NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

TECHNICAL REPORT

XML Schema-based Binary Compression (XSBC) and Forward Error Correction (FEC) Functionality for AUV Workbench (AUVW) Mission File Archiving

by

Terry D. Norbraten

23 May 2005

Technical Report NPS-MV-2005-00X

Approved for public release, distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** May 2005	**3. REPORT TYPE AND DATES COVERED** Technical Paper	
4. TITLE AND SUBTITLE: XML Schema-based Binary Compression (XSBC) and Forward Error Correction (FEC) Functionality for AUV Workbench (AUVW) Mission File Archiving		**5. FUNDING NUMBERS**	
6. AUTHOR: Terry D. Norbraten			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) MOVES Institute, Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER** NPS-MV-2005-00X	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

This paper will detail instructions for operation of the XSBC and FEC Server panels within the AUV Workbench and will also show the concept and design for XSBC and FEC operation within the source code and configuration files used to configure the AUV Workbench for operational and / or educational use.

XSBC gives the user / designer the ability to tokenize XML documents to reduce file size and FEC provides the data-healing capability for transmitted document packet erasure instances. Channel capacity optimizations are realized with a combination of these two technologies.

14. SUBJECT TERMS: XML Schema-based Binary Compression (XSBC), Forward Error Correction (FEC), Extensible Modeling and Simulation Framework (XMSF), Modeling, Virtual Environments and Simulation (MOVES)	**15. NUMBER OF PAGES** 35
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This paper will detail instructions for operation of the XSBC and FEC Server panels within the AUV Workbench and will also show the concept and design for XSBC and FEC operation within the source code and configuration files used to configure the AUV Workbench for operational and / or educational use.

XSBC gives the user / designer the ability to tokenize XML documents to reduce file size and FEC provides the data-healing capability for transmitted document packet erasure instances. Channel capacity optimizations are realized with a combination of these two technologies.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Associate professor Donald P. Brutzman, PhD., of the Naval Postgraduate School's Modeling, Virtual Environments and Simulation (MOVES) Institute is a major proponent of open source, open standards technology for use within DOD Information Technology (IT) programs. His inner motivation and personal insights for use of open source and open standards has spawned many creative student ideas that add every quarter to the furtherance of Extensible Modeling and Simulation Framework (XMSF) efforts. Dr. Brutzman has opened my eyes to these efforts and has helped me realize that the end user – our Soldiers, Sailors, Airmen and Marines, have research scientists and technical experts working diligently to produce and further technologies that save DOD enormous amounts of money and attempt, with passion, to enable that war-fighter to train and fight with better equipment that leverages the best of our technological edge.

My thanks goes out to Professor Brutzman for taking the time to show me a few things about the Modeling and Simulation world; to Alan Hudson of Yumetech, Inc., for his time and knowledge given to me of XSBC, to the late Dr. Richard Hamming, PhD., for his NPS sponsored course Hamming on Hamming: Learning to Learn and for his discovery of FEC; to CDR Duane Davis for his mentoring and direction concerning the AUVW and to my daughters Amanda, Melinda and Brittany for their unending support for me.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. XSBC

XML Schema-based Binary Compression (XSBC) has been developed as a general approach to binary serialization of XML documents. Elements and attributes are replaced via a tokenization scheme, which carefully preserves valid XML document structure [Husdon 2004]. XSBC uses XML schema as the basis for determining key document parameters such as legal elements, attributes and data types. Binary serialization of XML via XSBC appears suitable for both message streams and document-storage streams. Type-specific algorithms can compress attribute and leaf-node data. Ongoing work includes possible further annotation of schema to include binary compression parameters such as significant digits, significant bits (or bytes), skip-ability, lossy / lossless, etc.

Refinements to the XSBC algorithm and open-source implementation are ongoing. XSBC can further incorporate geometric-compression algorithms, and is being used to implement the forthcoming Compressed Binary Encoding [See Compressed Binary Encoding] for the ISO-approved Extensible 3D (X3D) Graphics. Further information on XSBC is available online at the Extensible Modeling and Simulation Framework (XMSF) project webpage [See Extensible Modeling and Simulation Framework].

