Information Paper

19 Aug 2019

Subj: MILSTD XML Implementation Recommendations

1. Background. XML Schema defined data models provide control measures that can be used to test, verify and validate software implementations. This paper defines and describes these capabilities for XML Schema defined Military Standard (MILSTD) information exchanges using the National Information Exchange Model (NIEM) XML Schema naming, design, and namespace management rules.

a. Purpose. The purpose of this guidance is to provide best practices that will ensure MILSTD information exchanges remain provably conformant with MILSTD Specifications, while leveraging secure, and efficient delivery strategies.

b. Authoritative Procedures. Military standardized information models are the result of continuous Joint and Combined technical collaboration to ensure Interoperability between systems and operational personnel. NIEM provides uniform requirements for XML Schema design that can be verified using automated processes.

c. Automated Test and Verification. The primary purpose of using XML Schema to define Information Models is to achieve verified, testable data control. The requirements outlined in this document must be accommodated by software automatically, completely, and in a way that can be independently tested, verified and validated.

d. XML Schema Validation. XML Instances must be validated by W3C XML Schema for structure, and with W3C XPath defined for Structural Relationship Rules. Each of these bring implementation, automation, and testing challenges.

2. XML Schema Design. Because XML Schema are defined using XML,rules can be created and used to automatically verify the extent to which an XML Schema is conformant to specified design constraints. From a security perspective alone, this provides valuable control measures. An XML Schema managed namespace defines design principles using XPath rules to support automated verification.

a. Use a Managed XML Namespace. A managed namespaces is one that defines specific XML Schema Naming and Design Rules, and provides a means to verify them. US DoD systems are directed to use the National Information Exchange Model (NIEM) as the basis for all managed namespaces. NIEM naming and design rules can be obtained as machine readable XML documents from the NIEM website for use in automated validation.

b. Use Normative Subset XSD for Individual Messages. A managed namespace best practice is to define all namespace nodes in a single XML Schema document. NIEM defines this level of XML Schema as a "Reference Schema."For MILSTD message formats, this file can be quite large, so it is practical to extract each message for separate implementation. These are known as "Subset Schema," and all represent the same namespace.

c. Use Restriction XSD to Define Delivery Formats. While the XML format provides invaluable validation capabilities, it is often impractical as a delivery format due to the size that text content brings. In order to ensure that a delivery format accurately represents the XML format, it is generally necessary to consume the XML Schema to create equivalent data objects. The various strategies for accomplishing this are discussed in Appendix A.

(1) Restriction Schema. A Restriction Schema is a schema that implements the Reference Schema and retains the same namespace, but It is not intended for for extension or restriction, and may include structures and external dependencies that do not conform to NDR specifications. All instances of a Restriction Schema will be valid against the Reference Schema.

(2) Delivery Strategies. Each of these presentation formats require that data be losslessly transformed back to valid XML on completion. The role of XML Schema is to ensure that all restrictions and extension which are required for this purpose, occur in a controlled and testable manner.

(a) Binary. Programming languages that provide the ability to send code objects in a binary format to a remote process running the same code, can provide a simple means of message delivery that will incorporate all required information in a manner that can be tested and verified.

1. Proto Buffers. This is a modeling language for objects that can be mapped to code objects in a cross-platform manner. Proto-Buf definitions can be generated from XML Schema using XSLT, or using a variety of available methods.

2. Type Length Value (TLV) Encoding. A TLV-encoded message is provides a way by which fields with variable lengths, types, and occurrence can be detected and handled with optimal efficiency. Because length limits and types are specified in XML Schema, this is a transparent and testable approach to the binary optimization of text.

3. MILSTD Binary. Several MILSTD formats define and maintain specified binary delivery strategies designed to increase efficiency, accuracy and testability. The role of XML Schema for these strategies is primarily testing and verification, but it can also be used to translate information to other formats in a controlled and verifiable manner.

(b) Efficient XML. This is a standardized methodology for creating binary content from XML Schema defined XML documents.

(c) Data Format Description Language (DFDL). This is is a modeling language, based on W3C XML Schema, that is used to describe general text and binary data in a standard way.

(d) XML Instance. As the default, authoritative, testable representation, XML is the most practical for information exchanges between separate processes on a computer, but should never be used for network communications.

d. Use Extension XSD to Define Presentation Formats. All MILSTD message formats will define presentation formats in a way that can be validated using XML Schema.

