NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A CONFIGURATION FRAMEWORK AND IMPLEMENTATION FOR THE LEAST PRIVILEGE SEPARATION KERNEL

by

Chee Luan Quek

December 2010

Thesis Co-Advisors:	Cynthia E. Irvine Paul C. Clark

Approved for public release; distribution is unlimited

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** December 2010	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE A Configuration Framework and Implementation for the Least Privilege Separation Kernel			**5. FUNDING NUMBERS**
6. AUTHOR(S) Chee Luan Quek			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
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. IRB Protocol number: N/A.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The Least Privilege Separation Kernel (LPSK) configuration vector defines the initial secure state and the operational configuration of the kernel, including its security policies. Enhancements made to the LPSK functional specification necessitated substantial changes to the configuration vector data format defined previously. Moreover, the earlier format used an ad-hoc syntax, which did not adhere to any standard. This work leverages Extensible Markup Language (XML) to standardize the configuration vector format. The new configuration vector format is depicted in a XML Schema, and its limitations are discussed. A more compact binary representation is defined, with an offline tool provided to generate binary configuration vectors for the target platform. Creation of a configuration vector file is a laborious and error-prone task. A good user interface can ease the process by removing underlying complexities from users. Pertinent features of XML editors were assessed in a survey. Using these as requirements, an XML editor with a suitable graphical user interface was selected.

14. SUBJECT TERMS Configuration vector, Configuration vector tool, LPSK, , XML, XML editors, XML Schema, XSD			**15. NUMBER OF PAGES** 181
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

A CONFIGURATION FRAMEWORK AND IMPLEMENTATION FOR THE LEAST PRIVILEGE SEPARATION KERNEL

Chee Luan Quek
Civilian, Defence Science & Technology Agency, Singapore
B.Eng., National University of Singapore, 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2010

Author:	Chee Luan Quek
Approved by:	Cynthia E. Irvine Thesis Co-Advisor
	Paul C. Clark Thesis Co-Advisor
	Peter J. Denning Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Least Privilege Separation Kernel (LPSK) configuration vector defines the initial secure state and the operational configuration of the kernel, including its security policies. Enhancements made to the LPSK functional specification necessitated substantial changes to the configuration vector data format defined previously. Moreover, the earlier format used an ad-hoc syntax, which did not adhere to any standard. This work leverages Extensible Markup Language (XML) to standardize the configuration vector format. The new configuration vector format is depicted in a XML Schema, and its limitations are discussed. A more compact binary representation is defined, with an offline tool provided to generate binary configuration vectors for the target platform. Creation of a configuration vector file is a laborious and error-prone task. A good user interface can ease the process by removing underlying complexities from users. Pertinent features of XML editors were assessed in a survey. Using these as requirements, an XML editor with a suitable graphical user interface was selected.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

DOM	Document Object Model
DTD	Document Type Definition
GUI	Graphical User Interface
HMAC	Keyed-Hash Message Authentication Code
KVM	Keyboard, Video and Mouse
LPSK	Least Privilege Separation Kernel
MAC	Mandatory Access Control or Message Authentication Code or Medium Access Control
MLS	Multi-Level Secure
NIST	National Institute of Standards and Technology
PAS	PIFP Acyclic Subset
PIFP	Partitioned Information Flow Policy
PL	Privilege Level
PTP	Partition To Partition
SAK	Secure Attention Key
SAX	Simple API for XML
SKPP	Separation Kernel Protection Profile
STR	Subject To Resource
TCX	Trusted Computing Exemplar
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSD	XML Schema Definition Language

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to express my gratitude to both my thesis co-advisors, Professor Cynthia Irvine and Paul Clark, for their patience and guidance through the course of this thesis, and to Professor Don Brutzman for loaning us a copy of his department's XMLSpy license. I would also like to thank my Singapore sponsor, the Defence Science & Technology Agency (DSTA), for giving me the opportunity to pursue my postgraduate studies, as well as the professors and lecturers at NPS who have taught me, and from whom I have learned a lot. I am also grateful to my family for giving me support and continuous encouragement during my course of study at NPS.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The operational configuration of a separation kernel is provided as configuration data that the kernel interprets during initialization. The configuration data for the Least Privilege Separation Kernel (LPSK) of the Trusted Computing Exemplar (TCX) project [1] is called the LPSK configuration vector. The configuration vector defines the initial secure state and the operational configuration of the LPSK. It reflects the policies that regulate the information flow among resources, which may include the mandatory access control (MAC) policies. The correctness of the configuration vector is critical for ensuring the intended security policy is enforced by the kernel. A configuration vector tool translates and presents configuration data to the system administrator in a visual form that enables him to create valid configuration vectors more easily.

A. MOTIVATION

Since the configuration vector format was last defined, several enhancements had been introduced in the LPSK Functional Specification [2]. In addition, the structure of the earlier LPSK configuration vector human-readable format used an ad-hoc syntax that did not adhere to any standard. The binary format could not be scaled to support an increased number of partitions, processes, and resources without significantly increasing the file size of the configuration vector to be loaded into the LPSK. This posed a problem for platforms with limited memory resources. To address these problems, it was imperative to define a new configuration vector format and to develop a configuration vector tool that supports the new format.

B. PURPOSE OF STUDY

Extensible Markup Language (XML) [3] is an open standard of the W3C and has been widely adopted in many real world applications to structure and display data. This thesis project sought to develop an offline tool that is able to format the human-readable configuration vector for the LPSK in XML. Two major elements were developed as part of this work. The first was a XML Schema. The second was a program that reads and converts a LPSK configuration vector between XML and binary representation for the LPSK, validates the vectors against the schema, and tests them against the rules for configuration values and bounds. The primary research objective was to determine whether an XML-based representation of the LPSK configuration is viable, determine whether it is a better representation of the LPSK configuration relative to other formats, how to implement the XML Schema that formalizes the configuration vector data structure and validation rules, and to determine the testing and validation required. The task of creating configuration vectors is tedious and error-prone because there are many settings to configure. A subsidiary objective of this project is to recommend a graphical user interface that is useful to system administrators when they create configuration vectors and helps to minimize human-generated configuration errors.

C. THESIS ORGANIZATION

The remainder of the thesis is organized as follows: Chapter II begins by providing the background on the LPSK, the LPSK configuration vector and the configuration vector tool. It also provides a brief description of related XML technologies, and summarizes the benefits of an XML representation of the LPSK configuration vector. Chapter III outlines the high level and detailed requirements for the tool. Based on these requirements, Chapter IV surveys existing XML editors; it states the selection criteria and outlines the selection process for suitable XML editors to meet the Graphical User Interface (GUI) requirements for the tool. Chapter V describes the design decisions and implementation of the various components of the tool - the XML Schema, the binary configuration vector format, the format conversion utility and the Keyed-hash Message Authentication Code (HMAC) utility. The detailed installation procedure and instructions for using the Format Conversion utility and HMAC utility is documented in Appendix A. Extensive testing was performed on the implementation. Then, Chapter VI outlines the tests that were performed and summarizes the test results. The test setup and details of the test procedures are documented in Appendix B. Chapter VII concludes by summarizing the results achieved, discussing related work, and providing recommendations for further work.

II. BACKGROUND

The Trusted Computing Exemplar (TCX) project is an on-going effort by Naval Postgraduate School Center for Information Systems Security Studies and Research (NPS CISR) to provide a working prototype that demonstrates the process of building high assurance systems. One of the deliverables is a least privilege separation kernel (LPSK) that is suitable for use in multilevel secure (MLS) embedded systems. The first three sections of this chapter provide background on the LPSK, the LPSK configuration vector and the configuration vector tool. The main contribution of this thesis is to design and develop a new LPSK configuration vector using Extensible Markup Language (XML). A brief description of XML technologies related to this thesis is also presented. The benefits of using XML to represent the configuration vector are presented in the last section of this chapter.

A. THE LEAST PRIVILEGE SEPARATION KERNEL

In emergency IT systems, sensitive Personally Identifiable Information (PII) data, such as electronic medical records, are protected with strict access control mechanisms to ensure the privacy of individuals. During emergencies, it is desirable that these data be accessible by first responders who often require quick access to them in order to handle the crisis. When the emergency is over, the IT system should revert to its original configuration, where all temporary accesses to PII data are revoked. A possible architecture to realize this is through a specialized separation kernel.

The concept of separation kernels was introduced in 1981 by Rushby [4]. It described a separation kernel to be an environment that appears “as if each regime is a separate, isolated machine, and information can only flow from one machine to another along known external communications lines. One of the properties we must prove for a separation kernel, is that there are no channels for information flow between regimes other than those explicitly provided.” In 2007, the National Security Agency published the Separation Kernel Protection Profile (SKPP) [5], which is a security requirements specification for separation kernels suitable to be used in environments requiring high robustness. In this specification, the primary security function of a separation kernel is to isolate the resources within a MLS system into security policy-equivalence classes (partitions), and to enforce the rules for authorized information flows between and within partitions.

One of the initiatives under the TCX project is the development of a LPSK that meets the SKPP requirements. It further enforces the Principle of Least Privilege [6] by granting only the least set of privileges to subjects. The rationale for doing so is to limit the amount of damage that results from accidental errors or malicious executions. Trusted subjects are allowed to execute under relaxed MLS policies that allow information exchanges between resources of different equivalence classes. In the example of an emergency IT system, trusted subjects may be configured to allow the first responders temporary access to restricted data that are not available to them under normal circumstances.

Figure 1 depicts the concept of partitions, subjects, other resources and information flows in the LPSK. In this example, there are three partitions, with five subjects and seven other resources allocated to their respective partitions. The arrows represent the directions of information flow between subjects and other resources that are allowed by policy. The isolation and information flow policies are defined within the separation kernel's configuration data, i.e., the LPSK configuration vector.

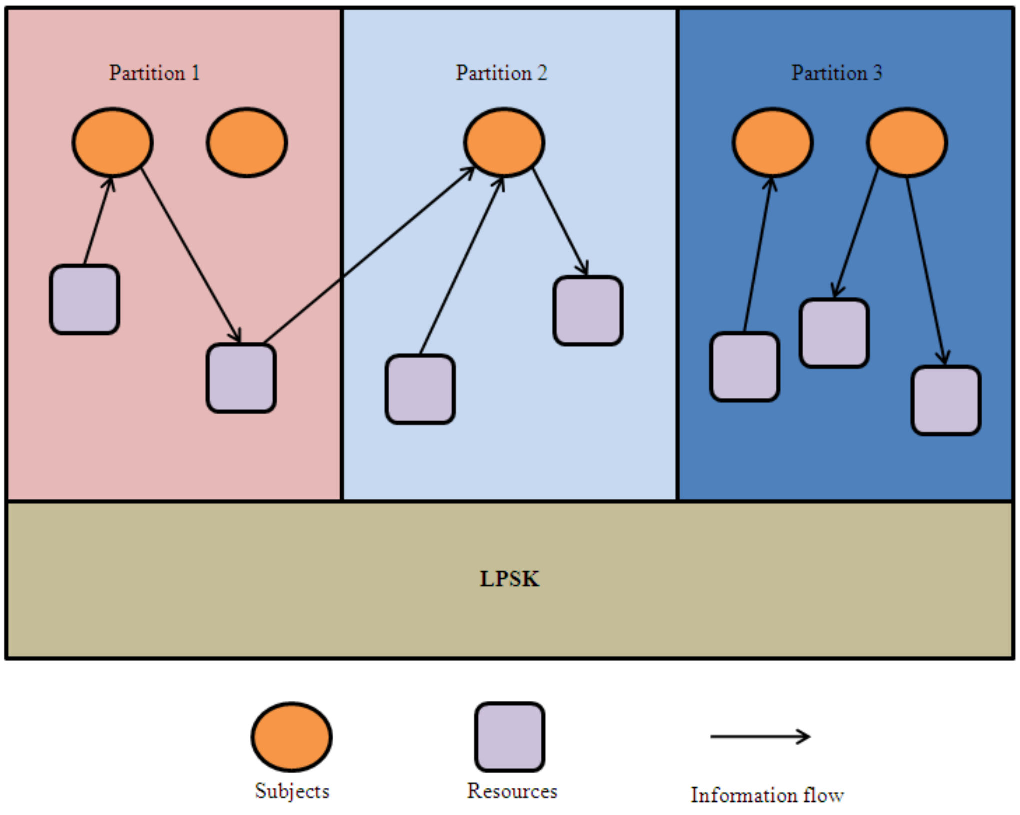


Figure 1. Concept of partitions, subjects, other resources and information flow. (After [2])

B. THE LPSK CONFIGURATION VECTOR

The configuration vector is required by the LPSK to establish an initial secure state. It defines the LPSK runtime scheduling, processes and subjects (active entities) and other resources (I/O and network devices, segments and synchronization primitives) that are exported by the LPSK. Subjects and other exported resources are allocated to partitions as defined by the configuration vector. The configuration vector also defines the LPSK's security policy regarding inter-partition information flow and subject-to-resource access bindings. This allows the LPSK to isolate subjects and other exported resources from one another and control the information flows between subjects and other resources.

