile any of the JTLS-GO HLA component programs. The C++ Compiler version delivered with the supported versions of Red Hat Linux ES is sufficient.

- The JTLS-GO DDS (Database Development System) application uses these open source libraries:

  JFreeChart, licensed under LGPL (GNU LESSER GENERAL PUBLIC LICENSE) by Object Refinery Limited, http://www.object-refinery.com.

  JCommon, licensed under LGPL2.1 (GNU LESSER GENERAL PUBLIC LICEN.SE version 2.1 or later) by Object Refinery Limited, http://www.object-refinery.com

  Commons-math3-3.0.jar, licensed under Apache Software Foundation (Apache License, Version 2.0), http://www.apache.org/licenses/LICENSE-2.0HLA Compliance.

- KML Operational Interface (KOI)

  The Keyhole Markup Language (KML) Operational Interface (KOI) server utility enables the model to feed operational simulation data to any version of Google Earth™. The display capabilities and data transfer features of this terrain viewer are sufficiently robust to be used as a base-level operational interface. Operational Players who are restricted from using the COP, C2PC, or other C4I systems may be able to install and use Google Earth and configure the KOI to provide a capability that resembles C4I for observing perception Force Side data.

  Chapter 3 of the *JTLS-GO C4I Interface Manual* describes requirements and procedures for using the KOI capabilities.

- JTLS-GO 5.1 implements SSH Tunneling between Apache and the services, and among the services. Rigorous testing should be done prior to use in any exercise, and particular attention should be paid to network performance under load.

1.3.2 JTLS-GO Cybersecurity Compliance

Because of recent incidents of intrusions into software systems, the United States Department of Defense (DoD) has implemented a strong and strictly enforced Cybersecurity program. JTLS-GO, as software that executes on DoD systems, must comply to the mandates of the program, as well as requirements of all of the third party software used by JTLS-GO, such as Oracle and Java.

One of the DoD requirements is that the software must implement a methodology that ensures that the end user keeps the software up-to-date and properly installs all security patches. The primary purpose of this release is to provide a version of JTLS-GO compiled with the latest security release of OpenJDK, OpenJDK 8 Version 262. To meet these Cybersecurity requirements, each user organization should ensure that this version of OpenJDK is loaded on the JTLS-GO servers and any client machines used to connect to JTLS-GO.

JTLS-GO has completed the CSIA program mandates and the JTLS-GO 5.1 series of releases has been granted an Authority To Operate (ATO) on DoD systems. Contact the U.S. Government Program Manager, Mr. Donald Weter (donald.e.weter.civ@mail.mil), for additional information.

1.3.3 JTLS-GO High Level Architecture Compliance

The JTLS-GO 5.1.9.0 release is fully High Level Architecture (HLA) compliant, and includes all the programs required to run JTLS-GO in an HLA mode. JTLS-GO is part of the federation known as "GlobalSim". This federation is a comprehensive constructive simulation solution for joint training and wargaming, that helps commanders and all levels of staff prepare for a range of operational scenarios.

The solution combines JTLS-GO with CAE's GESI constructive tactical entity-level simulation system. CAE's GESI constructive simulation system is designed to run complex and comprehensive exercises from the company level up to division level. The CAE GESI system is used to represent a virtual battlefield, including weapons, vehicles, aircraft, and ground forces.

Combining JTLS-GO and GESI brings together operational and tactical level constructive simulations to prepare commanders and staff to make timely, informed and intelligent decisions across the full spectrum of operations, including conventional combat, disaster relief, and operations other than war.

All JTLS-GO software needed to run GlobalSim Federation is included in this delivery. JTLS-GO uses the Federation Object Model (FOM), located in the $JGAME/data/hla directory. Federation testing of JTLS-GO with CAE's GESI wargaming system has been accomplished using JTLS-GO Version 5.1.9.0.

The HLA RTI (Run Time Infrastructure) executive program recommended for use with this release is Pitch pRTI Evolved 4.4.2.0. However, this program is not included in the JTLS-GO 5.1.9.0 delivery. Users may obtain a full installation package of the RTI executive program from Pitch Corporation (www.pitchtechnologies.com). For information about executing the HLA RTI Executive and other HLA-related software, refer to the appropriate HLA documentation and user guides.

1.4 DATABASE MODIFICATIONS

Significant database structure differences exist between the JTLS-GO 5.1 series and the previous JTLS-GO 5.0 series database structure.

To upgrade your JTLS-GO 5.0 scenario to JTLS-GO 5.1 compatibility, see instructions listed in Chapter 3.1 of the *JTLS-GO DDS User Guide*.

1.4.1 JTLS-GO Using Legacy Default Symbol Set

If a user organization is still using the pre-JTLS-GO 5.0.0.0 legacy default symbol set, prior to unloading your JTLS-GO 5.1.0.0 formatted data from your Oracle database server into the JTLS-GO 5.1.0.0 scenario American Standard Code for Information Interchange (ASCII) text files, you must execute the JSYMS program using the procedure outlined in Appendix B.11 of the *JTLS-GO DDS User Guide*. This procedure will reorganize the structure of the .gs and .scf symbols-related files.

1.4.2 JTLS-GO Using New Default Symbol Set

You should not make any modifications to the Default Symbol Set delivered with JTLS-GO 5.1.9.0, but end user organizations are free to use the Default Symbol Set in their scenarios and alter the scenario symbol set to meet specific organizational needs.

1.4.3 Standard Repository Changes

The JTLS-GO 5.1 series of JTLS-GO is the first series in which R&A is delivering an unclassified data repository called “repository51”. In future Major releases of JTLS-GO, APPENDIX B will provide a summary of the data structure changes made to the data repository. No data structure changes have been made is this Maintenance release; therefore, APPENDIX B is empty. Refer to Appendix B in the *JTLS-GO 5.1.0.0 Version Description Document*, included with this release, for data structure changes made for the JTLS-GO 5.1 series.

1.5 INSTALLATION

1.5.1 Installation Instructions

The *JTLS-GO Installation Manual,* a PDF file available for direct download, is part of this JTLS-GO delivery. It provides detailed instructions for installing a new version of JTLS-GO.

1.5.2 Oracle Compatibility and Installation

A full Oracle Client (not Instant Client) installation that matches your database server version is currently a requirement for running some JTLS-GO applications. The Oracle Instant Client is not sufficient for JTLS-GO applications because certain Oracle utilities, such as sqlldr, imp, exp, and tnsping, are missing. If you have applied a patchset to your database server, the same patchset should be applied to the Oracle Client installation. A 64-bit Oracle Client installation must be used.

The JTLS-GO scenario/database modification process also expects Oracle 11.2.0.1 or higher full Oracle Client installation. Some sites NFS mount their database server as Oracle Client; other sites prefer a full installation of the Oracle Client to a different directory that mounts to JTLS-GO

(a simple NFS mount will suffice). Your system administrator can choose the appropriate installation.

Assigning the full Oracle Client installation location (or mount point) as the ORACLE_HOME in the JTLS-GO .cshrc file allows connecting to an Oracle database server (11.2.0.1 or higher - including 11gR2 XE) running on any Oracle-certified database server platform.

Oracle offers free Express Editions (XE) of the Oracle relational database management system. Compared to the 11gR2 XE version, the newer 18c XE has a larger footprint and a much more complex database architecture. For test environments and scenario building purposes, or for collecting AAR data for a short period of time, the installation and setup of the 11gR2 XE version is much simpler.

The DDS application utilizes the Oracle GlassFish J2EE server, which, like the JTLS-GO WHIP Apache server, is delivered with JTLS-GO and requires no separate installation.

Refer to Chapter 6 of the *JTLS-GO Installation Manual* for additional details pertaining to the Oracle installation.

1.5.3 Special Installation Instructions

This section describes special instructions that should be followed because of errors corrected in this version and previous bug releases for the JTLS-GO 5.1 series.

1.5.3.1 Fix Stored Procedure For Renaming Transportation Classes And Small Boats

STR JTLS-2020-14818 fixes a problem with the stored procedure used to rename a Transportation Class entity and a Small Boat entity. If the entity is referred to in a Combat System Special Subcategory field, the Combat System table field is not properly updated. This causes a reference problem when reading the Combat System Data.