B. FEC

Forward Error Correction (FEC) has been around since the mid 1940's. It was invented by the late Richard W. Hamming, PhD., while he worked at Bell Telephone Laboratories in New Jersey. As a mathematician, he was never satisfied with computers that could tell you there were errors in a particular encoding, but would not tell you where that error was and would not do anything to fix it [Hamming 1997]. Since the time of his invention of the first Binary Digit (BIT) error correction algorithms, many other adaptations and variations of FEC have taken root. From the music we listen to encoded onto Compact Discs (CD)'s to deep space probes communicating data back to earth with very low

power transmission capabilities; these data transmitters utilize some type of FEC to correct BIT errors (Reed-Solomon codes) [See Reed-Solomon codes] and / or packet erasure instances (dropped packet data reconstruction / data healing algorithms) [Rizzo 1997].

Further information on the Java based FEC implementation used in this project is available online at Onion Networks, Inc. Developers Java FEC Library 1.0.3 [See Onion Networks].

C. PAPER LAYOUT

Chapter I is concerned with FEC Server panel operations detailing how to operate the panel, what to expect and details the process of the source code methods written to perform FEC operations.

Chapter II details the same information given as above concerning XSBC operations. The XSBC utility feature was first implemented within the AUV Workbench to facilitate efficient mission results file archiving. The FEC utility feature was designed using the existing XSBC feature as a model.

Appendix A gives a list of all abbreviations and acronyms used throughout this paper.

II. FEC SERVER PANEL OPERATION

A. OPERATION OF THE FEC SERVER PANEL

Operation is as follows:

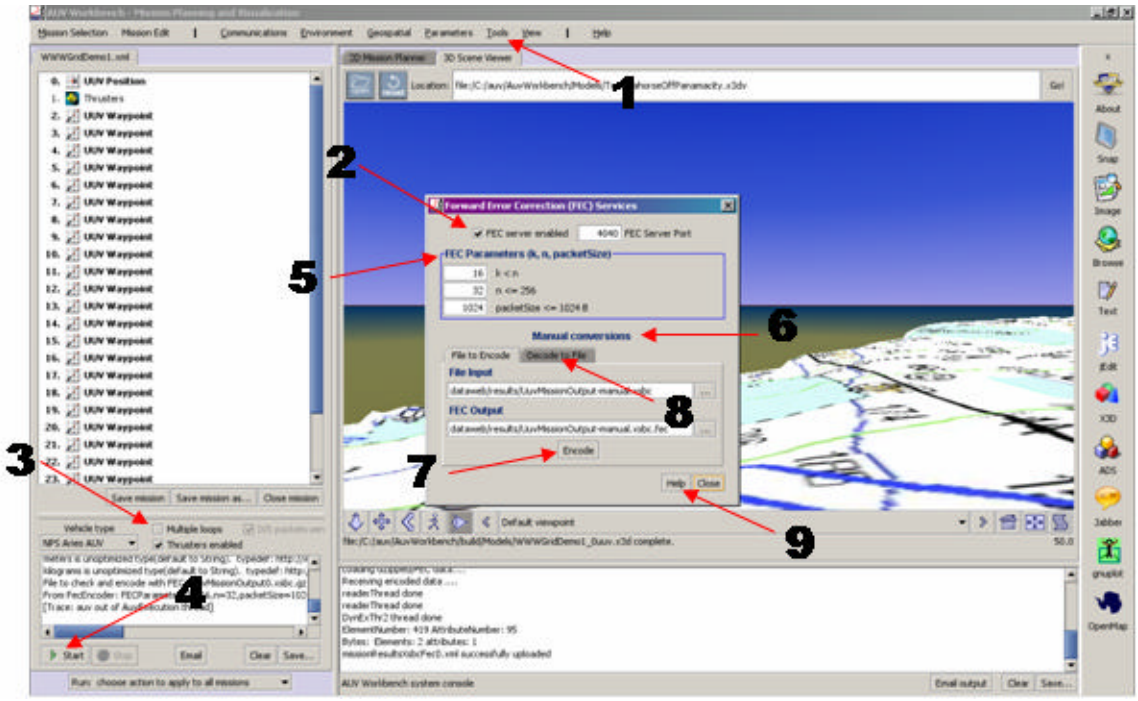

Figure 1. AUV Workbench Screenshot Showing FEC Server Panel

1) Once the AUVW is up and running, select the Tools menu and select FEC services. A separate Graphical User Interface (GUI) panel will become visible.

2) To enable mission results file archiving, select the FEC server enable checkbox. This will start a separate decoding server thread that will allow receiving of encoded repair packets via User Datagram Protocol (UDP). A built-in dropped packet, or packet erasure simulation will be invoked exercising the FEC decoder's ability to repair the erasures and reconstruct the entire transmitted data file. Close the dialog.