Prior to this work, there was a configuration vector format defined [7]. Since that work was completed, the LPSK Functional Specification has matured. As a result, the early configuration vector lacked many of the new settings that have since been specified. For example, it did not support the configuration of audit mechanisms that would be incorporated in the future. As another example, device configurations are also hardcoded in the LPSK. A new configuration vector would allow audit events and devices to be configured on the target platform without having to recompile the LPSK.

In addition, the number of partitions, processes and subjects was hardcoded in the existing LPSK configuration vector definition. Because of this, the file size of the configuration vector was fixed to allow for the maximum number of partitions, processes and subjects that the LPSK supports. Such an approach was still feasible because the current LPSK prototype was limited to eight partitions and one process per partition. However, since the LPSK will eventually support up to 256 partitions and 512 processes, per the specification, the footprint of the configuration vector may have become too large for embedded devices with limited memory resources.

Defining these configurations within a configuration vector instead of hard-coding them in the kernel provides flexibility to change settings without recompiling a new kernel, at the expense of more configuration responsibilities on the administrator. A good configuration vector tool is required for this task.

C. THE LPSK CONFIGURATION VECTOR TOOL

A LPSK Configuration Vector Tool was developed in 2009 [7]. The outcome of that research was an offline tool with a graphical user interface that allows a trusted user to visualize, create and organize LPSK configuration vectors. The tool was developed using Java Swing components and was based on the primitive LPSK configuration vector format that existed at that time. A new tool that supports the new configuration vector format was required. Creating a configuration vector for the LPSK is a complex and laborious task because there are many settings to configure. The main purpose of this tool is to assist a user when creating configuration data that correctly reflect the intended system security policy. Usability of the new tool is thus an important aspect that must be considered.

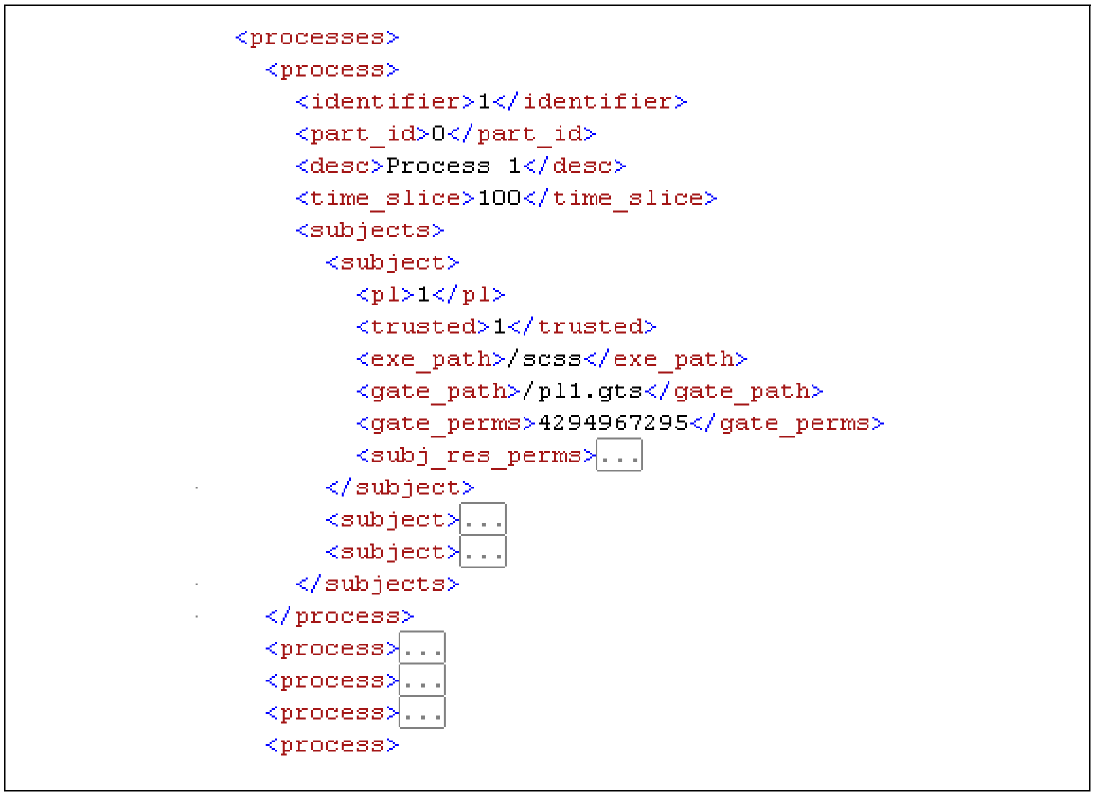
The structure of the earlier LPSK configuration vector format was ad-hoc syntax and did not adhere to any standard. One of the other recommendations made in the 2009 work was the development of a new XML-based configuration vector format.

D. OVERVIEW OF XML AND RELATED STANDARDS

There are many standards, technologies and publications that are related to XML. This section briefly describes XML technologies and standards that are related to this thesis. Detailed specifications of XML are usually publicly available from the World Wide Web Consortium (W3C).

1. Extensible Markup Language (XML)

XML is a standard markup language recommended by the W3C for representing data in structured formats using tags. The tags provide information about the content enclosed. Elements in XML documents can be nested to form a structure. In the example shown in Figure 2, there are multiple *processes* in a LPSK configuration vector; each *process* contains multiple *subjects* and each *subject* has subject-to-resource permissions (*subj_res_perms*) associated with it. Details of the specifications can be found in [3].



```
<processes>
  <process>
    <identifier>1</identifier>
    <part_id>0</part_id>
    <desc>Process 1</desc>
    <time_slice>100</time_slice>
    <subjects>
      <subject>
        <pl>1</pl>
        <trusted>1</trusted>
        <exe_path>/scss</exe_path>
        <gate_path>/pl1.gts</gate_path>
        <gate_perms>4294967295</gate_perms>
        <subj_res_perms>...
      </subject>
      <subject>...
      <subject>...
    </subjects>
  </process>
  <process>...
  <process>...
  <process>...
  <process>
```

Figure 2. An example of nested XML elements

XML has been widely adopted in many real world applications to structure, display, standardize and exchange data. Examples of the use of XML in the security domain include the OASIS WS-Policy framework policy assertions representation [8] and SOAP Message Security (e.g., Web Services Security Policy Language [9]) that allows Web Services to express security policies.

2. XML Schema

An XML Schema provides a grammar that is used to describe the structure of XML documents based on a set of pre-defined elements. These are defined in the XML Schema definition language specifications that can be found in [10], [11] and [12]. Since an XML Schema is defined using XML, it can also be edited using any XML editor. Sometimes, a Document Type Definition (DTD) [13] is used in place of an XML Schema to enforce a valid XML document structure.

An XML document may exist without a DTD or XML Schema. Without them, the computer can only determine if the document is well-formed. By coupling the XML-based LPSK configuration vector with a well-defined DTD or XML schema, it can be further used by computers to automatically validate the configuration vector to check for inconsistencies and errors.

3. Document Object Model and Simple API for XML

The Document Object Model (DOM) provides a standard set of objects for representing HTML and XML documents, a standard model of how these objects can be combined, and a standard interface for accessing and manipulating the objects [14]. SAX is an alternative to DOM [15]. It was originally a Java-only application programming interface (API) and was the first widely adopted API for XML in Java. It has since evolved and there are currently implementations available for several programming language environments other than Java. Both DOM and SAX provide the APIs for applications to parse and traverse XML documents and validate them against the DTDs and XML Schemas defined.

E. BENEFITS OF AN XML-BASED CONFIGURATION VECTOR

The benefits of using XML to represent the LPSK configuration vector are multifold. XML is text-based, and is an open standard that is vendor and platform independent. There exist a lot of tools that are readily available to view and edit an XML-based configuration vector. It is thus convenient, flexible and viable to be adopted as a standard format. XML is powerful in representing a hierarchical tree structure with multiple nested levels, making it natural for the configuration vector to store relationships between partitions, processes, subject-to-resource permissions. Using an XML over a binary representation makes the LPSK configuration vector both human-readable and machine-readable. This would make it possible for a human to scrutinize and detect accidental or malicious modification in the configuration. In addition, the XML schema would form the basis for user-configurable rules that allows flexibility and extensibility. It also allows relevant information to be readily transformed from the XML representation to multiple display formats and native representations. The main disadvantage of using XML is that it increases the size of the configuration vector, requiring more storage resources.

F. SUMMARY

This chapter outlined the background the TCX LPSK, the LPSK Configuration Vector and the Configuration Vector Tool. It also introduced XML standards relevant to this thesis, and briefly described the benefits of having an XML-based configuration vector. The next chapter describes the functional requirements of the LPSK Configuration Vector Tool.

III. CONFIGURATION VECTOR TOOL REQUIREMENTS

The U.S. Government Protection Profile for Separation Kernels (SKPP) [5] states the requirements for a configuration function that provides configuration vector generation and validation capability to convert configuration vectors from a human-readable form into a machine-readable (e.g., binary) form, and vice versa, while preserving the semantics of the data. The LPSK Configuration Vector Tool is an application that provides this capability.

A trusted individual or authorized administrator prepares a human-readable representation of a configuration vector using the specified format, and saves it to a file. The file is then given as input to the LPSK Configuration Vector Tool, which produces as an output, a binary representation of the configuration vector. Multiple configuration vectors may be created. The binary configuration vectors are then distributed with the run-time LPSK, non-LPSK executables and other products that need to be installed. Subsequently, during the booting process, a *LPSK Initializer* program receives a binary configuration vector from the *LPSK Boot Loader* program, and loads and configures the run-time LPSK. Secondary storage segments and applications, as specified in the configuration vector, are also loaded at this time.

A. HIGH-LEVEL REQUIREMENTS

The high-level requirements of the LPSK Configuration Vector Tool are derived from the SKPP requirements for Configuration Tool Design. The LPSK Product Functional Specification [2] refines these requirements, which are described here.

1. The LPSK Configuration Vector Tool shall be an offline tool that shall be capable of executing on a commodity operating system.
2. The LPSK Configuration Vector Tool shall take as an input a human-readable configuration vector and the file name where the output binary configuration vector shall be written. The LPSK Configuration Vector Tool shall produce a binary

equivalent of the input human-readable vector, which it will output to the specified file. The binary configuration vector shall comply with the data format specified in *Section B, Detailed Requirements*.

a. The LPSK Configuration Vector Tool shall verify that the input human-readable configuration vector complies with the data format. If the tool detects a syntax error, then an informative message shall be displayed, and the processing shall be stopped without producing a binary vector.

b. If the input human-readable configuration vector is syntactically correct, but the LPSK Configuration Vector Tool detects an invalid setting in the input human-readable configuration vector (e.g., a number that is outside of the defined range for a given field), then the tool shall display an informative message and stop the processing without producing a binary vector.

3. The LPSK Configuration Vector Tool shall take as an input a binary configuration vector and the file name where the output human-readable configuration vector shall be written. The LPSK Configuration Vector Tool shall produce a human-readable equivalent of the input binary vector, which it will output to the specified file. The output human-readable configuration vector shall comply with the format specified in *Section B Detailed Requirements*.

a. If the LPSK Configuration Vector Tool detects that the binary configuration vector is not compliant with the specified format, then the tool shall display an informative message, and the processing shall be stopped without producing a human-readable vector.

b. If the LPSK Configuration Vector Tool detects an invalid setting within the binary configuration vector, then the tool shall display an informative message, and the processing shall be stopped without producing a human readable vector.