> Due to the changes implemented as a result of STR JTLS-2020-14818, all JTLS-GO 5.1 scenarios loaded in Oracle should be downloaded and then immediately reloaded.

1.5.3.2 OTH-Gold Ship Types

JTLS-GO 5.1.0.0 was delivered with an old list of OTH-Gold ship types. This problem was corrected in JTLS-GO 5.1.1.0. To properly implement this solution, users must execute the following additional procedures for each of their JTLS-GO Version 5.1.0.0 scenarios loaded in Oracle.

Users who have already executed this procedure for their scenarios after installing JTLS-GO 5.1.1.0, do not need to execute the procedure again.

New OTH-Gold ship types were added to JTLS-GO, due to STR JTLS-2019-14238.

1. Execute the following command:

   cd $JTLSHOME/script/dds/version5.1/scripts/

2. Execute the following command:

   sqlplus yourScenario/OraclePassword @update_oth_gold_types.sql

3. Verify the related execution listing file under the $JDATA/scenario/ directory for errors.

4. Unload your scenario using the JTLS-GO Menu, Options 1 -> 1 -> 4

5. Verify the .srw ascii file for your scenario.

To fix the problem renaming Force Sides, due to STR JTLS-2019-14267:

6. Reload the database.

1.5.3.3 Aircraft Type Foreign Key Definitions

STR JTLS-2019-14541 Referenced Aircraft Class Can Be Deleted, delivered as part of JTLS-GO 5.1.5.0, solved a problem that allowed the deletion of aircraft types that were referenced by units. The STR was solved by removing a rule in the database that allowed the user to set the aircraft type-related foreign keys of the unit tables to NULL.

After loading JTLS-GO 5.1.5.0, the user must unload and then reload their JTLS-GO 5.1 scenarios to have the new foreign key definitions in their database schemas.

Users that have accomplished this procedure after installing JTLS-GO 5.1.5.0, do not need to re-execute this procedure after installing this version of JTLS-GO.

1.5.3.4 Generation Of Combat System Summary Files

STR JTLS-2019-14518 Move Combat System Summary Files, delivered as part of JTLS-GO 5.1.4.0, solved a problem overwriting the summary Combat System Character Separated Value (.csv) files during Batch Runs. This STR was solved by moving the location of the .csv files from the game/<scenario_name>/location directory to a sub-directory under each checkpoint.

For every active game scenario, one of the following procedure options must be executed to establish the directory structure needed by the STR solution. Users who have already executed

this procedure for their scenarios after installing JTLS-GO 5.1.4.0, do not need to execute one of the selected options again:

- Option 1: Rerun the Setup Procedure for each active scenario.
- Option 2: Hand-create a cbtsys_summary sub-directory in the game/<scenario_name> directory. This can be accomplished using the following steps from a command terminal for each existing game that has already been set up and prepared for execution:

  a. Enter the command: game - this puts the terminal in the $JTLSHOME/game directory.

  b. Enter the command: cd <scenario_name> (for example, cd wespac51) - this puts the terminal in the game/<scenario_name> directory.

  c. Enter the command: mkdir cbtsys_summary. This creates the necessary new directory.

2.0 ENGINEERING CHANGE PROPOSALS

The following model capabilities were added to JTLS-GO 5.1.9.0 as a result of implementing authorized Engineering Change Proposals (ECPs).

2.1 JTLS-2020-14816 Save JTOI Terminal Output To Debug File

Summary of Model Change Request

When the JTLS-GO Transaction Operational Interface (JTOI) is started and run, a number of messages are written to the console. This console output can be extremely useful when debugging issues. While this console remains open if the JTOI crashes or is exited, automatically saving the output to a debug from the console would allow searching.

Design Summary

Whether the JTOI is started from the JTLSMenu or the JavaMenu, it is run in a console window. Both programs were modified to copy the output to a time-stamped file in the $JGAME/<scenario>/debug/jtoi directory.

2.2 JTLS-2020-14824 Cancel Tactical Group Formation Move

Summary of Model Change Request

Currently, the Player may use the Move order to move a group of ground units in a tactical ground formation (TGF). The order must specify the lead unit and provide a list of the following units. Once the order is sent, the units remain in the TGF until they reach their destinations, and then the TGF is disbanded. The user cannot cancel and disband the TGF prior to that, but the user can, for example, send a Defend Order to each unit in the TGF directly to execute “Now”. In this case, the unit will leave the TGF and execute the order. This is not an efficient way to disband the TGF.

During the JTLS-GO 6.0 Beta Test, a tester requested that a capability to cancel a TGF move be added. That capability has been added to JTLS-GO 6.0.0.0 and has also been implemented in this version.

Design Summary

The Manage Land Unit Tasks order is used to cancel the TGF move. The user must specify the following on this order:

1. Unit: the lead unit
2. Manage Task Options: “Cancel Tasks”

3. Task Number: 0 (the task the lead unit is executing now)
4. Cancel Options
    a. If “Single Task” is selected, nothing else is required
    b. If “Multiple Task” is selected, enter the Last Task Number.

Only the lead unit can be specified. If a user attempts to cancel the currently executing task for one of the following units, the order will not be executed. A message is sent to the WHIP Message Browser explaining that the task cannot be deleted.

Given 1, 2 and 3 are entered as above, canceling multiple tasks beyond task number 0 will only be applied to the lead unit. The following units in the TGF will only have their task number 0 canceled.

When the order is received by the CEP, and it meets all of the criteria above, the units will be removed from the TGF. The CEP takes into account the following states each unit can be in when the TGF is disbanded:

- State 1 - The unit is moving.
- State 2 - The unit is waiting for one or more units to catch up because the maximum lag distance threshold has been exceeded.
- State 3 - The unit has stopped because it is out of fuel.
- State 4 - The unit has stopped because it has encountered a barrier it cannot circumvent.
- State 5 - The unit is delayed because of artillery fire or air mission attack.
- State 6 - The unit is attacked by another unit and is in combat.

When a unit is removed from the TGF, its TGF-related tasks is removed. The unit stops, its posture changes to defend, and the route it was given for the TGF move is removed.

- If there are no other tasks waiting in the task list, the unit will remain at its current location awaiting an order from the Player.
- If there are other tasks in the task list, the unit schedules and executes the next task.

Under both circumstances, a message is sent to the Message Browser stating that the unit has left the TGF.

The lead unit is the last unit removed and when that occurs, a message indicating that the TGF no longer exists is sent.

3.0 SOFTWARE TROUBLE REPORTS

This chapter summarizes Software Trouble Reports (STRs), which describe software code errors that have been discovered by JTLS-GO users or developers and have been corrected.

3.1 JTLS-2020-14806 Unnecessary Updates to ELS Templates

The Alter Database section of the Scenario Initialization Program (SIP) included functions to Create and to Update templates used by the Entity Level Simulation (ELS). The update-function was intended to modify existing templates when there was a change in the composition of a prototype. Specifically, if a combat system (CS) was renamed, or if the TOE of a CS was changed in a prototype, then changes were also required in the corresponding ELS template. However, some ELS templates were unexpectedly updated when this function was used.

The Update function in the SIP was updating templates when no database changes had occurred in the prototypes or in the combat systems. These updates were associated with the crew counts in the templates. When a template is created, the program automatically creates a physical position for each of the crew required by a combat system. These crew positions are based on the required crew counts, and not based on the TOE of crew in the prototype. If the prototype is given a TOE for crew which is less than the number of required crew, then the template was being updated to remove the extra crew positions.

The code was modified to stop updating the template positions for combat systems which represent crew. This change results in templates which have an appropriate number of crew positions to man all of the combat systems. If the user adds crew during run-time, the template can now support the additional systems and mount then to the associated equipment.

3.2 JTLS-2020-14807 Fire Weapon Task Cancel Reason Incorrect

An OAS mission was ordered to attack a radar-controlled SAM target that was not emitting. The model assigned a weapon load consisting of anti-radiation missiles (ARM). When the mission reached the launch location, the Fire Weapon task was canceled by the model and a player message stated "No remaining weapons are available to expend."

The model canceled the Fire Weapon task because the target was not emitting. The reason given in the Player message was misleading. To fix the problem, the task cancellation message was expanded to produce a message specifically for this situation: "The mission is not carrying the right weapons to use. The weapons are anti-radiation (AR) type and the specified target is not emitting. AR weapons cannot be expended on a non-emitting target."