3) Deselect Multiple loops (an option to facilitate quick demonstration of FEC operations only).

4) Select Start to run the default mission profile of the Predator Unmanned Arial Vehicle (UAV). You can run the entire mission until “quit,” or, you can select stop to watch the process: XML -> XSBC (serialize) -> GZip -> FEC (encode) -> UDP Transmit -> FEC (decode) -> Un-GZip -> XSBC (de-serialize) -> XML -> save to disc in both lower console windows.

5) The default FEC Parameters will work very well with very large files (30 MB+). The UDP listening port is also set by default to 4040.

6) Manual conversions (encoding / decoding) can be completed by selecting an XML or other file from the file chooser (../dataweb/results) path within the Manual conversions panel. Select or manually name a file with an *.fec extension to encode to.

7) Press Encode once a file is selected.

8) To decode that file, select the Decode to File tab, select the *.fec file and select or name a test file to decode to. Press Decode once a file is named.

9) Select the help button for more information concerning this panel from the JavaHelp menu.

This panel can be used in conjunction with the XSBC server panel to manually serialize / encode -> decode / de-serialize *.xml files generated by the AUVW.

B. FEC LIBRARY REQUIRED FILES

To run FEC routines within the AUV Workbench the following JAR files were placed in the lib/ path during Current Versioning System (CVS) upload to the xmsf project at Sourceforge.net:

- concurrent-jaxed.jar
- log4j-1.2.9.jar
- onion-common.jar
- onion-fec.jar

These files can be obtained in binary form from the Onion Networks, Inc. site [See Onion Networks] or by manually building the source code also obtained from that site.

C. FILES MODIFIED OR CREATED TO FACILITATE FEC OPERATIONS

The following files either were created from scratch, or modified (using the XSBC process as a model) in the existing source code base to enable FEC operations:

- execution/Execution.java
- execution/RuntimeFlags.java
- execution/auv/AuvExecution.java
- execution/auv/AuvFlags.java
- execution/fec/FecEncoder.java
- execution/mission/Mission.java
- execution/uav/UavExecution.java
- execution/uav/UavFlags.java
- worbench/main/AMVWmenuBar2.java
- workbench/main/FECservicesDialog.java

- workbench/main/MissionExecAndDynamicsHandler.java
- workbench/main/MultiMissionController.java
- workbench/xmlutilities/FecDecoder.java
- workbench/xmlutilities/FecServer.java

D. PROCESS OF FEC OPERATIONS WITHIN THE AUVW

The main schema to configure the AUV Workbench is the ConfigurationSchema.xsd located in the configuration/ path. This schema defines the element <xsd:complexType name="FecTransferOptionsType"> that is parsed during the build process by the Java Architecture for XML Binding (JAXB) Binding Compiler (XJC) target to create various java files that are placed in the configuration/jaxb source package path. The JAXB generated class files define parameters that facilitate FEC operation within the AUV Workbench.

```
<!-- Implement Forward Error Correction (FEC) 13 APR 05 (tdn) -->
<xsd:complexType name="FecTransferOptionsType">
    <xsd:annotation>
        <xsd:documentation>Element used to specify options for the UDP transfer of
compressed/serialized mission results files encoded with Forward Error Correction (FEC) back to
the workbench.</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="transfer" type="xsd:boolean" use="required"/>
    <xsd:attribute name="host" type="xsd:string" use="required"/>
    <xsd:attribute name="port" type="xsd:positiveInteger" use="required"/>
    <xsd:attribute name="schema" type="xsd:string" use="required"/>
    <xsd:attribute name="feck" type="xsd:positiveInteger" use="required"/>
    <xsd:attribute name="fecn" type="xsd:positiveInteger" use="required"/>
    <xsd:attribute name="fecpacketsize" type="xsd:positiveInteger" use="required"/>
    <xsd:attributeGroup ref="commonAttributes"/>
</xsd:complexType>
```

Figure 2. FecTransferOptionsType Element Defined in ConfigurationSchema.xsd Parsed by the XJC Target During the AUV Workbench Build Process

The AriesControlConfiguration.xml and PredatorControlConfiguration.xml vehicle configuration files, also located in the configuration/controlCoefficients path, define FEC operational parameters and are parsed during a mission start to begin a selected vehicle's execution process that refer to those FEC parameters.

```
<!-- Modified for Forward Error Correction (FEC) functionality 19 APR 05 -->
<FEC transfer="true" host="localhost" port="4040" schema="Scripts/AVCL.xsd" feck="16"
        fecn="32" fecpacketsize="1024"/>
```