4. The LPSK Configuration Vector Tool shall provide the option to calculate and display a message authentication code (MAC) for an input configuration vector, whether the input configuration vector is binary or human-readable. This is to place a cryptographic seal on the generated configuration vector, so that the user can be assured

that the integrity of the configuration vectors has not been compromised between when they are generated and when they are verified.

a. The message authentication code shall be displayed as the ASCII representation of its hexadecimal value.

b. The encryption key used to generate the message authentication code will be generated from text entered by a trusted individual when prompted by the LPSK Configuration Vector Tool.

5. The LPSK Configuration Vector Tool may also provide a user interface to assist in the creation of a human-readable configuration vector.

Figure 3 provides an overview of the flow of interactions of the trusted user using the tool.

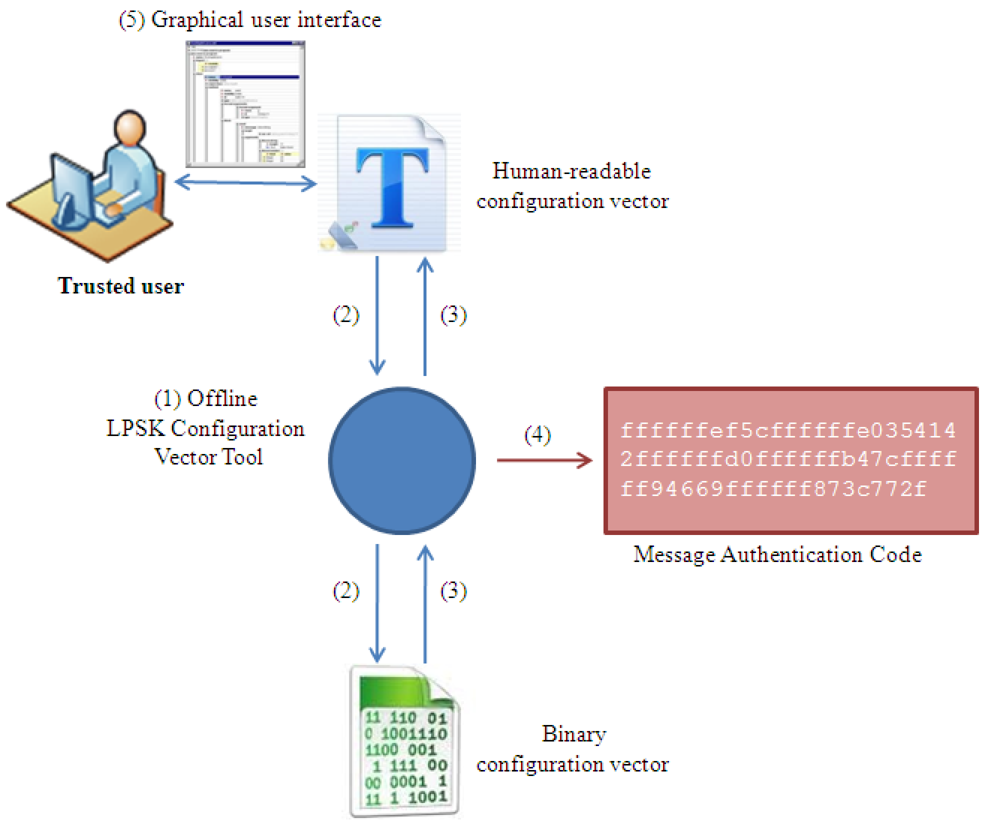

(5) Graphical user interface
Human-readable configuration vector
Trusted user
(2)
(3)
(1) Offline LPSK Configuration Vector Tool
(4)
ffffffef5cffffffe035414
2ffffffd0ffffffb47cffff
ff94669ffffff873c772f
Message Authentication Code
(2)
(3)
Binary configuration vector

Figure 3. Overview of the flow of interactions of the trusted user using the tool (After [2])

B. DETAILED REQUIREMENTS

1. Configuration Vector Format

Both human-readable and machine-readable forms of the configuration vector shall contain the fields in the order defined below. These data requirements are derived from [2]. The data types shall be defined in *Chapter V, Design and Implementation*.

a. Header

The following information shall be in the header:

- Magic number (binary vector)
- Version number
- Vector description (a human-readable ASCII description of the vector)

b. Audit Buffer Configuration

Audit buffer configuration is declared near the beginning of the configuration vector because it may impact the audit records that are generated during initialization.

- Enable audit? (optional, default=yes)
- Maximum number of audit records to buffer (between 1 and 65535)
- Action when buffer is full (overwrite, halt, shutdown) (optional, default=halt)
- Time delay prior to shutdown (optional, between 0 and 60 (seconds) default=no delay)

c. Run-Time LPSK

The declaration of the Run-Time LPSK shall include the following:

- Executable information:
    - The path to the kernel executable on secondary storage

    - The path to the kernel call gate configuration on secondary storage

- Configuration of the LPSK message area:
    - Display LPSK messages to the user? (default = no)
    - Type of messages (status messages, partition with focus, both)
    - Number of video character lines to use for messages
- Reserved memory ranges: (0 or more of the following)
    - Start address
    - Size

d. Partitions

The declaration of partitions includes the following

- Duration for one round-robin scheduling of all processes in all partitions (positive value)
- Declaration of each partition: (1 to 256 of the following entries)
    - Partition ID (0 through 255 and is unique across the vector)
    - Partition description (a human-readable ASCII description of the partition)
    - Percentage of a round-robin schedule
    - Percentage of non-kernel primary memory allocated to the partition
- Partition ID of the secure attention key (SAK) handling partition. This is an active partition. An active partition is a partition that has a process in its set of resources.

- Partition ID of the initial partition with focus. This is an active partition.

e. Partition Flow Policies

The declaration of flow policies includes the following:

- Partition-to-partition (PTP) policy (0 or more of the following)
    - Partition ID of a partition with a subject requesting a flow. This is an active partition.
    - Partition ID of a partition with a resource
    - Mode of access for the flow between the above two partitions
- Halt if a subject-to-resource (STR) policy is not included in the PTP (default=no)
- Partitioned information flow policy subset (PAS): (0 or more of the following)
    - Partition ID of a partition with a subject requesting a flow. This is an active partition.
    - Partition ID of a partition with a resource
    - Mode of access for the flow between the above two partitions (No access (NA), Read-only (RO), Read-write (RW), Write-only (WO))

f. Resources

- Data Segments (0 or more of the following entries)
    - Path
    - Data Segment description (a human-readable ASCII description of the data segment)

    - Partition ID of the home partition
    - Privilege level to be assigned to the data segment (0 to 3)
    - Start State (swapin, swapout)
    - Audit (optional, default=none)
        - When a swapin is requested (success, failure, both)
        - When a flush is requested (success, failure, both)
        - When a swapout is requested (success, failure, both)
- Memory Segments (0 or more of the following entries)
    - Memory Segment ID (unique across the vector)
    - Memory Segment description (a human-readable ASCII description of the memory segment)
    - Size
    - Partition ID of the home partition
    - Privilege level to be assigned to the memory segment (0 to 3)
    - Audit (optional, default=none)
        - When created (success, failure, both)
- Eventcounts (0 to 64 of the following entries)
    - Eventcount ID (0 through 63)
    - Eventcount description (a human-readable ASCII description of the eventcount)
    - Partition ID of the home partition
    - Audit (optional, default=none)
        - When an advance is requested (success, failure, both)

        - When a read is requested (success, failure, both)
        - When an await is requested (success, failure, both)
        - When a wakeup occurs (success, failure, both)
- Sequencers (0 to 64 of the following entries)
    - Sequencer ID (0 through 63)
    - Sequencer description (a human-readable ASCII description of the sequencer)
    - Partition ID of the home partition ID
    - Audit (optional, default=none)
        - When a ticket operation is requested (success, failure, both)
- Devices (variable number of the following entries). A device can be a keyboard, a network device, a screen device or a PL1 message device. The total number of keyboard devices, the total number of PL1 message devices, and the total number of screen devices must be between 0 and 1.
    - Major number
    - Minor number
    - Type (DATA or CONTROL)
    - Device description (a human-readable ASCII description of the device)
    - Partition ID of the home partition
    - Attribute (multiplexed or dedicated) [may not be configurable]

    - If dedicated and not a KVM device, the partition ID with initial access
- If a keyboard device:
    - Incoming scan code buffer size (between 32 and 1024).
- If a network device:
    - Initial state (started, stopped)
    - Encryption required?
    - Read behavior (promiscuous, non-promiscuous) (optional, default=non-promiscuous)
    - Incoming frame buffer size (between 32 and 256)
    - Outgoing frame buffer size (between 32 and 256)
    - MAC address (optional). The MAC address format is XX:XX:XX:XX:XX, where X is a hexadecimal digit between 0x0 and 0xF.
- If a screen device:
    - Number of video character lines
- If a PL1 message device:
    - Number of video character lines
- Audit (optional)
    - When a read data operation is requested? (success, failure, both)
    - When a write data operation is requested? (success, failure, both)

        - When a read metadata operation is requested? (success, failure, both)
        - When a write metadata operation is requested? (success, failure, both)
- Processes (1 to 512 of the following entries)
    - Process ID (unique across the vector)
    - Partition ID of the home partition. This is an active partition.
    - Process description (a human-readable ASCII description of the process)
    - Percentage of the partition time slice (optional, default=equally split)
    - Subject declaration (at least 1, and a maximum of 3 per process)
        - Trusted?
        - Path to an executable file
        - Path to its call gate configuration file (optional, default=none)
        - Privilege level (between 0 and 3)
        - Kernel gate calls callable by subject (optional, default=none)
        - Subject-to-resource permissions. It was decided to associate permissions with the subjects rather than with the other resources (i.e., rather than ACLs) because it appeared to be the best way of describing the permissions related to sending signals to subjects.

    - Resource type (eventcount, sequencer, device, subject, etc) to which the subject has access
    - Resource ID
        - For eventcounts, sequencers, and memory segments, this is the ID of the resource declared earlier.
        - For data segments this is the path on secondary storage.
        - For devices, this is <major, minor, DATA> or <major, minor, CONTROL>.
        - For subjects this is the <process ID, privilege level (PL)> pair.
    - Permitted actions on the resource by the subject
        - For eventcounts, data segments, memory segments, network devices, screen devices, and PL1 message devices and other subjects, they are RO, RW and WO.
        - For sequencers, they are RW.
        - For keyboards, they are RO.
- Audit (optional auditable events, default = no auditing of the subject)
    - When an interrupt is invoked
    - When a signal is received
    - When sending a signal (success, failure, both)

- When reading from a device (success, failure, both)
- When writing to a device (success, failure, both)
- When reading an eventcount (success, failure, both)
- When advancing an eventcount (success, failure, both)
- When waiting on an eventcount (success, failure, both)
- When a wakeup from an eventcount occurs
- When ticketing a sequencer (success, failure, both)

In addition, the tool shall validate a configuration vector against the following rules:

- There is at least one active partition.
- The total of CPU time allocated to all partitions is exactly 100%.
- A process can only belongs to a partition that has been allocated a time slice greater than 0%.
- A process is allocated a time slice greater than 0%.
- The sum of the time slices allocated to all processes in a partition is equal to 100% of the time slices allocated to that partition.

2. XML Representation

As mentioned in Chapter II, the human-readable configuration vector shall be represented in XML. A XML Schema shall be developed to define the XML elements and their attributes, the number and structure of each element and restrictions on the allowed data types, allowed values, boundary values (value ranges or string lengths) and the cardinality relationships between elements.

3. Binary Representation

To avoid having to re-compile the LPSK for each target platform whenever the configurations vary, or to avoid requiring a statically large configuration vector size, the binary format shall allow the size of the configuration vector to scale dynamically according to the number of partitions, processes, subjects and resources configured. With this enhancement, the size of the configuration vector shall automatically adjust based on the number of partitions, processes, subjects, and resources configured.