3.3 JTLS-2020-14808 Clear Button For Map Search Tool

The map search field does not have a clear button. The user is required to manually delete the entered search string. This was true for the WHIP, DDS, and JOBE. There was no option to clear the entry with one button.

A clear button was created to clear the search field.

3.4 JTLS-2020-14809 CAS Mission Did Not Expend Weapons - No ROE

An OAS mission order was submitted with the CAS Allowed flag set to YES. The mission's Air-to-Surface ROE was the default Hold Fire setting. When the mission reached its orbit location, a CAS Request order was sent. The model selected the mission and flew it to the weapon release location. However, the mission failed to deliver any weapons because its ROE was not set to Weapons Free. Because the mission was responding to a specific, positive request for ground support, the weapons should have been expended regardless.

New logic was added to the routine that finalizes the CAS assignment. If the Air-to-Surface ROE of the assigned CAS mission is set to Hold Fire or No Fire, the logic will temporarily change the ROE to Weapons Free (using the global Colocated Distance as the delivery range) so that the weapons will be expended. If the mission returns to its orbit location, its ROE, as always, reverts back to its home squadron's ROE.

3.5 JTLS-2020-14810 Rail Transporting Units Incorrectly Canceled

A rail convoy was sent to transport a unit. Prior to the resourcing event, the model computed that 12 rail cars were needed. A combination of three different rail cars was used by the convoy. When the rail convoy formed at the transport unit, the model determined there were not enough rail assets. There had been no changes to the unit from the time of the convoy composition and the train attempted to pickup the unit.

There was a difference in the unit's asset allocation over the rail cars. The model now uses the same asset allocation algorithm when composing the convoy and when loading the convoy.

3.6 JTLS-2020-14811 ARM Did Not Damage Fire Control Sensor

An OAS mission was directed to attack an emitting SAM target. The mission, consisting of two aircraft, was carrying a total of eight ARMs. All eight missiles were launched (in pairs at 15 sec intervals) against the SAM target. Although the target was suppressed for over 15 minutes, none of the missiles destroyed the SAM target's fire control radar. Only secondary (blast) damage occurred in the vicinity of the SAM target.

Faulty logic that determined whether or not the ARM actually hit the SAM target always concluded the missile missed, which prevented any direct damage against the target's sensor. Consequently, only secondary damage effects occurred.

The problem originated in three routines executed earlier that did not properly allocate the ARM's damage against the functioning fire control sensor. Instead, the weapons were improperly allocated against the SAM target's launchers. Normally within a database, AR weapons are most lethal against emitting sensors and not against launchers. Because the ARM's probability of hit was zero against launchers, the model concluded the missiles missed the SAM target entirely and computed only secondary blast damage.

To correct this problem, new logic was added in the three faulty routines to properly initialize the attributes that allocate the AR weapons to the functioning fire control sensor. Because the weapons have a positive probability of hit against the sensor, the model now determines that some hits are assessed against the SAM target and the logic allocates damage against the sensor based on the probability of kill.

While investigating this issue, it was discovered that a SEAD mission would not re-engage a SAM target after its suppression period had ended. The problem occurred because the fire control sensor effects (known as “tags”) were removed from the surrounding grid effects sets unnecessarily. The SEAD mission therefore did not recognize the newly functioning SAM target as a threat. To fix this problem, the code was corrected to not remove the tags.

3.7 JTLS-2020-14812 Targetable Weapon Pointer Change

An STR released with JTLS-GO 5.1.8.0, JTLS-2020-14771, incorrectly used some new code from JTLS-GO Version 6.0 and defined a Targetable Weapon variable as a pointer and not an integer. Although this oversight did not cause any functional errors in the computational logic, any unexpected crash in the same routine for another reason would be difficult to troubleshoot because a snap checkpoint and debug would not be possible.

The Targetable Weapon data type was changed from pointer to integer as used in JTLS-GO 5.1.

3.8 JTLS-2020-14813 Destroyed Transport Unit Convoy Crash

A truck convoy transporting a ground Unit was attacked by an air mission. Every truck in the convoy was destroyed, but the transported Unit was not wiped out. A new Transport Unit order for the Unit was submitted and rejected by the model, incorrectly stating the Unit was still being transported. A Situation Report was then requested on the Unit and the CEP crashed.

The Situation Report logic attempted to access the transported Unit's destroyed convoy to report the Unit's status, which caused the crash. However, the transported Unit should have been destroyed when its convoy was destroyed. The clear intent of the logic was to eliminate the Unit when the convoy was destroyed, but it failed to remove the Unit from the game. The Unit did suffer losses from the attack, but the damage was not enough to breach the wiped out threshold.

New logic was added to reduce the Unit's supplies and combat systems to zero to ensure it is wiped out when the convoy is entirely destroyed.

While investigating this problem, it was discovered that when a convoy's movement progress is delayed by an attack, the Next Move Time did not change in the Sitrep window. New logic was added to update the delayed convoy's Next Move Time in the Sitrep.

Also, when a convoy was partially destroyed, the transported Unit's strength was not updated after unloading until the next hourly update. This situation caused a Unit to remain at 96% after unloading for up to an hour, even though it had lost most of its combat systems and supplies. New logic was added to recalculate the Unit's weighted strength immediately after unloading from the convoy.

Users should note that losses to transported Units while en-route are assessed and tallied in the model, but do not appear in the IMT until after the Unit is unloaded.

3.9 JTLS-2020-14814 CEP Crash Modifying Mission in Attack Package

During the JTLS-GO 6.0 Beta test, an operator submitted an order to modify an air mission which was part of an Attack Package. The user was attempting to add additional attack tasks to the mission. Changes to the egress route were also desired, but not implemented due to this crash.

This crash occurred because the code was attempting to check if the extra attack task was allowed to be added to the package, but this check was done too late in the process to be effective. New verification checks were added to test the validity of adding new tasks to an Air Mission Package mission. Additional consistency checks were added to control the use of ingress and egress routes for air mission packages.

In all cases, only those packages using Time on Task and Reference Point options for their timing were allowed to use ingress and egress routes. For ground Alert, and for Airborne-On-Call packages, package ingress or egress routes cannot be used because the package has no predetermined destination or primary target.

3.10 JTLS-2020-14815 DRM Data Parameter Pages

The generated individual PDF documents for each data parameter described in Appendix B of the JTLS-GO Data Requirements Manual had all disappeared. These individual pages are accessed through DDS Client's tables as help options. Without them, a DDS user would not be able to get detailed description of the data fields.

This problem occurred because the documentation suite started to use a new version of Adobe FrameMaker. The program, the drmsplit program, used to create the individual parameter Portable Document Format (PDF) files was not robust enough to handle the new changes. To fix the problem, the drmsplit program was rewritten to be more resilient to changes when handling and processing the DRM PDF file.

3.11 JTLS-2020-14817 LC2IS Message Service Testing Issues

During C4I testing with the North Atlantic Treaty Organization (NATO), a number of minor issues were discovered and fixed in the LC2IS Message Service (LC2MS).

- In the equipment section of a unit, the only items that were appearing were personnel and some miscellaneous systems.
- LC2IS does not like the unit description data of a combat unit with a force of FA (Field Artillery), specialty of FAHOW (Howitzer), and mobility of LNDTRC (indicating self propelled).
- When reporting the country code of a unit and its personnel, if the country is an “exercise” country (non-existent in the real world), the NATO Digraph should be used, rather than the NATO Trigraph. Currently JTLS always output the NATO Trigraph. If the Trigraph begins with an X the country represents an “exercise” country and LC2MS retrieves the NATO Digraph instead.
- The LC2MS was incorrectly including only systems that did not have a RIC code in a unit's equipment list. This was changed to include systems that do have a RIC code.
- In terms of the XML to describe the self-propelled howitzers the LC2MS is following the standard specified in the LC2IS documentation. The task of informing LC2IS of this mismatch was assigned to JWC. The code was modified to remove the mobility so that LC2IS will parse the XML.

3.12 JTLS-2020-14818 Combat System Special Subcategory Renaming

Combat System Special Subcategory data was not properly updated when the related Transportation Class or Small Boat data was renamed.

The Transportation Class and Small Boat renaming related stored procedures were modified to properly update the Combat System Special Subcategory data when a related Transportation Class or a Small Boat record was renamed. Reloading a user scenario will automatically create the modified renaming stored procedures.