Figure 3. Predator FEC Configuration Element and Parameter Attributes Defined in the PredatorControlConfiguation.xml Vehicle Configuration File

1. Chain of events for the FECTransferOptionsType

By default, the above figure states that the transfer option is set to true. This causes the vehicle runtime flag "FECTRANSFER" to be true. Once a vehicle has completed a mission, or is stopped by the AUVW operator, an FEC mission archive transfer operation initiates. The FecEncoder is passed these FEC parameters and begins an XSBC / GZip process to compress the mission results XML file to its smallest possible file size. A reference to the *.xsbc.gz file path is maintained by the FecEncoder and the encoding process begins. A Message-Digest Algorithm 5 (MD5) is invoked to calculate file hash values. These file hashes are transmitted to the FecDecoder for later file verification required by the FECFile decoding process. Next, the size of the *.xsbc.gz file is transmitted to the FecDecoder. This is the last FECParameter required by the FecDecoder that was not given by each vehicle configuration file as each mission results file size is unique and can not be known until generated at the completion of each mission.

The next process is for the FecEncoder to break up the *.xsbc.gz into individual encoding blocks of which the total number is calculated by:

blockCount = filesize / (k * packetSize)

After the blockCount has been determined, each block is given an *n* vector of indices for Galois Field (*GF*) 2^8 encoding within an (*n * k)* dimension Vandermonde matrix. (*GF*) 2^8 is automatically invoked by the default *n* parameter hard coded in each vehicle's configuration file. (*GF*) 2^{16} is also available by defining an *n* parameter > 256. However, this will result in a significant overhead in order of operations as the required matrices will be too large to be contained in resident memory. For a (*GF*) 2^8 operation occupied memory for the required logarithmic table sizes are only $2^8 * 2^8 * 8$ bits = 64k of memory [Chapweske 2000]. For each block, its *n* vector of indices is transmitted to the FecDecoder.

The final encoding process step is that each block is encoded per the 1…*n* indice in which each of the *n* indices index an encoded repair packet of size packetSize = 1024 B. The 1024 B size vector conforms to most Maximum Transmission Units (MTU)'s of < 1500 B is size. Once each block's *n* vector indices and encoded repair packets have been transmitted, the FecEncoder's operation are complete.

If the FecServer has not been enabled as instructed above, these packets will be lost and the console will output that connection to the FecServer could not be established. If the FEC Server is enabled, then the FecDecoder will capture (receive) the file's hash values, fileSize, block indice values and repair packets. The decoder then generates an FECFile to decode (write) to. The reverse process of encoding takes place by placing each repair packet in its block indice within the decoding matrix, decoding each block and writing the file back to an *.xsbc.gz extension, un-gzipping the file, de-serializing by XSBC de-serialization process, verifying the MD5 hash values and then finally archiving the mission results file to local disc.

Receiving any *k* subset of the *n* repair packets transmitted is sufficient to reproduce the original file and this exact process is simulated by receiving all *n* repair packets, along with their corresponding indices, and selecting a random *k* subset of those repair packets along with their matching

indice. This code is considered systematic in that receiving *k* indice values < the maximum *k* value will represent repair packets in their original un-encoded form which saves cycles in the decoding process. Further information of what a systematic code is can be investigated from [Norbraten 2004].

2. Manual conversions

The process of using the FEC Server panel's Manual conversions tab is essentially the same except that no packets are being transmitted. An FECFile is saved to local disc that contains the encoding data of a file that was manually selected to encode. The decoding process then streams in the encoded *-manual.fec file and writes back to the same directory a *-manual.fec.* file.

The FEC Server does not need to be enabled to perform manual encodings of selected files.

E. CONCLUSIONS

The incorporation of an FEC utility process within the AUV Workbench gives the designer / researcher the ability to ensure redundancy in the transmitted file data so that in the event of packet erasure instances, the received packets will contain enough information to enable reconstruction back to the original file. Even very large file sizes (20 MB+) have no problem being transmitted as each file is broken up into blocks each containing repair packets of size 1024 B which satisfy most MTU restrictions.

Machine performance issues using the Java based FEC 1.0.3 library are explained in more detail in [Chapweske 2000].