(1) Extension Schema. In many cases it is necessary to add information to a Reference XML Schema. The most common occurrence of XML extension is in the addition of security tags. An instance of extension schema will be validated against the Reference Schema for the types which it employs, and additional XML definitions must be imported or defined as they are with the IC-ISM security markings.

(2) Presentation Formats. A key software development tenet is the separation of presentation and data. Text, digital display, and network delivery are all examples of presentation formats that will not generally be XML, but must adhere to an XML defined data model.

(a) Message Versions. Compatibility between different versions of messages, and different types of messages can be achieved by creating extension schema for all versions that employ the Namespace Simple Types, and only alter Complex Type and Element names as necessary to produce the original XML instance documents. This methodology will place the XML instances from prior versions into the managed namespace for validation purposes.

(b) Message Conversion. Backward compatibility and other translations can also be accomplished using XSLT, or standard programming languages. Results, and the process by which they are created must be verified and validated against a variant of the Reference XML Schema.

(c) Application Programming Interfaces (API). Access to XML defined data resources can be provided by a set of functions and data structures that are derived from, and mapped directly to, the data model. Automated methods employed to generate these data objects include JAXB, Efficient XML, DFDL, and XSLT.

(d) Automated Systems. Automated implementations can be employed to leverage delivery strategies that can leverage de-duplication, synchronization, peer networking, and other complex networking technologies that must, at specific points, result in artifacts that can be validated against normative XML Schema.

(3) Security Markings. Both US DoD and NATO provide XML Schema for security markings that are intended to be included in XML Schema defined data formats using XML extension. This allows validation of the security tags and the containing XML Schema using schema from separate and independent managed namespaces.

(4) Security Implementation. Military communications are inherently constrained by information security requirements, so implementations that do not include security tagging capabilities should never occur in the operational environment. Because XML validation and verification is required to implement both capabilities, it is not recommended to perform these tasks separately, or assume that they will be performed correctly by an independent process.

e. Implementation. Implementation requirements to Validate, Verify and , Log data artifacts against XML Schema are less stringent than those required to test adherence to an information standard. Process intensive tasks can be minimized, and in some cases eliminated during operation if they are included and rigorously tested during development cycles.

(1) Validate All XML Schema. As the basis of all instance validation,each XML Schema must be verified as valid against the W3C XML Schema, as well as against XPath defined structural relationship rules, usually provided by W3C Schematron rule-sets.

(a) W3C XSD Validation. All W3C XML Schema must be validated against the W3C XML Schema for content format and structure.

(b) Structural Rules. XPath defined rule-sets can be embedded in XML Schema, or provided separately. Beginning with XML Schema v 2.0, this capability will be included in W3C XML Schema, which is why the common language, XPath, is used to refer to both Schematron and W3C XSD V2 methods for structural relationship rules.

(2) Verify XML Schema prior to validating XML instances. This is compute process intensive, and XML Schema are re-used, so this should not be repeated once the hashes of each known good XML Schema are saved and can be used to verify each schema as part of any instance validation process.

(3) Maintain Log of valid documents with Hash Digests. In any case where a known valid XML artifact is to be re-used, verification against a hash should be preferred over validation. Logs are also required to verify that software processes are performing required validation and verification procedures as expected.

f. Testing. Instances of Subset Schema must be validated against the Normative Reference Schema.

(1) Verify All XML Schema using Authoritative Hashes. An authoritative implementation of an XML defined MILSTD data models must employ authoritative XML XML Schema that can be verified against known good hashes published by a DoD Agency.

(2) Validate All XML Schema using W3C XML Schema. Full validation of all artifacts is required as part of the development cycle.

(3) Validate Core XML Schema using Namespace Specific NDR XPath Rules. Naming and design rules are used to specify the purpose and context of namespace Schema, and apply specific testable rules for extension and restriction. Departure from these formats creates levels of variance and complexity that cannot be reasonably tested. Verification of adherence to these formats is an important departure point for testing any managed namespace.

(a) Reference Schema. NIEM provides specific Naming and deign rules for Reference Schema which is used to define all objects in a namespace using a singe XML Schema document. All NIEM design rules can be tested using Schematron and XSLT rule-sets provided by NIEM.

(b) Subset Schema. Subset schema follow the same rules as Reference Schema, and perform the same function, but only include data object necessary for a particular implementation. A subset Reference Schema can be provided for each message in a MIL STD.