3.13 JTLS-2020-14819 Unit in Multiple LOGFAS Profile Structure

A unit appears in two separate LOGFAS Profiles.

Units are assigned to LOGFAS Profiles individually (if they have no subordinates) or as part of a Command Structure. If a new LOGFAS Profile is created, which is a lower subset of an existing profile, all those units would be part of two profiles. SVP Error 167 has been add to identify units that have this problem.

3.14 JTLS-2020-14820 Errors in Satellite Orbits From JSAT

The JSAT program was used to generate orbital data for a large number of satellites. The orbital data included positions for the satellites over extended periods of time (multiple days). However, the resulting routes were incomplete when the satellites were imported into the CEP. Subsequent attempts to add points to the routes using the National Asset Pass Order did not produce any additional orbit points. After restoring from a checkpoint, the satellite data caused a crash associated with an attempt to decrease the simulation time.

A number of code errors were found and fixed. The orders to add points to an existing orbit were not processed correctly. The order was filed in a pending set, but never removed and executed from that set. When restoring from a checkpoint, the code was not reading the saved data correctly. A section of the code was missing, such that the Faction owning the satellite was never restored.

Finally, the reported attempt to decrease the simulation time was found to be associated with an uninitialized Starting Time for the satellite. When an order field was not initialized, the value for that field is negative one. This negative value resulted in a bad start time for some of the satellites which were scheduled to begin after time zero. When this value was correctly set, the model was able to successfully restart with satellites that begin orbiting after the start of the game.

3.15 JTLS-2020-14821 Mission Has Two Intercept Tasks

An air mission was observed to be holding two intercept tasks. This is not a legal configuration.

The issues causing this situation were very complicated, but they were all centered on the fact that an alert mission was sent to intercept an enemy mission and as soon as the intercept started it was canceled because of unequal odds.

Several changes were needed to completely solve the issue:

- A mission on alert was sent to intercept an enemy mission. To do so, it needed to take off. The take off logic placed the mission back in the available interceptor set. It was not available because it was already assigned but had not taken off. The logic was expanded to account for this situation.

- A mission that was currently on the ground executing either a Refuel Task or a Rearm task was being selected as a possible interceptor. This situation was stopped. The mission cannot be considered unless it is actually on alert.

- When a mission being intercepted wants to run away it postpones its current task. The postpone logic determined that the mission was currently not executing a task. The issue was the mission had a Move Task schedule to start in one second. The postpone logic basically said, since nothing is currently executing, the postponement was successful. The postponement was not successful.

  The logic needed to cancel the “execute mission task” event for the non-executing move. Once that was done, then the postponement was accomplished successfully.

- The true cause of the entire issue was that an interceptor was considered legal even though on its first move, it decided to cancel the intercept because of uneven odds. Before assigning the interceptor the model ensures that an interceptor will not be assigned if it will immediately come to the conclusion to stop the intercept.

3.16 JTLS-2020-14822 LOGFAS Update Service Fixes

The LOGFAS Update Service was mishandling Combat Systems and Supply Categories by using old data relationships that no longer apply.

The LOGFAS Update Service now handles Combat Systems and Supply Categories using the current data relationships.

3.17 JTLS-2020-14823 LOGFAS Naval Unit With No RIC

All Naval Units that are part of a LOGFAS Profile must have a RIC.

A new SVP Error has been added to notify the database builders when the situation arises.

3.18 JTLS-2020-14825 Vulnerable Attack Package Needed Exit Point

When a player created an attack package without an egress route, the missions in the package became vulnerable immediately after their weapons were fired. The package flew to the target location and released weapons as expected.

However, because the package had no egress route, the package ended while the missions were still located in enemy territory. The escort and SEAD missions did not provide support to the OAS mission because the package tasking was finished. This resulted in frequent losses of package missions right after they expended their munitions in their primary task.

When an egress route was built into a package, the missions flew that route before they withdrew from the package. When there was no egress route, the package ended prematurely. The code was modified to add a move task to packages which are created without egress routes. This supplemental task directs the package to return to the rendezvous point before the missions withdraw from the package. This allows the supporting missions to remain with the attack mission until they return from enemy territory.

3.19 JTLS-2020-14826 Cruise Missiles Database-Specified Altitude

It was possible for a cruise missile to not fly at its database specified altitude. If a cruise missile is launched from an air mission or launched from a target with no specified orbit point, the missile would always fly in altitude zone 1 instead of the database specified altitude.

The following corrected logic was implemented:

- For cruise missiles launched from an aircraft, the missile will fly at the air mission altitude for one move and then fly the remainder of its way to the target at the database-specified altitude (TW FLIGHT ZONE).
- For cruise missiles launched from a target and no orbit point was specified, the missile will have one move in Altitude Zone 1 and then fly in the database-specified altitude zone.

3.20 JTLS-2020-14827 Alert Mission Repeated Intercept And Break-off

A DCA Alert mission (with Manual Pair not required) repeatedly launched to intercept an enemy mission, only to immediately break-off due to Unequal Odds. After breaking off each time, the interceptor returned to base and went back on Alert. This behavior repeated multiple times against the same enemy mission.

Before a mission is assigned to intercept, the logic assesses the threat associated with the enemy mission. Part of this assessment includes a behind-the-scenes look at the enemy mission's ROE against the perceived Force Side of the interceptor.

While the interceptor is on the ground, the enemy perceives the Force Side of the interceptor to be Unknown with no ROE set which allows the mission to launch to begin the intercept (if other criteria are also met). With each move, the interceptor again assesses its chances, but now the enemy perceives the actual side of the interceptor. If the ROE are Weapons Free against the interceptor's side, a Measure of Effectiveness is calculated based on the PH/PK values of the weapons carried by both missions.

If the MOE is less than the AC CONTINUE ENGAGE MULT attribute for the interceptor's aircraft type, the interceptor immediately breaks off. Once back on alert, the mission is made available again and the process begins anew because the first assessment is based on the Unknown side which usually has no ROEs set.

To prevent the interceptor from launching unnecessarily, the logic was changed to use the interceptor's actual Force Side when checking the enemy's ROE settings, but only if the interceptor is on the ground. Once airborne, the process functions as before: The interceptor's perceived Force Side from the enemy's perspective is used to check ROE settings.

While testing the change, it was discovered that an alert mission considered and rejected for an intercept by the model may remain on alert, but never intercept again, even if a different enemy mission presented itself. The unusual situation that caused this issue to occur was corrected.

3.21 JTLS-2020-14828 Execute Air Move Task Already Executing

An OAS package mission was forced to withdraw from the package due to a lack of suitable weapons. After the mission started to move towards its home base, a logic error message was generated because the model attempted to execute the Move task that was already executing. The mission continued and landed normally.

The problem was related to an improper handling of a mission when it withdrew from a package under normal circumstances. The logic canceled the mission instead of simply removing the mission from the package. The cancel process generated an Execute Mission Task event for the Mission Complete task, which turned out to be redundant because the event was also generated at the end of the Execute Mission Task routine. The cancel mission logic was removed to solve the problem.

While testing this solution, a "Non-Viable Air Mission Package" logic error was generated when the OAS mission was forced to withdraw due to aircraft loss. The damaged mission was the last OAS mission in the package, so the package was no longer viable by definition. However, the viability check was not made when the OAS mission left the package. A minute later, when the support missions in the package attempted to move there was no OAS mission in the package; therefore a location centroid could not be calculated. The model generated the logic error and self-corrected by canceling those missions and the package.

To avoid the logic error, code was added to immediately check the package viability when a mission leaves the package due to aircraft loss. If the package is no longer viable, all remaining support missions are removed from the package.

3.22 JTLS-2020-14829 Logic Error Situation Still Caused Model Crash

During internal testing of JTLS-GO 6.0, the model crashed attempting to initialize a combat system to the JODA. The model caught a logic error, but did not properly handle the logic exception. Within JTLS-GO 6.0, the error that caused the generation of the Logic Error was found and fixed.

The issue of this STR has nothing to do with the actual code issue found and fixed within JTLS-GO 6.0. The problem is that when a Logic Error exception is caught, the exception is supposed to be corrected internally to stop the model from crashing. Although the JTLS-GO Engineering Team knows of no way that this Logic Error can be triggered within JTLS-GO 5.1, it was still important to ensure that if the exception is caught that the model properly corrects the issue so the model does not crash.