THIS PAGE INTENTIONALLY LEFT BLANK

III. XSBC SERVER PANEL OPERATION

A. OPERATION OF THE XSBC SERVER PANEL

Operation is as follows:

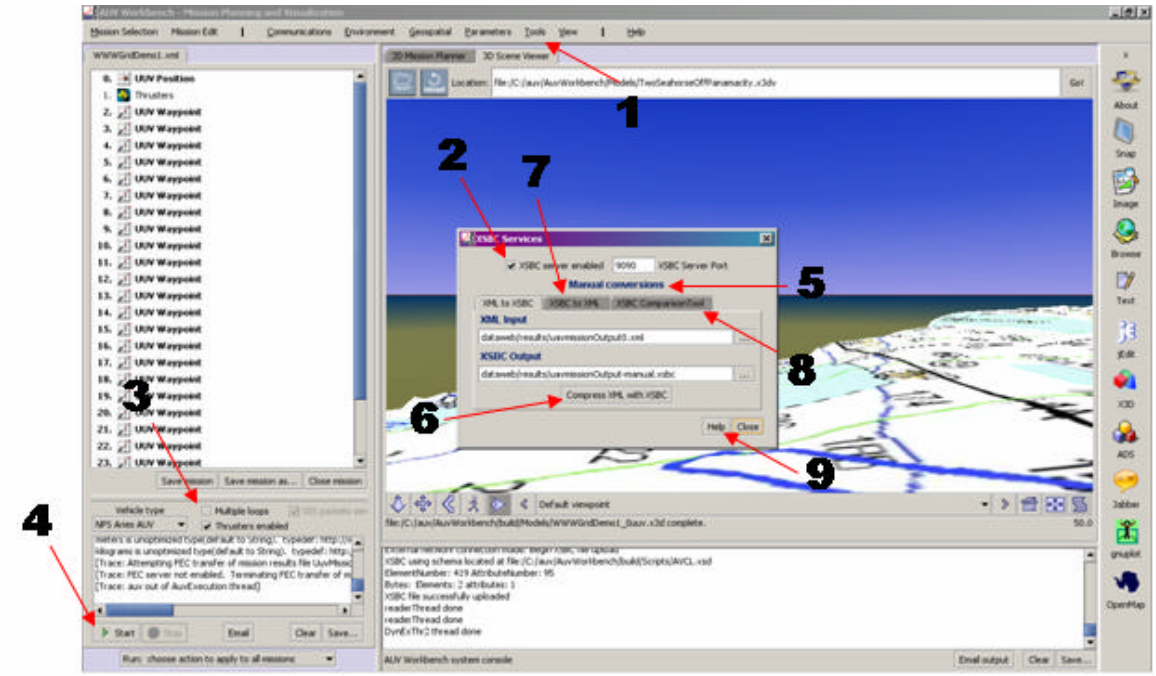

Figure 4. AUV Workbench Screenshot Showing XSBC Server Panel

1) Once the AUVW is up and running, select the Tools menu and select XSBC services. A separate GUI panel will become visible.

2) To enable mission results file archiving, select the XSBC server enabled checkbox. This will start a separate de-serializing server thread that will allow receiving of streamed packets via Transmission Control Protocol (TCP). Close the dialog. The default TCP listening port is set to 9090 by default.

3) Deselect Multiple loops (an option to facilitate quick demonstration of XSBC operations only).

4) Select Start to run the default mission profile of the NPS Aries AUV. You can run the entire mission until “quit,” or, you can select stop to watch the

process: XML -> XSBC (serialize) -> GZip -> TCP Transmit -> Un-GZip -> XSBC (de-serialize) -> XML -> save to disc in both lower system output console windows.

5) Manual conversions (serializing / de-serializing) can be completed by selecting an XML file from the file chooser (../dataweb/results) path within the Manual conversions panel. Select or manually name a file with an *.xsbc extension to serialize.

6) Press Compress XML with XSBC once a file is selected.

7) To de-serialize that file, select the XSBC to XML tab, select the *.xsbc file and select or name a test file to de-serialize to. Press Generate XML from XSBC once a file is named.

8) Select this tab to launch the XSBC ComparisonTool.

9) Select the help button for more information concerning this panel and the XSBC Comparison Tool panel from the JavaHelp menu.

This panel can be used in conjunction with the FEC server panel to manually serialize / encode -> decode / de-serialize *MissionOutput*.xml files generated by the AUVW.