(c) Restriction and Extension Schema. NIEM provides separate design rules for Restriction and Extension schema in order to retain XML Schema validation with respect to namespace defined data objects.

(4) Ensure that all XML Schema used for Validation are Verified. If an XML Schema document can be changed, and employed without detection, the potential for inadvertent and deliberate compromise is significant.

(5) Ensure Implementation Log Entries Reflect Verification and Validation Events. An operational implementation must produce logs that reflect that verification and validation events occur as required, and that the results can be verified.

3. Automated Processes. All XML Schema Design Implementation and verification strategies can be accomplished using XML Integrated Development Environments. Specific considerations must be accommodated in order to achieve these objectives using automated process as part of software test and development or execution cycles.

a. Verify Reference XML Schema. While the idea of retaining a log of known good hashes is fairly basic, doing it in such a way that will resist deliberate or inadvertent compromise requires more rigor than simply maintaining a list of hashes. This is an appropriate situation to employ an immutable ledger.

b. Verify XSLT Files. XSLT scripts are often used to perform conversions and extract data from XML objects for specific purposes. They are also used to evaluate XPath defined rule sets provided using Schematron, to include those required to evaluate security tags, so XSLT validation is not an optional step. All XSLT files must be valid against the W3C XML Schema that defines the XSLT language and results retained for verification.

c. Verify Test Data. All test and development cycles must employ relevant test data that is obtained and verified from an authoritative source. It is not necessary that this data be valid against an XML Schema, or even provided as XML, but it must be verified as originating from an authoritative source.

d. Generate Reference Schema for Normative Subset XSD. If a Subset XSD is provided for a specific purpose, this step is not necessary. If multiple Subset XSD are to be implemented, a process for extracting a Subset XSD from the Reference XSD is required. XSLT is the most direct method of achieving this.

e. Generate Restriction Schema from Reference XSD. While MILSTD XML implementations should leverage the NIEM architecture, but must retain constraints that do not allow certain NIEM dependencies or conventions. For this reason MILSTD technical bodies are recommended to provide normative XML Schema that retain validation against the managed namespace, but employ implementation specific design rules.

f. Generate Test Instances for Restriction XSD. Given an XML Schema and test data, an implementation must be able to generate a valid instance.

g. Generate Code and Code Tests for XSD Data Objects. There are various methods for processing XML Schema in order to generate code objects that can be used to read, write, validate and use information from XML documents.

h. Validate Test Instances with Reference XSD. The rule for namespace conformance is that all instances must be valid against the Reference Schema, as well as against the Restriction schema. Dual validation should not occur on every event, but it must be included in test cycles and initiation processes.

i. Validate Test Instances with Restriction XSD. This is the primary validation target for XML instances.

j. Marshal XML Instances using auto-generated code. This is required to test auto-generated code objects, as well as to run auto-generated code tests.

k. Execute Auto-Generated Code Tests. This requires the ability to marshal XML data correctly

l. UnMarshal XML Instances using auto-generated code. After being marshaled into code objects, software must be able to demonstrate efficacy by test of outputting the same document as XML for validation.

m. Validate UnMarshalled XML Instances using Reference XSD. This is the same procedure employed for generated instances.

n. Validate UnMarshalled XML Instances using Restriction XSD. Also repeating the process used for generated instances.

o. Log All Events. The preceding operations represent the minimum requirement for providing an implementation of an XML defined standard in a way that can be independently verified. Retaining the fact of these events in an immutable ledger, allows oversight and remote, automated verification capabilities for operational systems.

p. Use all Artifacts to Generate a Software Library for Re-use. The NIEM methodology include recommendations for Information Exchange Package Definitions (IEPD) which are intended to include all artifacts, code and documentation required to implement an information exchange. This will include the XML Schema and code required to perform the described tasks and tests.

4. Implementation

a. Software Development. All steps are automated and the cycle is repeated when any change is detected, or for verification events. All code must be open source, testable, and should minimize external dependencies to the greatest degree possible.

(1) Verify Stored XML. XML validation and transformation are compute intensive operations, so known-valid data artifacts should be tracked using hash digests, and verified. This requires creation of, or access to, a list of hashes in a database.

(2) Validate XML

(a) Format. This level of validation requires the use of an XML parsing library, and the means by which validation errors can be logged and displayed.

(b) Structural Relationships. XPath defined relationship validation requires the use of an XSLT parsing library, and the means by which validation errors can be logged and displayed.