The code was changed so if the Combat System initialization routine caught such an exception, the exception is properly reported and corrected internally to stop the model from crashing.

3.23 JTLS-2020-14830 JOBE Crashed Upon Launching

The JOBE threw an exception during launching and could not start up.

The new DJINN, a component of the DDS client, incorrectly wrote FARP units' XML element tag as “Farp” instead of “FARP”. The DJINN was fixed to write the correct tag for FARP units, as defined in the JOBEDynamic.xsd schema. The JOBE is also modified to accept both “Farp” and “FARP” tags.

3.24 JTLS-2020-14831 Order Graphics Did Not Close Polygons

The order graphics did not close a connecting polygon field by adding the closing edge from the last point to the first.

The closing edge was added to connected polygon fields in the order graphics.

3.25 JTLS-2020-14832 ATOT Refuel Chits For COMAO Mission

Translating an ATO message with the COMAO option selected produces order groups having the name of each COMAO described in the message. Each COMAO group should contain the mission orders that belong to the COMAO of the group. However, refuel chits for the COMAO missions are placed in a different group.

The ATO Translator was modified to include the refuel chit orders of COMAO missions within the COMAO group by the same COMAO name. This is only true if the COMAO option has been selected from the ATOT configuration prior to translating the ATO. Otherwise they are placed in the order group specifically for refuel chits of a particular time slot.

3.26 JTLS-2020-14833 Order Pattern Checking Not Enforced By OVT

Text fields on order panels can have a pattern that they must match. These text field patterns are defined in the order's XML file. The WHIP does not allow the user to type in a character that does not match the pattern, and provides the user an indicator by turning the field label red when an entry is not complete.

However, during JTLS-GO 6.0 testing, it was discovered that a user could backspace an entry in the field, leaving it incomplete, and this was not noticed by the WHIP. The user could also cut and paste an illegal character into the field. Neither activity was caught by the WHIP and the order could then be sent to the CEP. The OVT, which is responsible for catching all order format related errors, was not performing any checking on the pattern matching field.

Attempting to find all the ways a devious user might be able to bypass the order entry helper functions of the WHIP is outside the scope of the WHIP's responsibilities. Ensuring that all checks defined in the order XML file are enforced is clearly the responsibility of the OVT, and it was not performing this critical check. The OVT Library was modified to read, process, and enforce any pattern matching fields defined for text fields.

It was discovered that the pattern syntax being used in the order files was not entirely accurate. Though the syntax worked for the WHIP's helper functions it was not accurate enough for the OVT to strictly enforce the desired pattern. Therefore the patterns used in the order files were modified to enforce a strict syntax of what was desired.

3.27 JTLS-2020-14834 Convoy Moves In Range Of Enemy Error

When a convoy moves through an area covered by enemy units, it is possible that the enemy will fire on and destroy the convoy. If this happens and the convoy is destroyed, a logic error was generated because all proper data structures were not cleared when the convoy was destroyed under these circumstances.

The surrounding object set is now cleared if the convoy is destroyed by one of the enemy units surrounding the convoy.

3.28 JTLS-2020-14835 ATOT Error For Displaced Squadron

Whenever an ATO message calls for a mission to depart from the wrong airbase (ICAO), the ATO Translator gives an error and erroneously reports that the mission is canceled. In this case, the associated squadron from the ATO message has been linked to a valid, active JTLS-GO squadron, but not at the provided ICAO.

Although the error report properly gives the circumstances of the problem, it does not cancel the mission and continues to generate an order for the mission. The end user decided that is was operating as desired and so the reported error was changed to a warning. The new warning message includes the current ICAO location of the selected squadron.

3.29 JTLS-2020-14836 Runaway Mission Continued Flying Indefinitely

A runaway Reconnaissance mission continued to fly away from an interceptor mission that was blocked by a SAM threat (A "No Feasible Intercepting Location" Alert was displayed). The runaway mission continued to fly far beyond the blocked interceptor's weapon range and remaining fuel range.

New logic was added to determine if the runaway mission should continue moving further away based on the proximity of the interceptor mission. Two checks were added:

- If the runaway mission is within the maximum weapon range of the interceptor, the mission will continue to fly away.
- If the mission is no longer within maximum weapon range, the model calculates:
    a. The interceptor's remaining flight time based on its speed, remaining fuel, fuel consumption rate, and distance to home.

b. The distance the interceptor could fly toward the runaway mission in the remaining flight time is calculated.

c. The distance that the runaway mission could fly during the interceptor's remaining flight time is calculated.

d. If the distance the runaway mission could fly, plus the current separation distance, is less than the distance the interceptor could fly, plus its maximum weapon range, the runaway mission will continue to fly away.

   Otherwise, the mission will hold at its current location while maintaining a Runaway posture.

This new logic stops the runaway mission from flying further away from the interceptor when it reaches a “safe” distance.

While testing these changes, two problems were found with multiple intercepting missions:

- When more than one mission was intercepting the same runaway mission, the escape route calculated by the model sometimes caused the mission to fly toward its interceptors. A flaw in the logic dealing with bearing calculations was corrected.

- When one of the intercepting missions broke-off, the runaway mission stopped flying away from the remaining interceptors and attempted to return to its original tasking. To correct this problem, a check was added to prevent the runaway mission from canceling its runaway task if it is still being intercepted by another mission. This correction causes the runaway mission to continue flying away from the remaining interceptors.

3.30 JTLS-2020-14837 Off Sensors Making Targeting Detections

Some weapons require the delivering air mission to actively detect the target prior to firing. The mission's sensors are checked, but even if a sensor is turned off, it is allowed to detect. This makes it impossible to simulate a non-operative sensor.

The code was changed to check whether a sensor is turned on prior to allowing it to accomplish the needed detection. The default visual sensor for the mission is always assumed to be on.

3.31 JTLS-2020-14838 Firing On Specific Ship TTG Assignment

A user submitted an order to fire on a specific Naval Unit. The user's order specified a TTG list, but the list did not include the Unit's Ship Unit Prototype. The mission refused to fire on the ship and canceled the task.

There were two situations that needed to be corrected:

- If the Fire Weapon task has a specific Unit identified and that unit is a Naval Unit, the model has been changed. The Ship Unit Prototype (SUP) does not need to be in the air mission's Target Type Group (TTG) list. The model now assumes that the user-specified ship should be hit.

- If the Fire Weapon task has only specified a location, the model was corrected to only consider objects for which the air mission has ROE and for which the object satisfies the restrictions created by the air mission's TTG list.

3.32 JTLS-2020-14839 Air Mission Package Centroid Inaccurate

The computation of the location at which Air Mission Package Escorts and SEAD missions should be located was not as accurate as it could be.

There were two problems identified:

- The computation used a less accurate method of division.

- If the OAS package is conducted close to the International Date Line, where there were OAS missions on each side of the line, the computed location was not correct.

Both issues were corrected.

3.33 JTLS-2020-14840 Check Orders For Incorrect Group Definition

Some saved orders with group definitions failed to be recalled properly in the WHIP, and did not display the entry fields for the saved group selection when opening the saved order. This occurred because the order description had groups defined in an incorrect order.

The order checker was corrected by defining parent groups before any of their child groups are defined. Once the order checker uncovered the issues, the problem was solved for each of the offending orders.

3.34 JTLS-2020-14841 Addressed Fortify Issues

Fortify identified some code that was susceptible to XML External Entity Injection attacks.

The identified code was modified to protect against XML External Entity Injection attack.

3.35 JTLS-2020-14842 IIR Message Information

In the Imagery Interpretation Report (IIR) message, the USMTF format contains a record of data labeled the STATACT data. Here, STATACT refers to the Status or Activity of the object contained in the report.

The STATACT line of data contains three pieces of information: the primary STATACT, the secondary STATACT, and the Imagery Change Significance. These three data fields were not accurately filled by the model. The values were hard-coded to use a primary STATACT of OPR, for operational, no entry for the secondary STATACT, and a significance of NEW for a new detection. These hard-coded values were not suitable for damaged, or later reports on detections.