B. OPERATION OF THE XSBC COMPARISON TOOL

Operation is as follows:

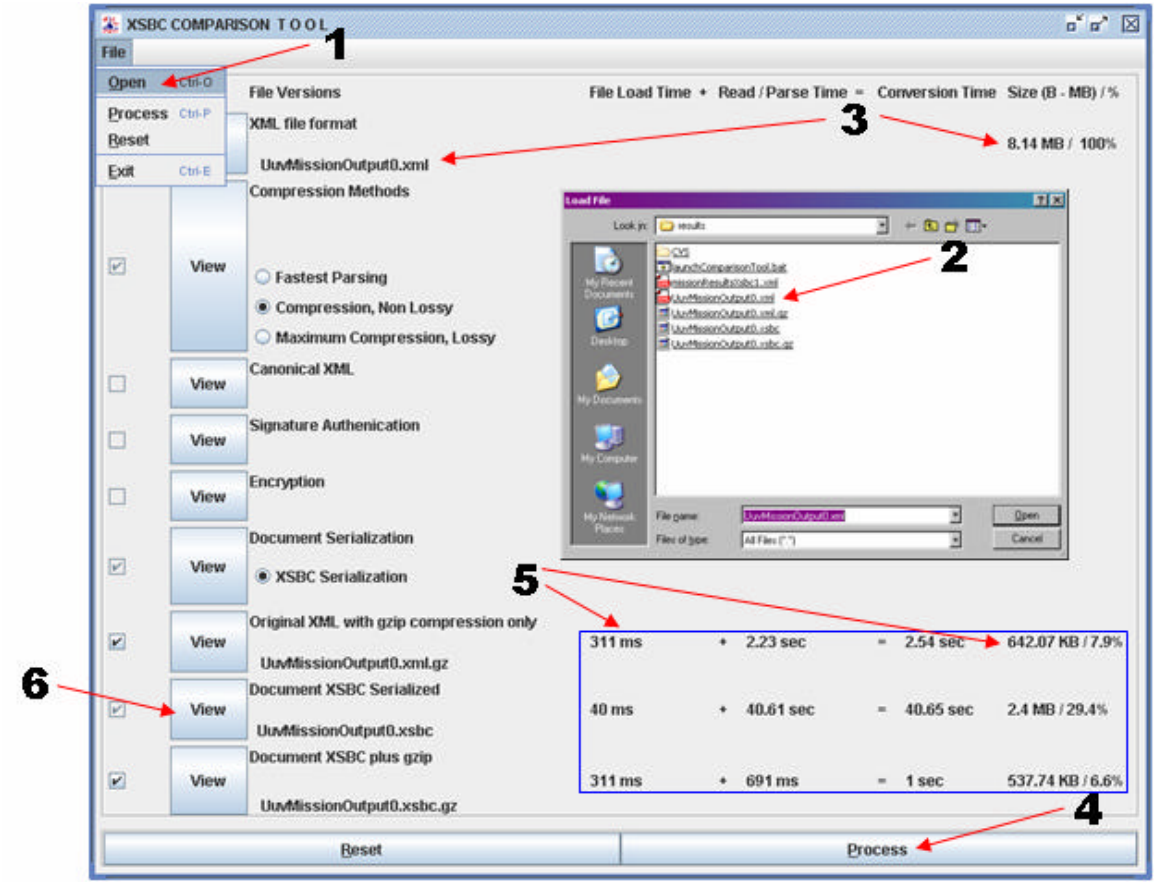

Figure 5. Snapshot of the XSBC Comparison Tool

1) Select File / Open to display the JFileChooser.

2) The file choose will open up to the dataweb / results directory by default. The vehicle type mission file must be selected as it contains a reference to the AVCL.xsd which is required by XSBC to perform serializations.

3) The Comparison Tool will load the file, read the location of the schema file defining the structure of this file, Scripts/AVCL.xsd in this case, and display the name of the file along with its original file size.

4) Select Process to begin the serialization process.

5) Observer the process outcome statistics in this area. File loading, read / parse and combined processing times will be displayed as well as the corresponding compressed file sizes for comparison. Note: A build of GZIP must be installed on your system to view GZIP compression results.

6) Select View to call the de-serialized file to load into XMLSpy for viewing. The process goes through a quick demonstration of FEC encoding / decoding before loading into XMLSpy. The resulting file is renamed as foo.xml which is placed in the dataweb/results directory.

C. XSBC LIBRARY REQUIRED FILES

To run XSBC routines within the AUV Workbench the following JAR files were placed in the lib/ path during CVS upload to the xmsf project at Sourceforge.net:

- batlik-util.jar
- dom4j-full.jar
- xercesImpl.jar
- xsbc.jar

These files can be obtained in binary form from the xmsf project site [See XMSF project] or by manually building the source code from the xsbc directory at this site.