4. Graphical User Interface (GUI)

The LPSK Configuration Vector Tool shall provide a Graphical User Interface (GUI) for users to visualize, create and edit configuration vectors in XML format. The GUI serves to simplify the editing process so that the user does not have to be conversant with the configuration vector XML syntax.

a. Basic Functional Requirements

The tool shall provide the following basic editing functionalities:

- Create a new configuration vector.
- Select and load an existing configuration vector based on the file name provided.
- Load a partially completed configuration vector.
- Save a configuration vector in a file name specified by the user, and automatically check the configuration for errors before saving the configuration vector.
- Save a configuration vector to XML even if it is partially completed or contains errors.

- Exit the tool or discard changes to a configuration vector without saving it.

The user shall to be able to use the XML Schema developed within the GUI. The GUI shall display the fields according to the format defined in the XML Schema. In addition, it shall allow the user to check that an XML configuration vector is both well-formed and valid according the XML Schema defined. This makes it easier for the user to recognize, diagnose and recover from errors while creating the XML configuration vector. Validation may be performed either on-demand or on-the-fly. The tool shall:

- Provide on-demand validation that allows validation of the XML configuration vector using the XML Schema defined anytime during the edit process, and prompt the user if an error exists in the configuration vector.
- Automatically validate the configuration vector against the XML Schema and prompt the user if there are errors whenever he tries to save it.

b. Usability Requirements

Usability is a qualitative attribute that assesses the ease-of-use of user interfaces. The previous work on the LPSK Configuration Vector Tool [7] also made several recommendations to improve the usability of the tool. One aspect of usability of the GUI is the set of useful but non-essential features to make it easier for users to perform the task of editing configuration vectors and to minimize the possibility of errors. On-the-fly validation is a useful feature since it provides instant feedback to the user, so that he is continuously guided while editing a configuration vector. The GUI should also assist the user by directing him to the location in the configuration vector where the error occurs. This improves usability by making debugging and error recovery easier for users.

5. Format Conversion

To convert a configuration vector from XML into binary form, the tool shall first validate the XML configuration vector against the XML Schema and rules defined. If the input configuration is not well formed or invalid (due either to invalid values or syntax errors), the tool shall display an error message, and halt the processing. Otherwise, the tool shall save the binary output to the file name specified.

To convert a configuration vector from binary into XML form, the tool shall first convert the binary configuration to its XML representation, and validate the latter against the XML Schema and rules defined. If the XML document is not well-formed or invalid, the tool shall display an error message, and halt the processing. Otherwise, the tool shall save the XML output to the file name specified.

6. Keyed-hash Message Authentication Code Generation

The tool shall provide a utility for the user to generate a keyed-hash message authentication code (HMAC) for a configuration vector. The message authentication code for a given configuration vector is calculated using an approved cryptographic hash function in combination with a secret key supplied by the user. It shall take as an input the file name of a XML or binary configuration vector and a text password. The cryptographic algorithm for generating the HMAC shall be a NIST-approved hash implementation with a message digest size of at least 256 bits [16]. The tool shall first verify that the configuration vector is syntactically correct. It shall then generate an encryption key based on the supplied password, use it to calculate the message authentication code of the input file, and display the code to the user in hexadecimal format.

C. SUMMARY

This chapter outlined the high level and detailed requirements of the LPSK Configuration Vector Tool, comprising a suite of utilities to support the creation of LPSK configuration vectors. These requirements resulted from the SKPP specification, the LPSK Product Functional Specification and previous work done on the LPSK

Configuration Vector Tool. The next chapter describes the evaluation process to ascertain if a suitable XML editor exists on the market to fulfill the GUI requirements. Chapter V describes the design and implementation of the LPSK Schema and other components to meet the remaining requirements.

IV. EVALUATION OF XML EDITORS

There are essentially two categories of XML editors – textual and graphical. Textual editors let users edit the element tags and content directly. They require the users to have knowledge of the XML syntax. Graphical editors render the XML documents so that users do not have to manipulate the XML tags directly. Hence, they do not require users to know the XML syntax. Most graphical editors provide an option for users to toggle between a graphical view and a text view.

A XML-based LPSK configuration vector provides the flexibility to edit the configuration vector using any XML editor. An experienced system administrator could capitalize on the use of the XML format and edit the configuration file with either a text-based or graphical editor. However, having a graphical user interface (GUI) to manipulate the configuration vector would be convenient for the user to visualize the configuration vector, including relationships between its elements, and ease the creation of configuration vectors by removing underlying complexities from the user. This may help to eliminate inadvertent errors.

There are many XML editors available on the market. This chapter outlines the selection criteria for a suitable XML editor for the LPSK Configuration Vector tool and the outcome of the evaluation process. The objective of this evaluation was to determine if there are XML editors on the market well-suited for the LPSK Configuration Vector tool, before proceeding to develop one in-house.

There are existing reviews of XML editors ([17]). However, the reviews are based on older versions of the editors, so one would expect the reviews to be irrelevant today as the companies continue to improve their products. Besides, requirements vary depending on the intended usage, and the LPSK has unique requirements. It was thus timely to do a new evaluation to select the most suitable software to meet the needs of the TCX project.

A. INITIAL SELECTION CRITERIA

This section provides a brief description of the editors surveyed. The starting point was the list of XML editors from an article published in Wikipedia (http://en.wikipedia.org/wiki/List_of_XML_editors). The first consideration was whether the products are actively maintained. Defunct products were removed from consideration, which left a total of fifty-eight (58) editors to be evaluated. This initial evaluation of the 58 was based on the first three (3) criteria explained below.

1. Graphical XML Editor

Since this research effort was focused on using the XML editor to provide the GUI for the LPSK Configuration Vector Tool, text-based editors were eliminated from consideration.

2. XSD Validation

The requirement for XML validation using XML Schema (XSD validation) is for the editor to be able to use the XML Schema developed within the editor. This allows the user to check that an XML configuration vector is both well formed and valid according to the XML Schema. To meet the configuration vector user interface requirements, the editor shall automatically validate the document and prompt the user if there are errors whenever he tries to save it. This makes it easier for the user to recognize, diagnose and recover from errors while creating the XML configuration vector. Validation may be performed either on-demand or on-the-fly. On-demand validation requires the user to manually trigger the editor to check if the XML document is valid. With on-the-fly validation, the editor continuously checks whether the right elements are being used. On-the-fly validation is a useful feature since it provides instant feedback to the user, so that he is continuously guided while editing a configuration vector.

3. Availability of U.S. Vendor

Since the LPSK could eventually be used by the U.S. military organizations such as the U.S. Department of Defense (DoD), it would also be helpful to know if the editor can be purchased through a U.S. vendor. The software can be developed in another country, but if it can only be purchased through the product Web site and outside of the U.S., then constraints in procedures could make it difficult for U.S. government organizations to purchase it.

The following are additional information, but were not used to eliminate any tools from consideration.

4. OS Platform Support

Since the LPSK Configuration Vector Tool is intended to be used offline, the OS platform support is not crucial. Nonetheless, a multi-platform tool is desirable because this would give the users greater flexibility during deployment.

5. Type

The type indicates if the product is available as a standalone editor or if it is a plug-in. If the product is only available as a plug-in, then there would be additional cost to purchase and maintain an additional host application.

6. License Pricing

The prices of the products were obtained directly from information published on the product Web site. Clearly, an extremely costly tool would be less attractive. However, the editor will not be required on a large number of platforms within the enterprise. Thus, this is a secondary consideration.

7. Open-source

Access to the source code allows the possibility of analysis for malicious code and unintended functionality in the product. It would also allow continued maintenance of the software after the product is no longer in production.

8. License Type

Knowledge of the type of licenses required for the editors is important to ensure license agreements will not be violated.

B. OUTCOME OF INITIAL SELECTION

Most editor vendors provide evaluation licenses for a limited period. For such editors, they were assessed using a sample LPSK configuration vector. Otherwise, assessment was based on information available from the product Web site. A few of the products could not be downloaded as their Web sites were down during the evaluation process. Table 1 shows the result of the initial evaluation of the XML editors, with the products listed in alphabetical order. A ● indicates the product fully met the requirements, ◖ indicates it partially met the requirements and ✖ indicates it did not meet the requirements. Table 2 shows additional information related to these editors. The list of fifty-eight (58) products was narrowed to seven (7). The editors elected were Altova XMLSpy, Liquid XML Studio, <oXygen/> XML Editor, Qxmledit, Stylus Studio, XML Notepad and XMLSpear. Only <oXygen/> XML Editor, XML Notepad and XMLSpear provided on-the-fly XSD validation.

Table 1. Result of the initial evaluation of XML editors (This information is accurate as of September 2010)

Product		Graphical XML Editor	XSD Validation	Availability of U.S. vendor	Remarks
1.	AjRa XML Editor 1.0	●	✖	-	1. XML validation feature was available, but the configuration vector could not be validated using the schema. 2. Software reported that it could not find the declaration of element ‘vector’. 3. Software hung and had to be re-started frequently.
2.	Altova XMLSpy 2010	●	●	●	1. Corporate Sales: us-sales@altova.com. All other matters: us-office@altova.com. 2. Validated the configuration vector whenever the configuration was saved. 3. Software provided on-demand validation only.
3.	Amaya 11.3	●	✖	-	

Product		Graphical XML Editor	XSD Validation	Availability of U.S. vendor	Remarks
4.	Angur XML Visualizer 1.01	✗	✗	-	Figure 4 shows an example of the visualization of this tool provided for a sample configuration vector, which is fairly poor. The XML content could not be seen, and there is no XML source view.
5.	Barium Visual XSL Transformer 1.5	✗	✗	-	An XSL transformation tool.
6.	Beacon Editor	●	No information	-	Could not be downloaded as its Web site domain name had expired; it was eliminated from further consideration.
7.	CEDIT XML	✗	✗	-	A text-based XML editor.
8.	Civ4 Mod Editor	✗	No information	-	Could not load the sample LPSK configuration vector. The user interface is not intuitive and there is no user manual or online help available.

Product		Graphical XML Editor	XSD Validation	Availability of U.S. vendor	Remarks
9.	Cladonia Exchanger XML Editor 3.2	✖	◖	✖	1. No U.S. contact listed on Web site. 2. Did not automatically validate the XML configuration vector when it was saved.
10.	codefunk XML Editor 1	●	✖	-	
11.	Dataset Editor 0.1	●	No information	-	Could not run; an error message was shown indicating that a DLL (dynamic link library) was missing; hence software was eliminated from further consideration.
12.	Debugging XSLT stylesheets using the Saxon processor 1.0	✖	✖	-	For debugging XSLT style-sheets.
13.	Dexter-XSL 0.3	✖	✖	-	A tool for producing XSLT 1.0 compliant stylesheets to style arbitrary XML data

Product		Graphical XML Editor	XSD Validation	Availability of U.S. vendor	Remarks
14.	Easy XML Editor 1.6	●	✖	✖	No U.S. contact listed on Web site.
15.	EditiX 2010	✖	◖	✖	1. No U.S. contact listed on Web site. 2. Did not automatically validate the XML configuration vector when it was saved.
16.	epcEdit 1.2	●	✖	-	
17.	ExtJS 3.0	●	✖	-	
18.	EzXML Desktop 1.0	●	No information	-	Web site was down during the evaluation; hence, software was eliminated from further consideration.
19.	Java XML XPath Eclipse Templates 1.5	✖	✖	-	XPath templates.
20.	JCAM Engine with XML Editor / Validator 1.9	✖	✖	-	A text-based XML editor.

Product		Graphical XML Editor	XSD Validation	Availability of U.S. vendor	Remarks
21.	Jedit Editor 4.3	✖	✖	-	A text-based XML editor.
22.	JSimplex 3.1	✖	✖	-	A visual XSLT tool using the jConfig library (www.jconfig.org).
23.	Liquid XML Studio 2010	●	◖	●	1. U.S. sales line: +1 866 766 6374 2. Did not automatically validate the XML configuration vector when it was saved. 3. Software provided on-demand validation only.
24.	MoreMotion Application Studio 5	✖	✖	✖	No U.S. contact listed on Web site.
25.	Notepad++ with XML Tools 2.3	✖	◖	-	1. A text-based XML editor. 2. Did not automatically validate the XML configuration vector when it was saved.
26.	Open XML Editor 1.6	✖	✖	-	A text-based XML editor that supported DTD validation only.