New code was added to the CEP to determine good values for the STATACTs of objects in the IIR message. Changes were also made to establish the significance of the reported imagery. These changes are reflected in the STATACT record in the USMTF versions of the IIR message.

3.36 JTLS-2020-14843 Submarine Stays On Surface After Leaving Port

A surfaced submarine in port was ordered to move into open waters. The submarine left the port and remained on the surface, despite entering an ocean grid that was deep enough to accommodate the submarine's operational depth.

There was no existing logic in the CEP to cause a submarine to submerge after it leaves port. A new routine was written that creates a Submerge Task scheduled to occur after the submarine leaves port. This time frame allows detection of the surfaced submarine while traversing the harbor. After this time frame has expired, the submarine will attempt to submerge to snorkel or operational depth, depending on the depth of the grid. Otherwise, it will continue to move on the surface until it enters a grid with sufficiently deep water.

While testing this new routine, it was discovered that a submarine did not automatically submerge when entering an ocean grid with deeper water. Likewise, a submarine at operational depth would not automatically snorkel or surface if the water was too shallow. A logic sequence error was found in the routine that moves the submarine that prevented a depth check each time the vessel entered a new terrain grid. The sequencing error was corrected.

Also while testing, it was discovered that a submarine that is magic moved near a port would enter port status as expected, but would not surface. Missing logic was added to force the submarine to surface if it entered a port as a result of a magic move. Similarly, when a surfaced submarine in a port was magic moved to a deeper grid, it did not change depth as appropriate.

3.37 JTLS-2020-14844 Mission Specified Altitudes in Orbit Path

A user sent a Reconnaissance Mission with an Orbit Type of Orbit Path. Each route point location in the path was assigned a different enroute altitude. The Mission traveled to the first route point at the correct enroute altitude. However, it did not travel at the enroute altitude for the second route point in the orbit path. Instead, the mission continued to fly at the enroute altitude of the first route point.

Whenever CHANGE.MISSION.ALTITUDE was called, every route point in the Mission's route was updated to reflect the new altitude change. Therefore, when following along an orbit path, the

originally assigned altitudes would be overwritten. Changing the altitude of every route point in the Mission's path is necessary when a user sends an order to change a Mission's altitude.

We added a new argument to the routine CHANGE.MISSION.ALTITUDE called “FULL.CHANGE” which indicates if we should change just the Mission's altitude or the Mission's altitude along with the altitude of the route points in the Mission's route. Each routine where CHANGE.MISSION.ALTITUDE is called has been examined and now passes the proper FULL.CHANGE argument.

3.38 JTLS-2020-14845 JXSR Subscriptions For Alerts Expire Too Soon

Subscriptions for Alerts are expiring before the WHIP can request an update for the alert data. This causes the WHIP to start a new request for the same alert data.

Subscriptions for Alerts had a life span that was hard-coded to 10 seconds. This amount of time, together with the 10 second request period for the WHIP, does not allow sufficient opportunity for the WHIP to request an update before the subscription gets removed.

The life span time for alerts has been changed to the same value used for other subscriptions, which is a configurable value in the ICP for the JXSR. The default is 300 seconds (5 minutes).

3.39 JTLS-2020-14846 Crash When TGF Unit Given Engineering Task

The model crashed when a user gave an Engineering Task order to a Tactical Ground Formation unit, and the unit could not leave the TGF.

The order was rejected, and the model incorrectly destroyed the order during the rejection process and again when the routine was finished with the order. The logic was changed to notify the user during the rejection process, but only destroy the order at the end of the order processing routine.

3.40 JTLS-2020-14847 Duplicate DSAs For Air Mission

After assigning a DSA to a Reconnaissance Mission, it was possible to assign the same DSA to the Reconnaissance Mission multiple times using the CHANGE.AIR.MISSION.PARAMETER Order.

A DSA can no longer be assigned to an Air Mission more than once. After sending a CHANGE.AIR.MISSION.PARAMETER to modify a Mission's DSA List, the user will receive a detailed message describing the status of DSA. If the DSA already existed, then the user will be notified accordingly. The message also displays the Mission's current DSA List.

3.41 JTLS-2020-14848 Invalid OTH Gold Command Levels

Echelon names passed from JTLS-GO to C4I systems are not checked for validity.

There is a distinction between what JTLS-GO calls Echelon Level and what many C4I devices call Echelon Level. The Army and all army symbols refer to Echelon when they are discussing Divisions, Brigades, and so on. The Army's reference to Echelon is well understood and used to determine the echelon symbols at the top of JTLS-GO symbol icons. The problem is that the Army Echelons do not apply easily to the Air Force and definitely not the Navy. So, for C4I devices, there is an extended list of Echelons that covers the concepts needed by all Services.

We did not want to refer to these as Echelons in JTLS-GO, because then we would have to define the symbols to place on top of our icons, the majority of which would be blank. Our decision years ago was to create a new concept in JTLS-GO called Command Level. So every unit has a Command Level, and a Command Level has the two attributes important to this discussion:

- Icon Echelon (Short Army List)
- OTH-Gold Echelon (Long C4I List)

When building a database, the Icon Echelon points to a Lookup table and may be null, which means no echelon marking should be placed on the icon.

The OTH-Gold Echelon must be filled, but JTLS-GO did not have a lookup table for the attribute; it is simply a text variable. This caused a problem when feeding a real-world system in that database builders were not entering legal values. Unfortunately this error was discovered after the JTLS-GO Version 6.0 database structure was closed. This means the issued cannot be properly corrected until the release of JTLS-GO 6.1.

Until then, SVP Error 122 is generated when the OTH Gold Name for a Command Level is not in the C4I long list.

3.42 JTLS-2020-14849 Order Graphics Duplicate Points Across Utilities

The Offensive Air Support order graphics did not display the mission ingress route points set in the utility whenever the mission's tasking utility used the same location points to instruct the mission's return path to its home squadron. This occurred because the graphics did not previously allow points to be duplicated across utilities within an order.

The graphics were corrected to allow similar location points on different utilities in an order, and now properly connects all the order's utility data.

3.43 JTLS-2020-14850 CEP Crash When Taking Checkpoint

When the CEP took a checkpoint, it crashed because the $JGAME/<scenario>/cbtsys_summary directory was accidentally removed.

A check was added to ensure the $JGAME/<scenario>/cbtsys_summary directory exists before the CEP starts or restarts. If the cbtsys_summary directory does not exist, the CEP launch script

generates the directory. The directory could be missing if the Technical Control staff prematurely stops the checkpoint process for any reason.

3.44 JTLS-2020-14851 Invalid XML Format In Controller Damage Report

The Controller Damage Report (Message 7300) did not properly generate when a ship was damaged by a minefield.

The TARGET.WORKING.SET was not being cleared at the proper time in the Mine Damage Ship routine. The call to clear the set was moved up in the code and retested successfully.

3.45 JTLS-2020-14852 Unit Automatic Move Error

Some logic errors were generated by the model that reported that there were leftover line grids at the end of the Assess Weapon Damage event.

An internal collection of grids was used by the model to determine the path of a moving unit. Here, the model was attempting to automatically move the unit to a new firing position. The collection (set) of grids was used to check if each move along the path of the unit was legal. At the end of this check, the model never cleared the contents of the collection which caused the logic error. The code was changed to properly clear this set.

3.46 JTLS-2020-14853 TGF Unit Executing Move Task Error

The event EXECUTE UNIT TASK generated logic errors. These errors were generated when units in a TGF attempted to execute a new move task while it was already executing a task.

The logic errors were the result of the TGF's lead unit encountering a river that caused it to stop. The leader tried to move again by finding a way around the obstacle. This triggered an EXECUTE UNIT TASK event to execute a move task for the leader. This was fine when the routine dealt with a new TGF; however, for an existing TGF the followers already had an executing move task, so the routine gave each follower an additional (and unnecessary) move task. This led to the logic error.

The routine that attempts to build the TGF route was rearranged and a new task is only generated when the move is for a new TGF and not an existing TGF. Additional code was added to this block to prevent the followers from receiving the extra move/attack task.

3.47 JTLS-2020-14854 Package Mission Tank Refuel Error

An air mission belonging to an airborne on-call package ran low on fuel and began the process of selecting the best available tanker. While determining the amount of fuel needed based on the remaining tasks to execute, a logic error message was generated because one of the tasks did not have a location. This error did not cause any adverse effect to the mission or mission package.