Since the XSBC application and example files call certain FEC routines as well as the embedded FedTestFiles that come embedded with the current XSBC build, the required jar files for FEC mentioned above will also be needed to properly build XSBC.

D. FILES CREATED TO FACILITATE XSBC OPERATIONS

The following files were created from scratch to enable XSBC operations:

- execution/Execution.java
- execution/RuntimeFlags.java
- execution/auv/AuvExecution.java
- execution/auv/AuvFlags.java
- execution/xsbc/XsbcSerializer.java
- execution/mission/Mission.java
- execution/uav/UavExecution.java
- execution/uav/UavFlags.java
- worbench/main/AMVWmenuBar2.java
- workbench/main/MissionExecAndDynamicsHandler.java
- workbench/main/MultiMissionController.java
- workbench/main/XSBCservicesDialog.java
- workbench/xmlutilities/XsbcTransaction.java
- workbench/xmlutilities/XsbcServer.java

There are a host of additional files created by the XJC process during the AUW Workbench build process. These files are contained in the configuration/jaxb path and are subject to be cleared and re-built with each AUV Workbench build process initiated.

E. PROCESS OF XSBC OPERATIONS WITHIN THE AUVW

The main schema to configure the AUV Workbench is the ConfigurationSchema.xsd located in the configuration/ path. This schema defines the element <xsd:complexType name="XsbcTransferOptionsType"> that is parsed during the build process by the XJC target to create various java files that are placed in the configuration/jaxb source package path. The JAXB generated class files define parameters that facilitate XSBC operation within the AUV Workbench.

```
<xsd:complexType name="XsbcTransferOptionsType">
    <xsd:annotation>
    <xsd:documentation>Element used to specify options for the transfer of compressed mission results back to the workbench.</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="transfer" type="xsd:boolean" use="required"/>
    <xsd:attribute name="host" type="xsd:string" use="required"/>
    <xsd:attribute name="port" type="xsd:positiveInteger" use="required"/>
    <xsd:attribute name="schema" type="xsd:string" use="required"/>
    <xsd:attributeGroup ref="commonAttributes"/>
</xsd:complexType>
```

Figure 6. XsbcTransferOptionsType Element Defined in ConfigurationSchema.xsd Parsed by the XJC Target During the AUV Workbench Build Process

The AriesControlConfiguration.xml and PredatorControlConfiguration.xml vehicle configuration files, also located in the configuration/controlCoefficients path, define XSBC operational parameters and are parsed during a mission start to begin a selected vehicle's execution process that refer to those XSBC parameters.

```
<XSBC transfer="true" host="localhost" port="9090" schema="Scripts/AVCL.xsd"/>
```

Figure 7. Predator XSBC Configuration Element and Parameter Attributes Defined in the PredatorControlConfiguation.xml Vehicle Configuration File

1. Chain of events for the XSBCTransferOptionsType

By default, the above figure states that the transfer option is set to true. This causes the vehicle runtime flag “XSBCTRANSFER” to be true. Once a vehicle has completed a mission, or is stopped by the AUVW operator, an XSBC mission archive transfer operation initiates. The XsbcSerializer is passed these XSBC parameters and begins an XSBC / GZip process to compress the mission results XML file to its smallest possible file size.

If the XsbcServer has not been enabled as instructed above, the transfer process will halt and the console will output that connection to the XsbcServer could not be established. If the XSBC Server is enabled, then the XsbcTransaction will receive, via TCP, the serialized file to un-gzip back to an *.xsbc extension, then de-serialize by XSBC de-serialization process before archiving the mission results file to local disc.

The serialization and de-serialization processes use the AVCL.xsd schema in the Script/ path to build XML tree representations in memory. Further information on these particular processes can be investigated from [Serin 2003].

2. Manual conversions

The process of using the XSBC Server panel’s Manual conversions tab is essentially the same except that no files are being transmitted. A mission results file saved to local disc is manually selected to serialize. The de-serialization process then de-serializes the file and writes back to the same directory a *-manual.xsbc.* file.

The XSBC Server does not need to be enabled to perform manual serializations of selected files.

F. CONCLUSIONS

The incorporation of an XSBC utility process within the AUV Workbench gives the designer / researcher the ability to ensure channel capacity issues are dealt with efficiently by utilizing the dual compression algorithms available in the XSBC utility. While it is noted that parsing speeds are not increased by the XSBC process, XSBC does add to channel capacity efficiencies by significantly reducing the verboseness of terse XML documents.