Product		Graphical XML Editor	XSD Validation	Availability of U.S. vendor	Remarks
27.	<oXygen/> XML Editor 12.0	●	◖	●	1. U.S. line: +1 650 352 1250. Phone Orders US: 1 800 903 4152 2. Did not automatically validate the XML configuration vector when it was saved. 3. On-the-fly validation was available in text mode only.
28.	PTC Arbortext Editor 5.4	●	✗	●	1. U.S. Phone: 781 370 5000. U.S. Fax: 781 370 6000 2. Word processor like interface. 3. Software could not validate the keys and keyrefs defined in the configuration vector XML Schema.
29.	Quark XML Author 3.0	●	No information	●	1. E-mail: serviceplus@quark.com. U.S. Phone: 800 998 1716 2. Unable to verify if XSD validation with the sample configuration vector worked correctly since no evaluation copy was available for download. Eliminated from further consideration.

Product		Graphical XML Editor	XSD Validation	Availability of U.S. vendor	Remarks
30.	Qxmledit 0.4.9	●	◖	-	1. Tree-view. 2. Did not automatically validate the XML configuration vector when it was saved.
31.	Rinzo XML Editor 0.8	✖	●	-	A text-based XML editor.
32.	STDU XML Editor 1.0	✖	✖	No information	A text-based XML editor.
33.	Stylus Studio 2010	●	◖	●	1. U.S. Enterprise Sales Team: 305 748 4155 2. Did not automatically validate the XML configuration vector when it was saved.
34.	Syntext Serna XML Editor 4	●	✖	●	1. U.S. Sales & software licenses: sales@syntext.com 2. Did not automatically validate the XML configuration vector when it was saved. 3. Could not validate the *keys* and *keyrefs* defined in the configuration vector XML Schema.

Product		Graphical XML Editor	XSD Validation	Availability of U.S. vendor	Remarks
35.	Veloeclipse 2.07	✖	✖	-	A text-based XML editor.
36.	VEX 0.5 alpha	●	✖	-	Word processorlike interface.
37.	Xacobeo 0.13	✖	✖	-	GUI for constructing and executing XPath queries.
38.	Xcarecrows 4 XML 1.3.8	●	No information	-	1. Tree-view. Required understanding of XML as user interface used notions of “#attributes” and “#text.” 2. Could not start plug-in in Eclipse to verify that XSD validation worked as expected, and was eliminated from further consideration.
39.	XMetaL Author Enterprise 6.0	●	✖	✖	1. No U.S. contact listed on Web site. 2. Supported DTD only.
40.	XML Assistant 1.2	●	✖	✖	North America / Canada: +1 866 647 2003
41.	XML Code Editor 0.91	✖	✖	-	Software was still under development and not available for download.

Product		Graphical XML Editor	XSD Validation	Availability of U.S. vendor	Remarks
42.	XML Copy Editor 1.2	✖	●	-	A text-based XML editor.
43.	XML Editor 1.0.1	✖	✖	-	Could not load the sample configuration vector for editing in the table view.
44.	XML Editor Tool 6	✖	◖	-	Did not automatically validate the XML configuration vector when it was saved.
45.	XML Explorer 4.01	✖	●	-	An XML viewer.
46.	XML Manager 1.0	✖	✖	-	Only a German version of the software was available.
47.	XML Notepad 2007	●	●	-	1. Tree view 2. Provided XSD validation on-the-fly.
48.	XML Visualiser 2	✖	◖	-	Did not automatically validate the XML configuration vector when it was saved.
49.	XML Webpad 1.0	●	✖	-	Web-based framework in C# for creating custom editors.

Product		Graphical XML Editor	XSD Validation	Availability of U.S. vendor	Remarks
50.	XMLBlueprint XML Editor 7.5	✖	◖	✖	1. No U.S. contact listed on Web site. 2. Did not automatically validate the XML configuration vector when it was saved.
51.	XMLmind XML Editor 4.6	●	✖	✖	1. No U.S. contact listed on Web site. 2. Did not validate the sample configuration vector correctly – software reported XSD "cannot derive simpleType by restriction: minInclusive facet and maxExclusive facet are both specified."
52.	XMLSpear 3.10	●	●	-	1. Tree view 2. Provided XSD validation on-the-fly.
53.	XMLspark 0.5b	✖	✖	-	A text-based XML editor.
54.	XMLwriter 2.7	✖	◖	✖	1. No U.S. contact listed on Web site. 2. Did not automatically validate the XML configuration vector when it was saved.

Product		Graphical XML Editor	XSD Validation	Availability of U.S. vendor	Remarks
55.	XNGR XML Browser 2.0 beta	✖	✖	-	1. A text-based XML editor. 2. Did not validate the sample configuration vector correctly.
56.	Xopus 4.1.6	●	No information	✖	1. No U.S. contact listed on Web site. 2. Did not work without developing the XSLT style sheets and programming javascripts. Therefore, was unable to study them properly and get them to work with the sample configuration vector. Thus, it was eliminated from further consideration.
57.	xsl:easy 4.0	●	✖	✖	1. No U.S. contact listed on Web site. 2. Tree view. 3. Could not validate the *keys* and *keyrefs* defined in the configuration vector XML Schema.
58.	Ymacs	✖	✖	-	A text-based code editor.

Table 2. Additional information on the XML editors evaluated (This information is accurate as of September 2010)

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
1.	AjRa XML Editor 1.0	Windows	Standalone	-	Yes	GNU GPL	http://sourceforge.net/projects/ajraxmleditor/
2.	Altova XMLSpy 2010	Windows	Standalone	$499.00 per license (Professional Edition). $623.75 with 1 year support. Volume discounts available.	No	Proprietary	Altova - Austria http://www.altova.com

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
3.	Amaya 11	Windows, Unix, Mac OS X (browser)	Standalone	-	Yes	W3C	http://www.w3.org/Amaya/
4.	Angur XML Visualizer 1.01	Windows	Standalone	-	No	GNU GPL	http://sourceforge.net/projects/angur/
5.	Barium Visual XSL Transformer 1.5	Windows	Standalone	-	No	Freeware	http://www.dunnsolutions.co.uk/products/visualxsltransformer/barium.html
6.	Beacon Editor	No information	Standalone	-	No information	GNU GPL	http://beaconeditor.org
7.	CEDIT XML	Java-based platforms	Standalone	-	Yes	No information	http://sourceforge.net/projects/ceditxml/

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
8.	Civ4 Mod Editor	Windows	Standalone	-	No	No informatio n	http://sourceforge.net/projects/ c4me/
9.	Cladonia Exchanger XML Editor 3.3	Windows, Mac OS X, Linux/Unix	Standalone	No price listed on Web site.	No	Proprietary	Cladonia - Ireland http://www.exchangerxml.com http://code.google.com/p/exch angerxml
10.	codefunk XML Editor 1	Windows	Standalone	-	No	No informatio n	http://sourceforge.net/projects/ codefunkxmledit/
11.	Dataset Editor 0.1	Windows, Linux	Standalone	-	No	GNU GPL	http://sourceforge.net/projects/ dataseteditor/

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
12.	Debugging XSLT stylesheets using the Saxon processor 1.0	Multi-platform	Eclipse plug-in	-	Yes	GNU GPL	http://marketplace.eclipse.org/content/debugging-xslt-stylesheets-using-saxon-processor
13.	Dexter-XSL 0.3	Java-based platforms	Standalone	-	Yes	Artistic License / GPL	http://code.google.com/p/dexter-xsl/
14.	Easy XML Editor 1.6	Windows	Standalone	€20 per license Volume discounts available	No	Proprietary	Richard Würflein Software Development -Germany http://www-edit-xml.com

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
15.	EditiX 2010	Windows, Mac OS X, Linux	Standalone	$119 per license $149 with 1 year support.	No	Proprietary	JAPIsoft - Europe http://www.editix.com
16.	epcEdit 1.2	Windows, Linux, Solaris	Standalone	-	No	Freeware	http://www.epcedit.com
17.	ExtJS 3.0	Multi-platform, Web-based	Standalone	-	Yes	GNU GPL	http://code.google.com/p/extjs-xmleditor/
18.	EzXML Desktop 1.0	Multi-platform	Eclipse plug-in	-	No information	EPL	http://marketplace.eclipse.org/content/ezxml-desktop

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
19.	Java XML XPath Eclipse Templates 1.5	Multi-platform	-	-	Yes	GPL	http://marketplace.eclipse.org/content/java-xml-xpath-eclipse-templates
20.	JCAM Engine with XML Editor / Validator	Multi-platform	Standalone	-	Yes	OSL	http://sourceforge.net/projects/camprocessor
21.	Jedit Editor 4.3	Java-based platforms	Standalone	-	Yes	GNU GPL	http://www.jedit.org
22.	JSimplex 3.1	Windows, Mac OS, Linux	Standalone	-	Yes	GNU GPL	http://sourceforge.net/projects/jsimplex

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
23.	Liquid XML Studio 2010	Windows	Standalone	$320.56 per license (Designer Edition). $384.68 with 1 year support. Volume discounts available.	No	Proprietary	Liquid Technologies –U.K. http://www.liquid-technologies.com
24.	MoreMotion Application Studio 5	Windows	Standalone	Free for non-commercial use. No price listed on Web site.	No	Proprietary	Mor Yazilim Ltd - Turkey http://www.moremotion.com

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
25.	Notepad++ with XML Tools 2.3	Windows	Plug-in to Notepad++	-	Yes	GNU GPL	http://sourceforge.net/apps/mediawiki/notepad-plus/index.php?title=Plugin_Central
26.	Open XML Editor 1.6	Windows	Standalone	-	Yes	MPL / GPL	http://www.philo.de/xmledit/
27.	<oXygen/> XML Editor 12.0	Windows, Mac OS X, Linux/Unix	Standalone	$349 per license (Professional Edition). $422 with 1 year support. Volume discounts available.	No	Proprietary	Syncro Soft - Romania http://www.oxygenxml.com

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
28.	PTC Arbortext Editor 5.4	No information	Standalone	No information	No	Proprietary	PTC - U.S. http://www.ptc.com/products/arbortext/
29.	Quark XML Author 3.0	Windows	Plugin to Microsoft Word	No information	No	Proprietary	Quark – U.S. http://dynamicpublishing.quark.com/xml_author
30.	Qxmledit 0.4.9	Windows, Ubuntu	Standalone	-	Yes	GNU GPL	http://code.google.com/p/qxmledit/
31.	Rinzo XML Editor 0.8	Multi-platform	Eclipse plug-in	LGPL	No	No information	http://marketplace.eclipse.org/content/rinzo-xml-editor
32.	STDU XML Editor 1.0	Windows	Standalone	$9.95	No	Proprietary	STDUtility http://www.stdutility.com/stdu-xml-editor.html

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
33.	Stylus Studio 2010	Windows	Standalone	$450 (Professional edition)	No	Proprietary	Progress Software Corporation - U.S. http://www.stylusstudio.com
34.	Syntext Serna XML Editor 4	Windows, Linux, Mac OS X, Sun Solaris / SPARC	Standalone	Free and open-source (Home edition) $948 per license with 1 year support. (Enterprise edition) Volume discounts available.	No	Proprietary	Syntext Inc - U.S. http://www.syntext.com

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
35.	Veloeclipse 2.07	Multi-platform	Eclipse Plug-in	-	Yes	Free GPL	http://marketplace.eclipse.org/content/veloeclipse
36.	VEX 0.5 alpha	Multi-platform	Eclipse Plug-in	-	Yes	EPL	http://marketplace.eclipse.org/content/veloeclipse
37.	Xacobeo 0.13	Multi-platform	Standalone	-	Yes	Artistic License / GPL	http://code.google.com/p/xacobeo
38.	Xcarecrows 4 XML 1.3.8	Multi-platform	Eclipse Plug-in	-	Yes	Apache License / EPL	http://marketplace.eclipse.org/content/xcarecrows-4-xml
39.	XMetaL Author Enterprise 6.0	Windows	Standalone	No information	No	Proprietary	JustSystems - Japan http://na.justsystems.com