(3) Generate Source Code. Libraries are available to generate code objects for XML Schema defined data objects. This can also be accomplished using XSLT.

(4) Test Source Code. Auto-generated code must include auto-generated tests which employ authoritative test data. Additional code must also include tests. Test coverage for normative data objects in MILSTD implementations must be 100%.

(5) Report All Events. An integrated logging system must be able to produce a machine readable report that contains results of all automated processes and tests for independent verification, comparison, authentication, and update notification.

b. Authoritative Data. As data objects are created, or otherwise obtained,they must be verified, and validated.

(1) Verify Document. Known good valid data artifacts that have been logged on a shared ledger, verification is sufficient. Some delivery strategies will share a hash by other means for this purpose.

(2) Validate New Documents. Validate new or un-logged artifacts, and log them.

(3) Employ. Parse, transform, marshal, or otherwise use the data to support presentation formats for information exchange and data processing.

(4) Test Data. In order to auto-generate instances, or code tests, known-good data sets, with known bad options,must be provided.

(5) Reports. Logs for known good XML artifacts, and results of all verification and validation process must be retained.

c. Security Requirements. Correct handling of security tags is far more challenging than adding them to XML Schema documents, because this requires conditional logic that must be defined, executed and reported.

(1) Security Tags. All military information exchange requirements include the need for security tagging for use by automated systems in controlling access. Information can only be shared to the extent that it can be specifically tagged.

(a) IC-ISM. US DoD uses Intelligence Community XML Schema defined information security markings (IC-ISM), and requires that they be applied where appropriate. This indicates that guards will use XML validation, and drives the requirement to use XML Schema to define and validate tagged content.

(b) NATO STANAGs 4776/4778. The same function is defined by NATO using XML Schema.

(c) Risk Mitigation. If security markings are not explicitly provided, a computer process cannot be used to automate the dissemination of discrete data items from MILSTD messages. Once sent, unmarked messages create security risks, because without authoritative retention and releasability information, and associated logging, spillage can occur without detection.

(2) Slash Delimited Payload. Anyone sending slash delimited messages must either perform the conversion to XML, or rely on some upstream process to do this, or implement an alternative way to verify that the slash delimited content contains expected content.

(3) Dirty Word Search. An alternative approach,is to wrap the slash-delimited message with a security tag matching the highest one in the text, and run a "dirty-word search" on the message before sending across a guard. This of course must be carefully, tested, constrained, verified and then only trusted in a specific context.

(4) Affordable Guards. If XML Schema validation can be used to verify security tags and content, because the content is well enough defined so that it can be trusted - then guards will be easier to verify, less expensive to maintain, and pose less risk of compromise.

d. Information Exchange

(1) Verify. If a a message has been received, and a corresponding entry made to a shared ledger, a hash can be used to verify the data object without transforming it to XML.

(2) Validate. For messages created for delivery, this will have been accomplished and logged during the creation process.

(3) Send. This payload must contain all information necessary to allow guards to verify security information and content,and allow receiving nodes to complete the information exchange life-cycle. All transmission events must be logged.

(4) Receive. All message receipt events must be logged, with appropriate hash information for verification purposes.

(5) Test. Because incorrect information sent to a guard is indistinguishable from an attack, information exchanges must be tested internally before data is sent. This can be accomplished using information about what each node expects in terms of security markings, message version, message restriction schema, and structural relationship rules.

(6) Report. Logs for each of these steps must be accessible for independent verification.

e. Data Life-cycle. To effectively exchange and leverage message information, data must be changed, selectively used, redacted, or minimized.

(1) Presentation Format. The human readable representation of a MILSTD

message must follow specified guidelines and contain appropriate markings. They will vary for screen display and print. These specifications can be provided and verified using XPath rule-sets.

(2) Normative Format. This is the format specified by standards bodies, and required for all XML validation operations. While it must be complete, and immutable, it does not always have to be delivered in its entirety. Some delivery strategies may choose to replicate a recent message, and synchronize changed fields.

(3) Transmission Format. This is the means by which required secure information exchanges occur, using a delivery strategy appropriate for the mission environment. The update strategy represents a potentially extreme redaction, but the included data items can inherit the validity of the source if they remain unchanged. A Transmission Format is valid if the Normative Format can be achieved with information available to the recipient. A transmission format that sends Normative XML information over the network is rarely advisable.

f. Guards. Guards cannot be expected to operate as filters. Rule set evaluation and validation must be performed as part of the XML Validation cycle using the guard rule set before it can be sent, so any implementation must, in effect, also implement a guard.