The mission had been airborne on-hold, waiting for a time-on-target assignment. By design, all of the mission's pending execution tasks hold either a location or an associated object (i.e. a target to strike). Also by design, the Withdraw task at the end holds the package itself as the associated object. When the Withdraw task is executed, the mission leaves the package because its responsibilities have ended. When the model was calculating how much fuel was needed to refuel the mission from the tanker, it considered all the pending tasks, including the Withdraw task.

However, a package location is meaningless in the Withdraw task, so the model could not calculate the fuel required for that task, which caused the logic error message to generate. To prevent a recurrence of the message, the model was changed to skip the Withdraw task when calculating how much fuel is required.

3.48 JTLS-2020-14855 OVT Library Error With Illegal Characters

The JOVT program was throwing an error when a field contained an illegal character, but was marking the order as ready to send. If this order had actually been sent to the CEP, it would have resulted in a game crash.

There was a badly formatted XML response from the OVT Library, which is part of the OMA. When attempting to parse this badly formatted XML file, the JOVT verification thread would crash, but that crash was not being propagated into a bad order. The underlying cause of the badly formatted XML was a coding issue in the OVT Library that was causing random memory garbage to be loaded into the response. This has been identified and fixed.

3.49 JTLS-2020-14856 Amphibious Operations Inaccurate Estimates

When a player requested a time estimate for an amphibious operation, the resulting value was much different from the actual duration for the operation. The estimates were always less than the required time.

There were several problems with the code used to estimate the time to complete an amphibious operation. The time needed to load the equipment and supplies was not computed correctly. For boat operations, the time to perform a pickup did not include the time needed for some of the boats to arrive at the pickup location. Code corrections were made and the resulting estimates were found to agree with the actual times needed to complete the operations.

3.50 JTLS-2020-14857 MDP Large Numbers Not Printing

When the MDP was instructed to forward a Unit Parameters report, the result of a Set Individual Unit Parameter order, some of the supply data showed values of “nan” for the on hand, stockage objective, and reorder levels. "nan" is a representation meaning Not A Number, indicating that what the code encountered did not translate as a number.

It was separately noted that durations are being printed out with a total fractional day attribute followed by the hour, minute, and second component as in 00.14D01H00M00S instead of the integral days of 00D01H00M00S, and that the seconds portion is always displaying as 00S even when there is a non-zero second component.

The condition was only happening for some of the supply data, not all of it. The difference turned out to be whether the data was written by the CEP as a decimal number or whether scientific notation was used. When scientific notation was used, the XML library used to get the value did not recognize it. This code was changed to retrieve a string instead and use the C library function to convert the string to a number. The situation existed in both a UOM conversion and a supply UOM conversion.

It was also discovered that when dealing with locations the latitude and longitude were already using this new method, but the string was not being released after converting to a number. This represented a memory leak and was fixed.

The issue of the bad duration was located in a different library and was caused by a failure to initialize a flag to indicate that routing to integer numbers was desired.

The issue of omitting the seconds was a failure to copy the entire string into a working buffer, and if a string of fewer than 8 bytes was generated by the CEP, the issue would have resulted in a buffer overflow situation.

3.51 JTLS-2020-14858 JODA Restart Not Repopulating Convoy Assets

If the JODA needs to be restarted, convoy assets generate an error and do not populate the JODA. The game needs to be restarted to get the convoy assets back for viewing by any JODA client.

On a JODA restart, the model incorrectly believes that the convoy assets were already in the JODA and so an update was sent. This resulted in a logged JODA error. The code was changed to always assume the JODA does not have convoy assets when the JODA is restarted.

3.52 JTLS-2020-14859 SET.SHIP.UNIT.PROTOTYPE Message Error

When sending a SET.SHIP.UNIT.PROTOTYPE order, using the "SHOW ALL" and "SUBMARINES" parameter group for a SUP that is not an electric submarine, the message returned had errors. The supply category for recharge was written as "(Index 0)", and the UOM for the supply amount required to fully recharge the drained battery was written as "0 Error".

The Message Definition Files now uses the "No_Supply_Category" translation from the Dynamic Vocabulary File. When the message is printed, the Supply Category required to recharge the drain battery is displayed as "NONE".

3.53 JTLS-2020-14860 SITREP On Super WHIP Did Not Update

On a Super WHIP, the perception side for the SITREP was not consistent with the component perception that invoked the object information display. This prevented the displayed data from being updated.

This was corrected by setting the SITREP side perception to be the perception on the component that initiated the SITREP, and to update the display for that corresponding object's updates.

3.54 JTLS-2020-14861 Fortify Finding On XML External Entity Injection

A common library used by the WHIP, DDSC and JOBE map component was flagged as a Fortify finding for external entity injection. While the external injection was already disallowed, it had not been done correctly to satisfy the vulnerability finding.

A correction was implemented to prevent Fortify's external entity injections finding.

3.55 JTLS-2020-14862 Simscript Compile Scripts Allow Backup Files

Whenever a Simscript program is built using the compile scripts, the existence of backup files in the source directory prevents the program from being built completely.

The compile scripts for building object files for the jcomp and jpcompdir Simscript programs were failing to remove and exclude any of the backup source files having names starting with a tilde ("~"). These backup files often do not compile, and force the build script to fail for the remainder of the source files in the current compile directory.

These build scripts have been modified to first discover any backup source files with this naming format and remove them prior to compiling the remainder of the intended source files for the program.

4.0 REMAINING ERRORS

Every effort has been made to correct known model errors. All reproducible errors that resulted in Combat Events Program (CEP) catastrophic software failures (crashes) have been corrected. Other corrections were prioritized and completed according to their resource cost-to-benefit relationship.

As JTLS-GO 5.1.0.0 represents a major release of new functionality, remaining outstanding errors from the JTLS-GO 4.1 series and earlier have been considered to be obsolete and no longer relevant to JTLS-GO and have been removed from consideration for correction at this time. In future Maintenance releases, outstanding errors related to JTLS-GO will be listed in this chapter, with information provided regarding the extent of the error, as well as suggestions to avoid or minimize the effects of the problem.

4.1 DDSC – TMU Line Mode Changes Multiple Grids

When using the line mode in the TMU, more grids than the ones the line passes through are changed. This can also cause a warning about trying to change multiple layers to appear.

4.2 DDSC – Multiple Types In DDS History Table

If records for more than one table type are selected in the DDS History table, “History Details” will display details for only one type.

4.3 WHIP - Pipeline Not Shown On IMT

A pipeline being operated by a non-detected unit is not shown in the pipeline IMT.

4.4 DDSC/WHIP/JOBE - CADRG Map Zoom

When using the CADRG map projection, if the width of the map is less than the height the zoom tool does not work correctly.

4.5 WSM - Many Messages Cause Lockup

If a service produces a large number of log or error messages in a short period of time, it can cause the WSM to lockup.

APPENDIX A ABBREVIATIONS AND ACRONYMS

Terms are included in this Appendix to define their usage in JTLS-GO design, functionality, and documentation.

AAA	Anti-Aircraft Artillery
AAL	Air-to-Air Lethality
A/C	Aircraft
ACP	Air Control Prototype
ADA	Air Defense Artillery
AEW	Airborne Early Warning
AFB	Air Force Base
AG	Air-Ground (Air-to-Ground)
AI	Air Interdiction
AIM	Air Intercept Missile
AIREF	Air Refueling
AKL	Area Kill Lethality
AMMO	Ammunition
AO	Area of Operations
AOC	Air Operations Center
APC	Armored Personnel Carrier
ARECCE	Armed Reconnaissance
ARTE	Air Route
ARTY	Artillery
ASC	Automatic Supply Calculation
ASCII	American Standard Code for Information Interchange
ASW	Anti-Submarine Warfare
ATC	Aircraft Target Category
ATGM	Anti-Tank Guided Missile
ATK	Attack
ATO	Air Tasking Order
ATORET	Air Tasking Order Retrieve Program
ATOT	Air Tasking Order Translator
AWACS	Airborne Warning And Control System
AZ	Altitude Zone
BAI	Battlefield Air Interdiction