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
40.	XML Assistant 1.2	Windows	Standalone	€20 per license Volume discounts available	No	Proprietary	Richard Würflein Software Development -Germany http://www-edit-xml.com
41.	XML Code Editor 0.91 alpha	Windows	Microsoft Silverlight	-	Yes	MIT License	http://xmlcodeeditor.codeplex.com/releases/view/45990
42.	XML Copy Editor 1.2	Windows, Linux	Standalone	-	Yes	GNU GPL	http://xml-copy-editor.sourceforge.net/
43.	XML Editor 1.0.1	Windows	Standalone	-	No	Freeware	http://www.asaapplications.com/asa/
44.	XML Editor Tool 6	Windows	Standalone	-	No	Freeware	http://sourceforge.net/projects/xmleditortool/
45.	XML Explorer 4.01	Windows	Standalone	-	No	GNU GPL	http://xmlexplorer.codeplex.com

	Product	OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
46.	XML Manager 1.0	Windows	Standalone	-	No	GNU GPL	http://code.google.com/p/xmlmanager/
47.	XML Visualiser 2	Windows	Standalone / Visual Studio Plug-in	-	No	CDDL	http://xmlvisualizer.codeplex.com
48.	XML Notepad 2007	Windows	Standalone	-	No	Proprietary	Microsoft - U.S. http://www.microsoft.com
49.	XML Webpad 1.0	Multi-platform, Web-based	C# framework	-	No	MIT License	http://xmlwebpad.codeplex.com
50.	XMLBlueprint XML Editor 7.5	Windows	Standalone	$45 per license. Volume discounts available.	No	Proprietary	Monkfish XML Software – The Netherlands http://www.xmlblueprint.com/

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
51.	XMLmind XML Editor 4.6	Windows, Mac OS X, Linux	Standalone	Free (Personal Edition) $300 per license. Volume discounts available.	No	Proprietary	Pixware - France http://www.xmlmind.com
52.	XMLSpear 3.10	Windows, Linux, Mac OS	Standalone / Java web start	-	No	Freeware	http://www.donkeydevelopment.com/#downloads
53.	XMLspark 0.5b	Java-based platforms	Standalone	-	Yes	No information	http://sourceforge.net/projects/xmlvisualizer

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
54.	XMLwriter 2.7	Windows	Standalone	AUD139 Volume discounts available.	No	Proprietary	Wattle Software - Australia http://xmlwriter.net
55.	XNGR XML Browser	No information	Standalone	-	Yes	Mozilla Public License (MPL)	http://xngr.org
56.	Xopus 4.1.6	Windows, Mac OS X, Linux	Standalone Web-based	€180 per license Volume discounts available.	No	Proprietary	SDL - The Netherlands http://xopus.com
57.	xsl:easy 4.0	Windows	Standalone / Eclipse Plug-in	€298	No	Proprietary	SoftProject - Germany http://www.xsl-easy.com

Product		OS platform support	Type	Pricing (USD)	Open source	License type	Company HQ / Product Web site
58.	Ymacs	Multi-platform, Web-based	Standalone	-	Yes	BSD	http://www.ymacs.org/

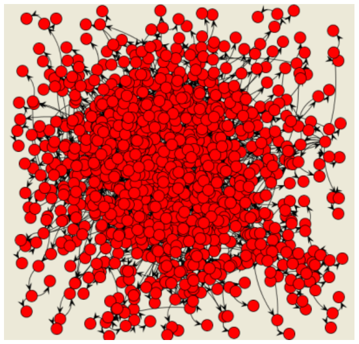


Figure 4. This display of the sample LPSK configuration vector using the Angur XML Visualizer illustrates its poor visualization quality

C. FEATURES COMPARISON

A good user interface helps to reduce the time needed to construct a configuration and to eliminate unnecessary human errors. Instead of assessing factors such as learnability and memorability of the software which are difficult to measure, GUI features that are considered to be useful for users to perform their task were identified. They were used to compare the seven (7) XML editors that were still in contention.

1. Nested Grid View

Visualizing the relationships between elements of the configuration vector is essential to the GUI. The previous version of the LPSK Configuration Vector tool adopted a tabbed interface to visualize partitions, processes, synchronization primitives and resources [7]. A nested grid view is another way to present the data. Figure 5 is an illustration of a nested grid view. The advantages of using a nested grid view are that the data is collated in a single view, and relationships between partitions, processes, subjects and subject-to-resource permissions are made explicit. It also allows the repetitive XML content (e.g., subjects and permissions) to be edited in a table-like fashion, similar to a spreadsheet application.

2. Source-Code View

Most graphical-based XML editors can show the source code for changes best made while looking at the XML source. This is very useful for users who are experienced with authoring XML. The editor should also color-code the syntax, so that it is easy for users to distinguish the code from the content. They should also include a tree view**.** This provides a hierarchical view of the document, which expands and collapses elements so that the user can focus only on the elements that need to be edited, making it easy to manage large configuration vectors.

3. Auto-Completion

Based on the associated XML Schema and location in the XML document, a list of possible elements (XML tags), attributes, or attribute values is shown whenever the user types an XML tag while in text mode, or creates a new row while in graphical mode. If the field has a pre-defined list of values defined by the schema enumeration constraint, a list of valid values is displayed. This is useful as a way of encouraging the correct structure and content, thus minimizing errors.

4. Embedded XML Schema Editor

An XML Schema editor is required to edit the configuration vector XML Schema. The advantage of using a schema editor is that it can be used to check that the configuration vector XML Schema is well formed and valid according to the W3C standards.

5. Error Handling

The requirement of error handling is to prompt the user if an error exists in the configuration vector. The editor should assist the user by directing him to the location in the configuration vector where the error occurs. This improves usability by making debugging and error recovery easier for users.

D. FINAL EVALUATION OUTCOME

Table 3 compares the editors based on their implementation of the features described above. The number of suitable editors was narrowed down from seven (7) to four (4) after reviewing their feature set. The finalists were XMLSpy, Stylus Studio, Liquid XML Studio and <oXygen/> XML Editor.

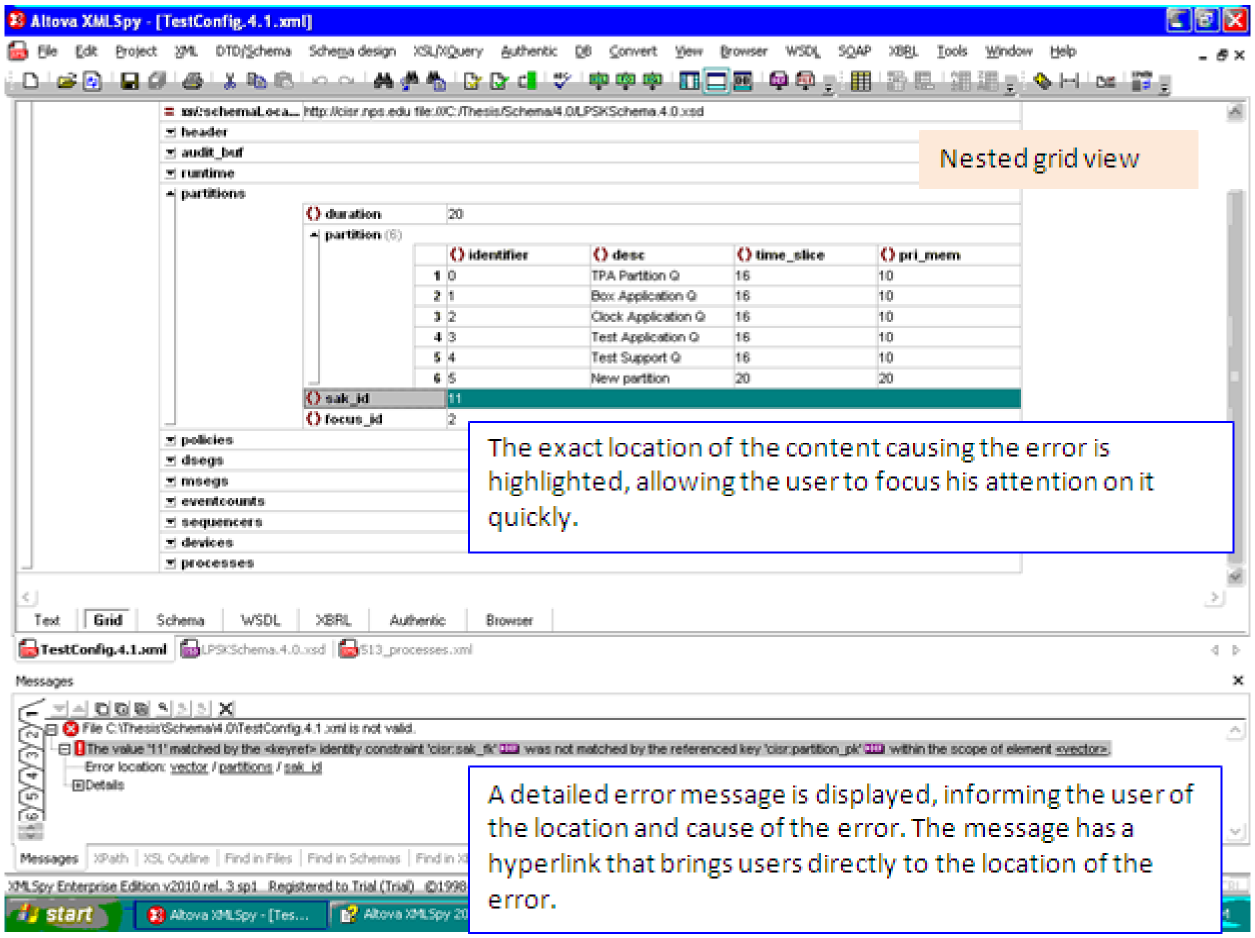
Table 3. Comparison of the XML editors by features (This information is accurate as of September 2010)

Product		**Nested grid view**	**Source code view**	**Auto-completion**	**Embedded XML Schema Editor**	**Error handling**	**Remarks**
1.	Altova XMLSpy 2010	●	●	●	●	●	
2.	Liquid XML Studio 2010	●	●	●	●	◖	Unable to highlight or direct user to the exact element that cause the error for referential constraints.
3.	<oXygen/> XML Editor 12.0	●	●	●	●	◖	Unable to highlight or direct user to the exact element that cause the error for referential constraints.
4.	Qxmledit 0.4.9	✗					Tree view

Product		Nested grid view	Source code view	Auto-completion	Embedded XML Schema Editor	Error handling	Remarks
5.	Stylus Studio 2010	●	●	◖	●	◖	1. Did not provide users with a list of valid values for fields limited to a set of acceptable values (enumeration constraint). 2. Did not highlight the element that caused the error.
6.	XML Notepad 2007	✗					Tree view
7.	XMLSpear 3.10	✗					Tree view

XMLSpy was the most expensive of the four remaining products. However, the error handling features in XMLSpy were superior compared to the other products. It highlighted the exact location of XML validation errors, while the mouse cursor was on another arbitrary location in the XML document. Efficiency is enhanced with XMLSpy since users do not have to spend time figuring out which content caused the error. In contrast, the other editors led the users the parent XML element where the defined constraint was violated, without pinpointing the exact child element that triggered the error. This kind of error can be hard to locate, especially if the parent element contains many child elements. This is illustrated by an example in Figures 5 to 8. In the LPSK configuration vector XML Schema, it is required that the partition that receives focus when the secure attention key (SAK) is invoked be an existing partition. This is defined by specifying a key on partitions *partition_p*k and a key reference *sak_fk* that references *partition_pk*, as shown in the code snippet in Figure 9. An error is flagged if *sak_fd* contains a partition ID that is not defined by the configuration vector. Since the constraint is defined inside the element *vector*, the editors only reported that a constraint had been violated in the context of element *vector*. This also applies to all other referential constraints in the XML Schema.

In conclusion, XMLSpy meets all the requirements for the GUI of the LPSK Configuration Vector Tool, and is recommended for use.