(1) Rule-Sets. Guard security settings will include handling instructions, releasability, and destruction information as well as rules to verify the exclusion or inclusion of specific information in the message payload. These settings must be published to all subscribers.

(2) Dynamic Rule-Sets. Because IEPD XML and code artifacts can be generated and verified automatically, it is not unreasonable to expect this to occur in accordance with mission requirements,in order to allow mission driven changes to guard settings without service interruption.

(3) Restriction Schema. In many cases, security requirements dictate the removal of all unnecessary information. For example, in a machine-to-machine exchange, Free Text is not needed. The XML automation and verification procedures described can be employed in direct response to operational inputs, with changes applied, published and enforced continuously using auto-generated, verified and validated XML defined rule-sets and auto-generated XML Schema.

5. Conclusions

a. Directive Guidance. The use of uniform XML Schema design rules creates an environment in which tooling can be re-used across all XML Schema defined information products for purposes of test and evaluation. The intent of this paper is to provide understanding that because these procedures can be verified,

b. Common Terminology. The appendices are intended to provide in-depth guidance on the concepts and procedures required to implement a MILSTD message, but until these concepts are included in required test and verification cycles, they remain notional.

Appendix A. Automated XML Processing

a. W3C XML Schema. The core functionality of W3C XML Schema defined format is provided by an XML parser, which evaluates data objects using content and contextual information provided using the XML Schema language.

(1) Tests. XML Schema parser validation asserts that values are within specified minimum and maximum values,lengths and occurrences, and adhere to specified Regular Expressions. Structures retain specified order and are repeated within specified occurrences. This level of validation only reports errors on failure, or “True” on success. There are no other informational or “partially” valid options.

(2) Processing. This level of validation requires that an XML Instance document and and W3C XML Schema be provided to an XML Parser. If the XML document is valid, there is no response beyond the Boolean value. If it is false, errors are produced.

b. W3C Schematron. The core functionality of Schematron or other XPath defined format is provided by an XSLT parser, so at some point all XPath rules found in Schematron must be converted to an XSLT document for processing.

(1) Tests. Anything that can be expressed with XPath logical statements can be tested on an XML document. Schematron rules are used to verify XML Schema Naming and Design rules in XML Schema, which are XML documents. Schematron is used to verify XML defined Security Tags, such as are defined by IC-ISM and NATO STANAG 4774. Information errors, and other content can be returned. Results use the Schematron Validation Result Language (SVRL) which is defined by an XML Schema and can be validated. XPath rules are processed using XSLT must be generated from SVRL.

(2) Processing

(a) Convert Schematron to XSLT. The use of W3C SVRL XML Schema instance as the result document for the XSLT provides another level of validation.

(b) Evaluate XML Document. Pass, Fail, and Informational results must be evaluated for specific purposes. These include testing, and validation for purposes of guard traversal.

(c) Process Report. In order to preserve the "fact of validity" an event log is mapped to the hash of the document, and this document can be verified using a distributed hash table, vice validated. SVRL can be validated, marshaled, and UnMarshalled using the same process that performed the validation, thereby allowing the report to be used as an authoritative artifact for purposes of verification.

(3) W3C XML Schema v2.0. The latest versions of W3C XML Schema support XPath relationship rules organically, which means that both structural and relational validation can be accomplished in a single operation. XML Schema V1 processing libraries such as XMLPROC, and XSLTPROC have been open source and widely used for decades, have minimal dependencies, and accomplish the required tasks effectively. The adoption of XML Schema V2 requires much newer parsers, and introduces additional dependencies. Until XML Schema V2 capabilities are show to provide required advantages, which is not unlikely, the time tested solutions are recommend.

c. W3C Extensible Stylesheet Language for Transformation (XSLT). The basis for all Structural Rules analysis is the XPath language, which is processed using XSLT. Because this is the methodology used to evaluate security tags, and because all secure messages are expected to be restricted and mission specific structural rules defined, XSLT cannot be considered as a threshold, vice target, requirement for MILSTD Message implementation.