Current streaming limitations have been identified in XML files > 20MB in that not enough Java Virtual Machine (JVM) heap size can be allocated for the enormous table sizes needed to reconstruct / de-serialize those size files. JVM heap size allocation, of course, is an independent machine capability issue.

APPENDIX A. LIST OF ABBREVIATIONS

AUV	Autonomous Un-manned Vehicle
AUVW	AUV Workbench
B	Byte
BIT	Binary Digit
CD	Compact Disc
CVS	Current Versioning System
FEC	Forward Error Correction
JAXB	Java Architecture for XML Binding
GL	Galois (pronounced “gal wah”) Field
GNU	Recursive Acronym for “GNU's Not UNIX”
GUI	Graphical User Interface
GZIP	GNU Zip
ISO	International Standards Organization
IT	Information Technology
JVM	Java Virtual Machine
M&S	Modeling and Simulation
MD5	Message-Digest Algorithm 5
MOVES	Modeling, Virtual Environments and Simulation
MTU	Maximum Transmission Unit
NPS	Naval Postgraduate School
PhD	Doctor of Philosophy
SAVAGE	Scenario Authoring and Visualization for Advanced Graphical Environments

TCP	Transmission Control Protocol
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
UUV	Un-manned Underwater Vehicle
USW	Undersea Warfare
X3D	Extensible 3D
XJC	JAXB Binding Compiler
XML	Extensible Markup Language
XMSF	Extensible Modeling and Simulation Framework
XSBC	XML Schema-based Binary Compression

LIST OF REFERENCES

Chapweske, J.F., (2000), "Forward Error Correction Performance", paper contained in the docs folder of the binary fec-1.0.3.zip file in .pdf format titled "FECPerformance".

Compressed Binary Encoding. Retrieved April 2005 from http://www.web3d.org/x3d/specifications/ISO-IEC-19776-3-CD-X3DEncodings-CompressedBinary/index.html

Extensible 3D (X3D) Graphics. Retrieved April 2005 from: http://www.web3d.org/x3d

Extensible Modeling and Simulation Framework (XMSF). Retrieved April 2005 from: http://www.movesinstitute.org/xmsf/xmsf.html#Projects-XSBC

Hamming, R.W., *The Art of Doing SCIENCE and Engineering: Learning to Learn*. Amsterdam B.V., The Netherlands: Gordon and Breach Science Publishers, 1997.

Hudson, A. XSBC concept excerpt from the Sourceforge CVS site for XSBC source code retrieved April 2005 from http://cvs.sourceforge.net/viewcvs.py/xmsf/xsbc/docs/xsbc.html?rev=1.2&view=markup

Norbraten, T.D., "Utilization Of Forward Error Correction (FEC) Techniques With Extensible Markup Language (XML) Schema-Based Binary Compression (XSBC) Technology", Master's Thesis, Naval Postgraduate School, Monterey, CA, December 2004. Last accessed April 2005 from: http://theses.nps.navy.mil/04Dec_Norbraten.pdf

Onion Networks, Inc. Retrieved April 2005 from http://www.onionnetworks.com/about.php

Reed-Solomon Codes, (n.d.), "An introduction to Reed-Solomon codes: principles, architecture and implementation", Last accessed April 2005 from http://www.4i2i.com/reed_solomon_codes.htm

Rizzo, L., "Effective erasure codes for reliable computer communication protocols", ACM Computer Communication Review, Vol. 27, n. 2, April 1997. Also available as DEIT Technical report LR-970115, retrieved April 2005 from http://www.iet.unipi.it/~luigi/fec.html

Serin, E., “Design and Test of the Cross-Format Schema Protocol (XSFP) for Network Virtual Environments”, Master’s Thesis, Naval Postgraduate School, Monterey, CA, March 2003. Retrieved December 2004 from http://www.movesinstitute.org/Theses/Serinthesis.pdf

INITIAL DISTRIBUTION LIST

1. Dr. Donald P. Brutzman
   Naval Postgraduate School
   Monterey, CA

2. LT Terry D. Norbraten, USN(Ret)
   U.S. Navy
   Aptos, CA

3. CDR Duane T. Davis, USN
   Naval Postgraduate School
   Monterey, CA

4. Margaret Bailey
   Sonalysts
   Waterford, CT

5. Alan Hudson
   Yumetech, Inc.
   Seattle, WA