Altova XMLSpy - [TestConfig.4.1.xml]
File Edit Project XML DTD/Schema Schema design XSL/XQuery Authentic DB Convert View Browser WSDL SOAP XBRL Tools Window Help
Text Grid Schema WSDL XBRL Authentic Browser
TestConfig.4.1.xml LPSKSchema.4.0.xsd 513_processes.xml
Messages XPath XSL Outline Find in Files Find in Schemas
XMLSpy Enterprise Edition v2010 rel. 3 sp1 Registered to Trial (Trial) ©1998
start Altova XMLSpy - [Tes... Altova XMLSpy 20

xsi:schemaLoca... http://cisr.nps.edu file:///C:/Thesis/Schema/4.0/LPSKSchema.4.0.xsd

header

audit_buf

runtime

partitions

duration 20

partition (6)

	identifier	desc	time_slice	pri_mem
1	0	TPA Partition Q	16	10
2	1	Box Application Q	16	10
3	2	Clock Application Q	16	10
4	3	Test Application Q	16	10
5	4	Test Support Q	16	10
6	5	New partition	20	20

sak_id 11

focus_id 2

policies

dsegs

msegs

eventcounts

sequencers

devices

processes

Nested grid view

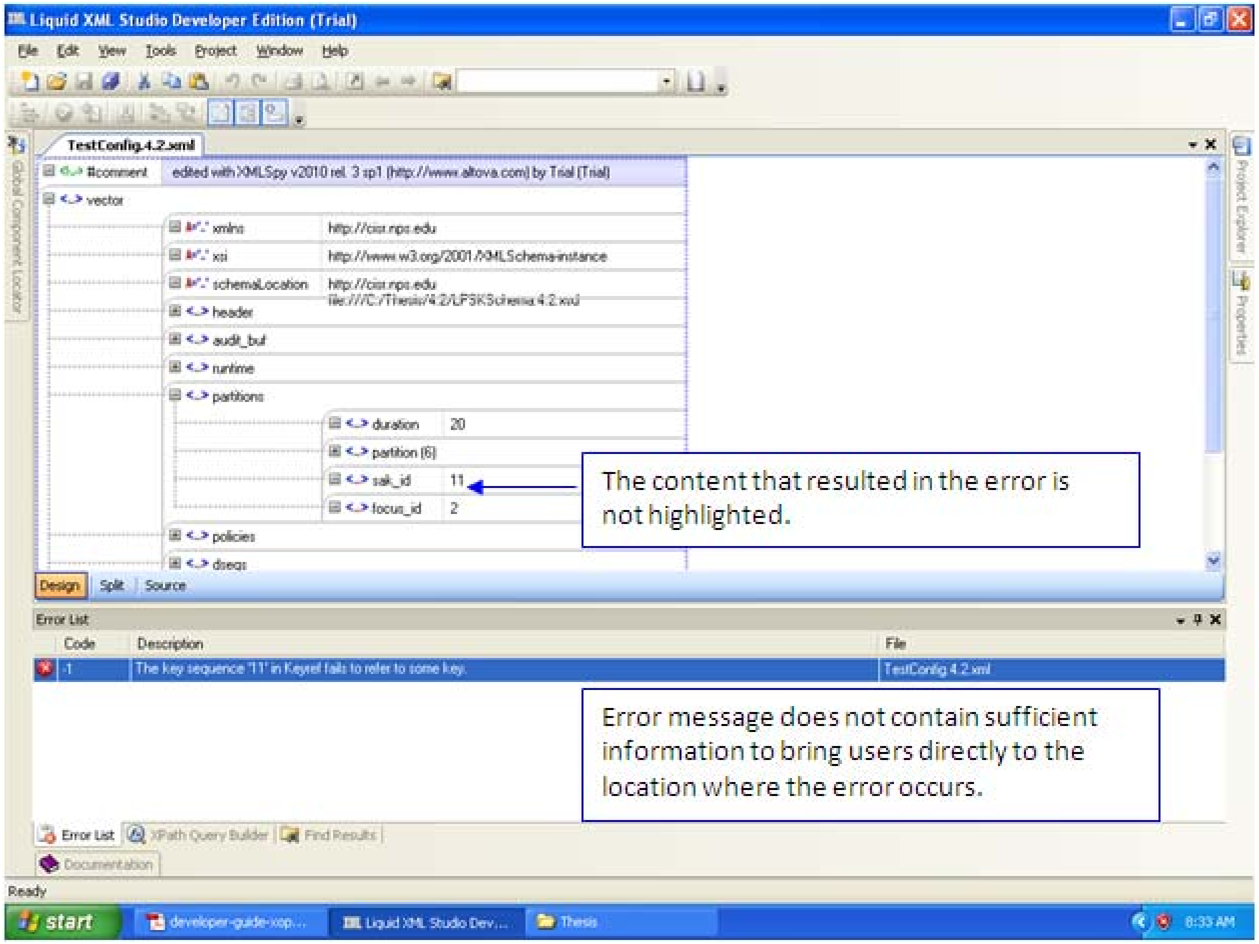
The exact location of the content causing the error is highlighted, allowing the user to focus his attention on it quickly.

```
File C:\Thesis\Schema\4.0\TestConfig.4.1.xml is not valid.
  The value '11' matched by the <keyref> identity constraint 'cisr:sak_fk' was not matched by the referenced key 'cisr:partition_pk' within the scope of element <vector>.
    Error location: vector / partitions / sak_id
    Details
```

A detailed error message is displayed, informing the user of the location and cause of the error. The message has a hyperlink that brings users directly to the location of the error.

Figure 5. Error handling in Altova XMLSpy, which also illustrates a nested grid view.

(The error message indicates that the error location is at *vector/partitions/sak_id*; the error is also highlighted.)


Liquid XML Studio Developer Edition (Trial)
File Edit View Tools Project Window Help
TestConfig.4.2.xml
Design Split Source
Error List
Error List XPath Query Builder Find Results
Documentation
Ready
start developer-guide-xop... Liquid XML Studio Dev... Thesis 8:33 AM

#comment	edited with XMLSpy v2010 rel. 3 sp1 (http://www.altova.com) by Trial (Trial)
vector	
xmlns	http://cisr.nps.edu
xsi	http://www.w3.org/2001/XMLSchema-instance
schemaLocation	http://cisr.nps.edu file:///C:/Thesis/4.2/LPSKSchema4.2.xsd
header	
audit_buf	
runtime	
partitions	
duration	20
partition (6)	
sak_id	11
focus_id	2
policies	
dseqs	

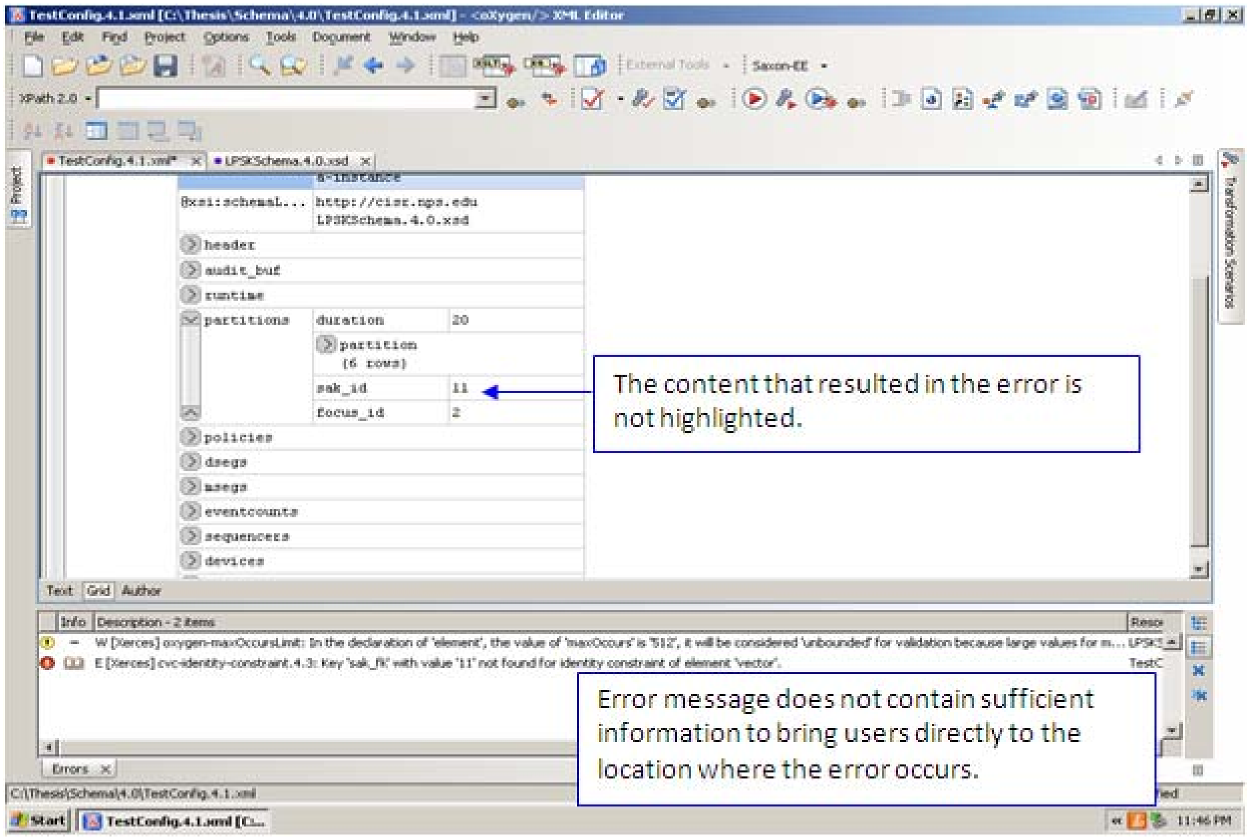
The content that resulted in the error is not highlighted.

Code	Description	File
-1	The key sequence '11' in Keyref fails to refer to some key.	TestConfig.4.2.xml

Error message does not contain sufficient information to bring users directly to the location where the error occurs.

Figure 6. Error handling in Liquid XML Studio.

(The error message indicates "the key sequence in Keyref fails to refer to some key", which is not intuitive.)



```
TestConfig.4.1.xml [C:\Thesis\Schema\4.0\TestConfig.4.1.xml] - <oXygen/> XML Editor

@xsi:schemaL...   http://cisr.nps.edu
                  LPSKSchema.4.0.xsd
header
audit_buf
runtime
partitions        duration        20
                  partition
                   (6 rows)
                  sak_id          11
                  focus_id        2
policies
dsegs
msegs
eventcounts
sequencers
devices
```