Appendix B. Information Exchange Life-Cycle

a. **Create Presentation Format**
  (1) XML Schema information
  (2) Security Information
  (3) Data Interface
    (a) Machine
    (b) User
  (4) Message Data

b. **Create Normative Format**
  (1) Message XML
  (2) Information Security XML
  (3) Combined Payload
  (4) Verify Data
  (5) Validate Data
    (a) W3C XML Schema - Format and Content
    (b) W3C Schematron - Conditional Rules

c. **Create Transmission Format**
  (1) Compress Normative Format
  (2) Marshal Normative Format create to Binary Objects
    (a) Efficient XML
    (b) Proto Buffers
  (3) Initiate Transmission Strategy
    (a) High Throughput
      1. Full Binary Message
      2. Full XML Message
      3. Message Synchronization
    (b) Low Throughput
      1. Logically Separated Objects
      2. Incremental, Ordered Delivery
      3. Optimized Synchronization
    (c) Security
      1. Redact Information
      2. Restrict Content
  (4) Encrypt Message
    (a) IC Trusted Data Object Keys
    (b) Other key management and delivery strategies

d. **Send Transmission Format**
  (1) Strategy
    (a) XML IC Trusted Data Object
    (b) Binary IC Trusted Data Object
    (c) Efficient XML Data Object
  (2) Report Receipt

e. **Receive Transmission Format**
  (1) Decrypt Message
    (a) IC Trusted Data Object Keys
    (b) Other key management and delivery strategies
  (2) Terminate Transmission Strategy
    (a) UnMarshal Binary Object to Normative XML
      1. XML IC Trusted Data Object
      2. Binary IC Trusted Data Object
      3. Efficient XML IC Trusted Data Object
    (b) Read Normative XML
  (3) Verify Data
    (a) Hash Digests
    (b) IC TDO Keys
  (4) Validate Data
    (a) W3C XML Schema - Format and Content
    (b) W3C Schematron - Conditional Rules
  (5) Report Receipt

f. **Repeat Cycle to View, Use, or Re-transmit**

Appendix C. Interoperability Testing Checklist

a. Background. There are many aspects of Interoperability Testing that cannot be achieved using XML Schema. This testing checklist applies to those things that can, and is intended to be fully automated, and available for use as an independent test tool from which results can be verified.

b. XML Verification Requirements

(1) Authoritative Resources

(2) Managed XML Namespace

(a) Single Reference Schema

(b) Namespace Management Conventions

(3) XML Schema Design

(a) Conformant XML Schema Design

(b) Tested XML Schema Naming and Design Rules

(4) Extensibility Conformance

(a) Conformant XML Schema Extension Design

(b) Conformant XML Schema Restriction Design

c. XML Validation Requirements

(1) Build Cycle

(a) Validate all XML Schema, XSLT Stylesheets, and Schematron Schema using respective W3C XML Schema

(b) Validate XML Test Instances using XML Schema for Content

(c) Validate XML Test Instances using Schematron Schema for Conditional Content

(d) Failed XML verification and validation events cause a failed build.

(2) Execution

(a) Validate Normative XML Instances using XML Schema and Schematron during Transmission Format Creation

(b) Validate Normative XML Instances using XML Schema and Schematron during Transmission Format Termination

(c) Failed XML verification and validation events cause a reportable incident.

(3) Reporting

(a) All XML Schema verification and validation events must be logged and reported.

Appendix D. Information Exchange Package Definition Content

a. Content

(1) dist

(2) doc

(a) index.html

1. purpose
2. requirements
3. dependencies
4. structure
5. installation
6. employment
7. maintenance
8. code-documents
9. graphics
10. changelog
11. license
12. distribution
13. classification

(b) readme

(3) json

(a) jsonld

(b) jsondata

(4) src

(a) milstd

1. mtfAPI
2. tdlAPI
3. vmfAPI

(5) test

(a) xml

(b) json

(c) software

(d) signature

(6) xml

(a) xsd

(b) xsl

(c) xmldata

(7) ledger

b. Implementation

(1) Message Validator API

(2) Message Sender API

(3) Message Receiver API

Appendix E. <u>Terms of Reference</u>

a. <u>MILSTD</u>. This refers to any information standards maintained to support unique military problem sets, and which have proscribed test and verification requirements.

b. <u>XML Schema</u>. An XML language used to specify data content type, format, and context.

c. <u>XPath</u>. An XML language which allows the identification and verification of information structures and data provided in XML formats.

d. <u>XSLT</u>. An XML language which uses iterative XPath evaluation to transform XML data. It is the core technology for relational rule evaluation.

e. <u>Schematron</u>. A standard which provides a normative namespace and framework for defining relational evaluation rules definition and providing results in an XML Schema defined format.