BDA	Battle Damage Assessment
BDE	Brigade
BN	Battalion
C3	Command, Control, and Communications
C3I	Command, Control, Communications, and Intelligence
C4I	Command, Control, Communications, Computers, and Intelligence
CA	Civil Affairs
CADRG	Compressed ARC Digitized Raster Graphics
CAP	Combat Air Patrol
CAS	Close Air Support
CAT	Category
CCF	Central Control Facility
CCP	Command Control Prototype
CCU	Controller Change Unit
CEP	Combat Events Program
CMDR	Commander
COP	Common Operational Picture
CP	Combat Power
CS	Combat System
CSP	Combat System Prototype
CTAPS	Contingency Tactical Air Planning System
CTG	Commander Task Group
CTRL	Control keyboard command
DCA	Defense Counter Air
DCL	Digital Command Language
DDS	Database Development System
DISA	Defense Information Systems Agency
DIV	Division
DMA	Defense Mapping Agency
DoD	Department of Defense
DOS	Days of Supply
DPICM	Dual Purpose Improved Conventional Munitions
DS	Direct Support
DSA	Directed Search Area

DTG	Date Time Group
EC	Electronic Combat
ECM	Electronic Counter Measure
ECP	Engineering Change Proposal
EEI	Essential Elements of Information
ELINT	Electronic Intelligence
ELS	Entity Level Server
EODA	Entity Level JTLS Object Data Authority
ETA	Estimated Time of Arrival
FARP	Forward Arming and Refueling Point
FLP	Fire Lethality Prototype
FLOT	Forward Location of Troops
FOL	Forward Operating Location
FWL	Frederick W. Lanchester (originated a differential equation model of attrition)
GAL	Gallon
GCCS	Global Command and Control System
GRTE	Ground Route
GS	General Support
GSR	General Support Reinforcing
GUI	Graphical User Interface
HARM	High-speed Anti-radiation Missile
HE	High Explosive
HELO	Helicopter
HMMWV	High Mobility Multipurpose Wheeled Vehicle
HQ	Headquarters
HRU	High Resolution Unit
HTML	Hypertext Markup Language
HTT	High Resolution Unit Target Type
HUP	High Resolution Unit Prototype
ICM	Improved Conventional Munitions
ICP	Interface Configuration Program
ICPLogin	Interface Login Program
ID	Identifier
IFF	Identification Friend or Foe

IIP	Intelligence Information Prototype
IMT	Information Management Tool
INFO	Information
INTEL	Intelligence
JCATS	Joint Conflict And Tactical Simulation
JDPI	Joint Desired Point of Impact (formerly DMPI: Desired Mean Point of Impact)
JDS	JTLS Data System
JDSP	JTLS Data System Protocol
JEDI	JODA Entity Data Identifier
JMCIS	Joint Maritime Combat Information System
JMEM	Joint Munitions Effectiveness Manuals
JODA	JTLS Object Distribution Authority
JOI	JTLS Operational Interface
JPL	Jet Propulsion Laboratory
JRSG	Joint Rapid Scenario Generation (formerly JIDPS: Joint Integrated Database Preparation System)
JSDF	Japan Self-Defense Forces
JTLS	Joint Theater Level Simulation
JTLS-GO	Joint Theater Level Simulation - Global Operations
JTOI	JTLS Transaction Operational Interface
JXSR	JTLS XML Serial Repository
KIA	Killed In Action
KM	Kilometer
KNOTS	Nautical miles per hour
LA	Lethal Area
LAN	Local Area Network
LAT	Latitude
LB	Login Build (JTLS order type)
LDAP	Lightweight Directory Access Protocol
LDT	Lanchester Coefficient Development Tool
LOG	Logistics
LOGIN	Logistics Input
LOGREP	Logistics Report
LONG	Longitude
LOTS	Logistics Over The Shore

LR	Long Range
M&S	Modeling and Simulation
MAPP	Modern Aids to Planning Program
MB	Megabyte
MCP	Mobility Counter-mobility Prototype
MCR	Model Change Request
MG	Machine Gun
MHE	Material Handling Equipment
MIP	Model Interface Program
MOGAS	Motor Gasoline
MOPP	Mission-Oriented Protective Posture
MOSAIC	NCSA user interface software
MOTIF	X Window System graphical interface
MP	Maneuver Prototype
MPP	Message Processor Program
MSC	Major Subordinate Command
MSG	Message
MTF	Message Text Formats
MUREP	Munitions Report
MUSE	Multiple Unified Simulation Environment
NCSA	National Center for Supercomputing Applications (University of Illinois)
NEO	Noncombatant Evacuation Operations
NFS	Network File Server
NGO	Non-Governmental Organization
NIS	Network Information Service or Network Information System
NM	Nautical Mile
NTSC	Naval Telecommunications System Center
OAS	Offensive Air Support
OBS	Order of Battle Service (formerly UGU: Unit Generation Utility)
OCA	Offensive Counter-Air
OJCS	Organization of the Joint Chiefs of Staff
OMA	Order Management Authority
ONC	Operational Navigation Chart
OPM	Online Player Manual

OPP	Order Preprocessing Program
OTH	Over The Horizon
OTH Gold	Over The Horizon message specification
OTH-T	Over The Horizon-Targeting
pD	Probability of Detection
pE	Probability of Engage
pH	Probability of Hit
pK	Probability of Kill
PKL	Point Kill Lethality
POL	Petroleum, Oil, and Lubricants
POSIX	International operating system standard based on System V and BSD
PPS	Postprocessor System
PSYOPS	Psychological Operations
RAM	Random Access Memory
RDMS	Relational Database Management System
RECCE	Reconnaissance (air missions)
RECON	Reconnaissance (ground missions)
REGT	Regiment
RNS	Random Number Seed
ROE	Rules Of Engagement
RPT	Report
RSP	Reformat Spreadsheet Program
SAL	Surface-to-Air Lethality
SAM	Surface-to-Air Missile
SAM/AAA	Surface-to-Air Missile/Anti-Aircraft Artillery
SC	Supply Category
SCP	Simulation Control Plan
SEAD	Suppression of Enemy Air Defense
SIMSCRIPT	Simulation programming language (product of CACI, Inc.)
SIP	Scenario Initialization Program
SITREP	Situation Report
SLP	Sustainment Log Prototype
SOF	Special Operations Forces
SP	Survivability Prototype

SQL	Structured Query Language
SR	Short Range
SRP	Start/Restart Program (a JTLS-GO component)
SRTE	Sea Route
SSM	Surface-to-Surface Missile
STR	Software Trouble Report
SUP	Ship Unit Prototype
SVP	Scenario Verification Program
SYNAPSE	Synchronized Authentication and Preferences Service
TADIL	Tactical Digital Interface Link
TCP/IP	Transmission Control Protocol/Internet Protocol
TEL	Transporter Erector Launcher
TG	Target entity attribute prefix
TGS	Terrain Generation Service (formerly TPS:Terrain Preparation System)
TGT	Target
TMU	Terrain Modification Utility
TOE	Table of Organization and Equipment
TOT	Time Over Target
TOW	Tube-launched Optically-tracked Wire-guided missile
TPFDD	Time-Phased Force Deployment Data
TTG	Target Type Group
TTL	Target Types List
TUP	Tactical Unit Prototype
TW	Targetable Weapon
UBL	Unit Basic Load
UIM/X	GUI builder tool
UNIX	POSIX-compliant operating system
UNK	Unknown
UOM	Unit Of Measure
USA	United States Army (U.S. and U.S.A. refer to United States and United States of America)
USAF	United States Air Force
USCG	United States Coast Guard
USMC	United States Marine Corps
USMTF	United States Message Text Format

USN	United States Navy
UT	Unit entity attribute prefix
UTM	Universal Transverse Mercator
VIFRED	Visual Forms Editor
VMS	Virtual Memory System
VTOL	Vertical Take-Off and Landing aircraft
WAN	Wide Area Network
WDRAW	Withdraw
WEJ	Web Enabled JTLS
WHIP	Web Hosted Interface Program
WIA	Wounded In Action
WPC	Warrior Preparation Center
WPN	Weapon
WT	Weight
XML	Extensible Markup Language
XMS	XML Message Service

APPENDIX B Version 5.1.9.0 DATABASE CHANGES

No database structure changes were made for JTLS-GO 5.1.9.0.

APPENDIX C Version 5.1.9.0 REPOSITORY CHANGES

No changes were made to the JTLS-GO 5.1 repository.