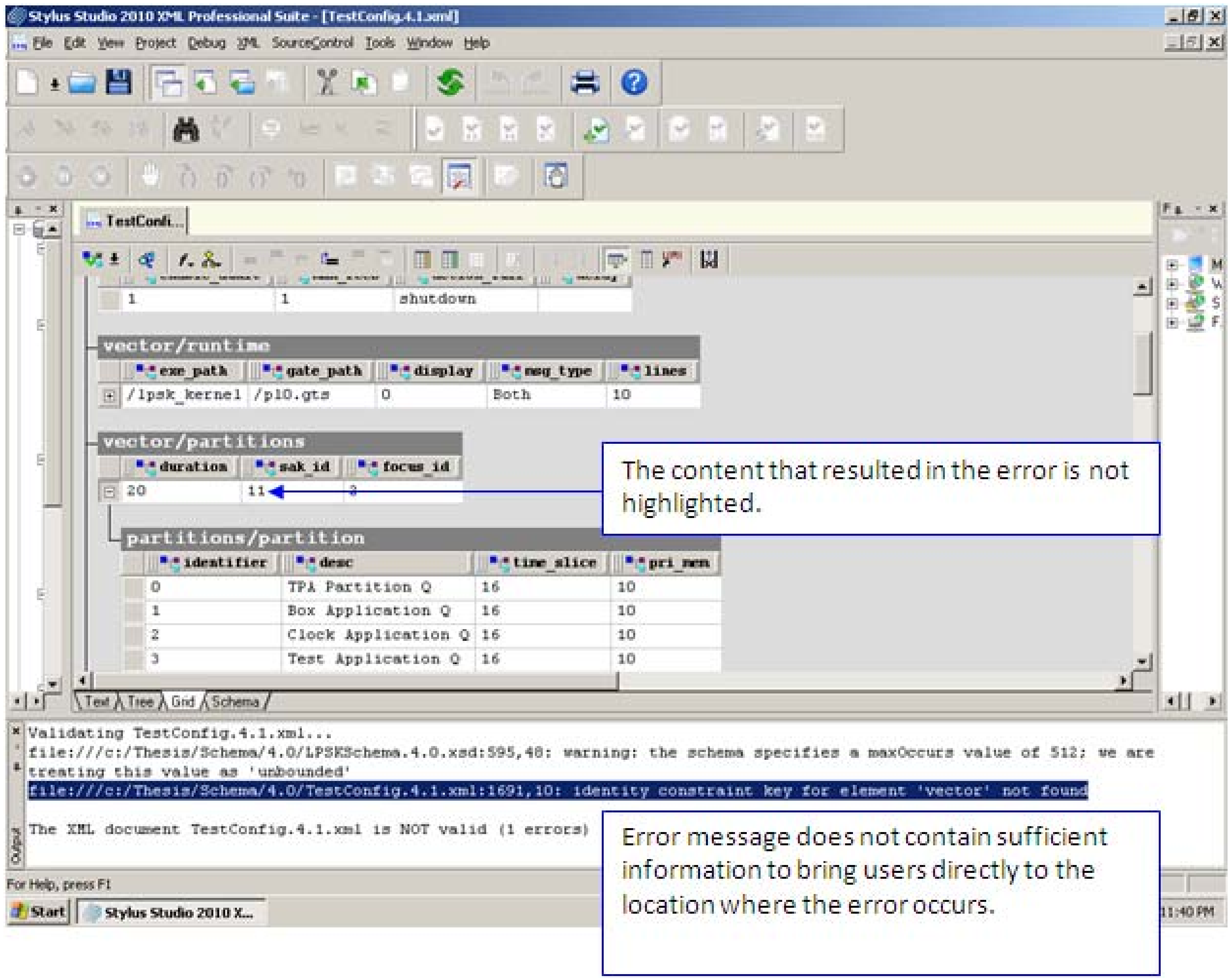
```
Info  Description - 2 items
W [Xerces] oxygen-maxOccursLimit: In the declaration of 'element', the value of 'maxOccurs' is '512', it will be considered 'unbounded' for validation because large values for m...  LPSK
E [Xerces] cvc-identity-constraint.4.3: Key 'sak_fk' with value '11' not found for identity constraint of element 'vector'.  TestC
```

The content that resulted in the error is not highlighted.

Error message does not contain sufficient information to bring users directly to the location where the error occurs.

Figure 7. Error handling in <oXygen/> XML Editor.

(The error message indicates the “sak_fk” constraint is violated for the identity constraint of element “vector.”)


Stylus Studio 2010 XML Professional Suite - [TestConfig.4.1.xml]
File Edit View Project Debug XML SourceControl Tools Window Help
TestConfi...
Text Tree Grid Schema
For Help, press F1
Start
Stylus Studio 2010 X...
11:40 PM

1	1	shutdown	

vector/runtime

exe_path	gate_path	display	msg_type	lines
/lpsk_kernel	/p10.gts	0	Both	10

vector/partitions

duration	sak_id	focus_id
20	11	3

partitions/partition

identifier	desc	time_slice	pri_mem
0	TPA Partition Q	16	10
1	Box Application Q	16	10
2	Clock Application Q	16	10
3	Test Application Q	16	10

The content that resulted in the error is not highlighted.

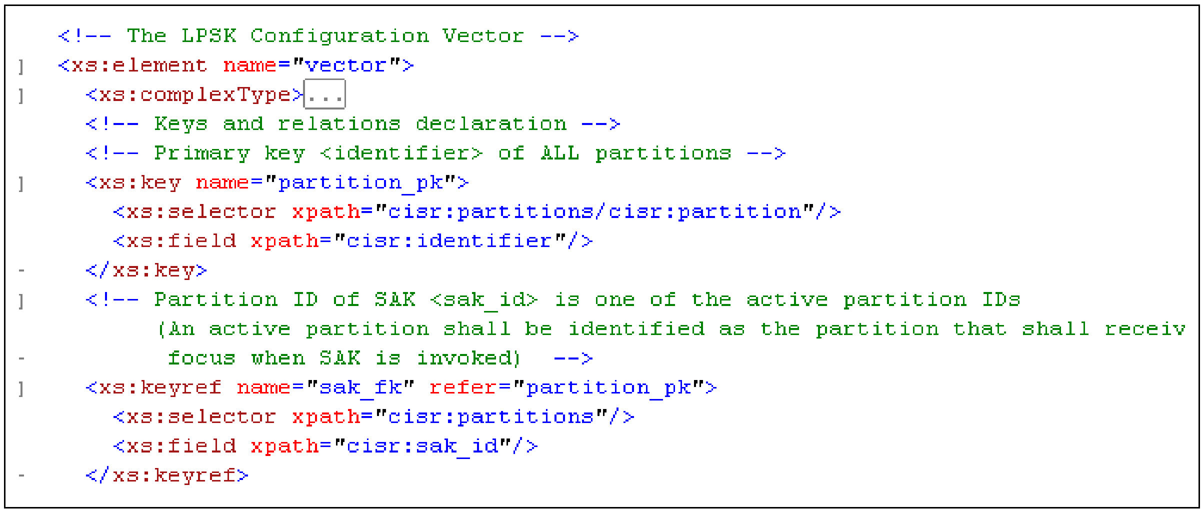
```
Validating TestConfig.4.1.xml...
file:///c:/Thesis/Schema/4.0/LPSKSchema.4.0.xsd:595,48: warning: the schema specifies a maxOccurs value of 512; we are
treating this value as 'unbounded'
file:///c:/Thesis/Schema/4.0/TestConfig.4.1.xml:1691,10: identity constraint key for element 'vector' not found

The XML document TestConfig.4.1.xml is NOT valid (1 errors)
```

Error message does not contain sufficient information to bring users directly to the location where the error occurs.

Figure 8. Error handling in Stylus Studio.

(The error message indicates the identity constraint key for element ‘vector’ is not found, which is not intuitive.)



```
   <!-- The LPSK Configuration Vector -->
]  <xs:element name="vector">
]    <xs:complexType>...
     <!-- Keys and relations declaration -->
     <!-- Primary key <identifier> of ALL partitions -->
]    <xs:key name="partition_pk">
       <xs:selector xpath="cisr:partitions/cisr:partition"/>
       <xs:field xpath="cisr:identifier"/>
-    </xs:key>
]    <!-- Partition ID of SAK <sak_id> is one of the active partition IDs
          (An active partition shall be identified as the partition that shall receiv
-          focus when SAK is invoked)  -->
]    <xs:keyref name="sak_fk" refer="partition_pk">
       <xs:selector xpath="cisr:partitions"/>
       <xs:field xpath="cisr:sak_id"/>
-    </xs:keyref>
```

Figure 9. Code snippet of the SAK partition constraint, defined in the context of a configuration vector.

E. SUMMARY

This chapter outlined the selection process for suitable XML editors available on the market for the LPSK Configuration Vector Tool. Altova XMLSpy was found to be the most suitable software that meets the requirements for the GUI. The next chapter describes the design and implementation of the LPSK Configuration Vector XML Schema and the other components of the LPSK Configuration Vector Tool.

THIS PAGE INTENTIONALLY LEFT BLANK

V. DESIGN AND IMPLEMENTATION

The LPSK Configuration Vector Tool is conceived to be a suite of utilities to support the creation of LPSK configuration vectors. The previous chapter discussed the selection of a GUI-based editor to facilitate the editing of XML-based configuration vectors. This chapter describes the design considerations and implementation of other components in the tool.

A. CONFIGURATION VECTOR FORMAT

This section describes the design and implementation of the LPSK Configuration Vector XML and binary formats.

1. XML Representation

The benefits of defining a XML Schema for the LPSK Configuration Vector were discussed in Chapter II. XML Schema Definition Language (XSD) 1.0 [10] was used to implement the schema. XSD 1.0 has excellent tool support. The rationale for choosing XSD over Document Type Definition (DTD) is that it has a richer grammar that allows precise definition of the number of occurrences of elements, provides a way to describe the relationship between elements, and contains a large number of data types for describing elements.

User-defined data types were extensively used to restrict the values of some fields to their acceptable values or ranges. The configuration vector XML Schema user-defined data types are summarized in Table 4. It also facilitates re-use in other elements. For example, *path_type* is defined once, and re-used in several elements.

Table 4. User-defined data types defined in the LPSK Configuration Vector XML Schema

User-defined Data Types	Description	Definition / Valid values
`action_type`	The set of actions the LPSK can take when the audit buffer is full.	overwrite, halt, shutdown
`bool_type`	The Boolean data type.	0 or 1
`buf_type`	The data type for the audit buffer size in bytes.	An integer between 1 and 65535.
`data_or_control_type`	The set of values for device types.	data, control
`delay_type`	The data type for the delay in seconds before LPSK shuts down when the audit buffer is full.	An integer between 0 and 60.
`desc_type`	The data type for *description* field.	A string of length between 0 and 32 characters.
`device_attr_type`	The set of device attributes.	dedicated, multiplexed
`device_event_type`	The set of events to audit for devices.	read data, write data, read metadata, write metadata

User-defined Data Types	**Description**	**Definition / Valid values**
`dseg_event_type`	The set of events to audit for data segments.	swapin, flush, swapout
`event_type`	The set of audit event status.	success, failure, both
`eventcounts_event_type`	The set of events to audit for eventcounts.	advance, read, await, wakeup
`kb_buf_size_type`	The data type for the keyboard incoming scan code buffer size.	An integer between 32 and 1024.
`lpskmsg_type`	The set of LPSK message types.	Status, Partition with focus, Both
`mac_addr_type`	The data type for the network device MAC address.	A string with format XX:XX:XX:XX:XX:XX, where X is a hexadecimal digit between 0x0 and 0xF.
`mandatory_path_type`	The data type for a mandatory path.	A string of length between 1 and 64 characters.
`mseg_event_type`	The set of events to audit for memory segments.	created
`network_buf_size_type`	The data type for each network device's incoming and outgoing buffer size.	An integer between 32 and 256.

User-defined Data Types	Description	Definition / Valid values
`network_read_type`	The network device read behavior.	promiscuous, non-promiscuous
`network_state_type`	The set of network device states.	started, stopped
`nillable_part_id_type`	The range of values that valid partition *identifiers* can take. Null values are allowed.	An integer between 0 and 256 or an empty string.
`part_id_type`	The range of values that valid partition *identifiers* can take.	An integer between 0 and 256.
`path_type`	The data type for a path. Null values are allowed.	A string of length between 0 and 64 characters.
`percent_type`	The range of values for percentages.	A numerical value between 0 and 100.
`perm_type`	A set of permissions.	NA, RO, RW, WO
`pl_type`	The range of values for privilege levels.	An integer between 0 and 3.
`positive_percent_type`	The range of values for percentage greater than 0.	A numerical value greater than 0, and up to a maximum of 100.
`proc_id_type`	The range of values that process *identifiers* can take.	An integer between 0 and 511.
`ro_perm_type`	A set of permissions.	NA, RO

User-defined Data Types	**Description**	**Definition / Valid values**
`rw_perm_type`	A set of permissions.	NA, RW
`screen_line_type`	Data type for the number of video character lines.	An integer with minimum value 1.
`sequencers_event_type`	The set of events to audit for sequencers.	ticket
`start_state_event_type`	The set of start states for data segments.	swapin, swapout
`subj_event_type`	The set of events to audit for subjects.	interrupt, received signal, send signal, read device, write device, read eventcount, advance eventcount, wait eventcount, wakeup eventcount, ticket sequencer
`sync_id_type`	The range of values that eventcount *identifiers* and sequencer *identifiers* can take.	An integer between 0 and 63.

The LPSK configuration vector XML Schema is illustrated in Figures 10 to 14, where *vector* is the root element.

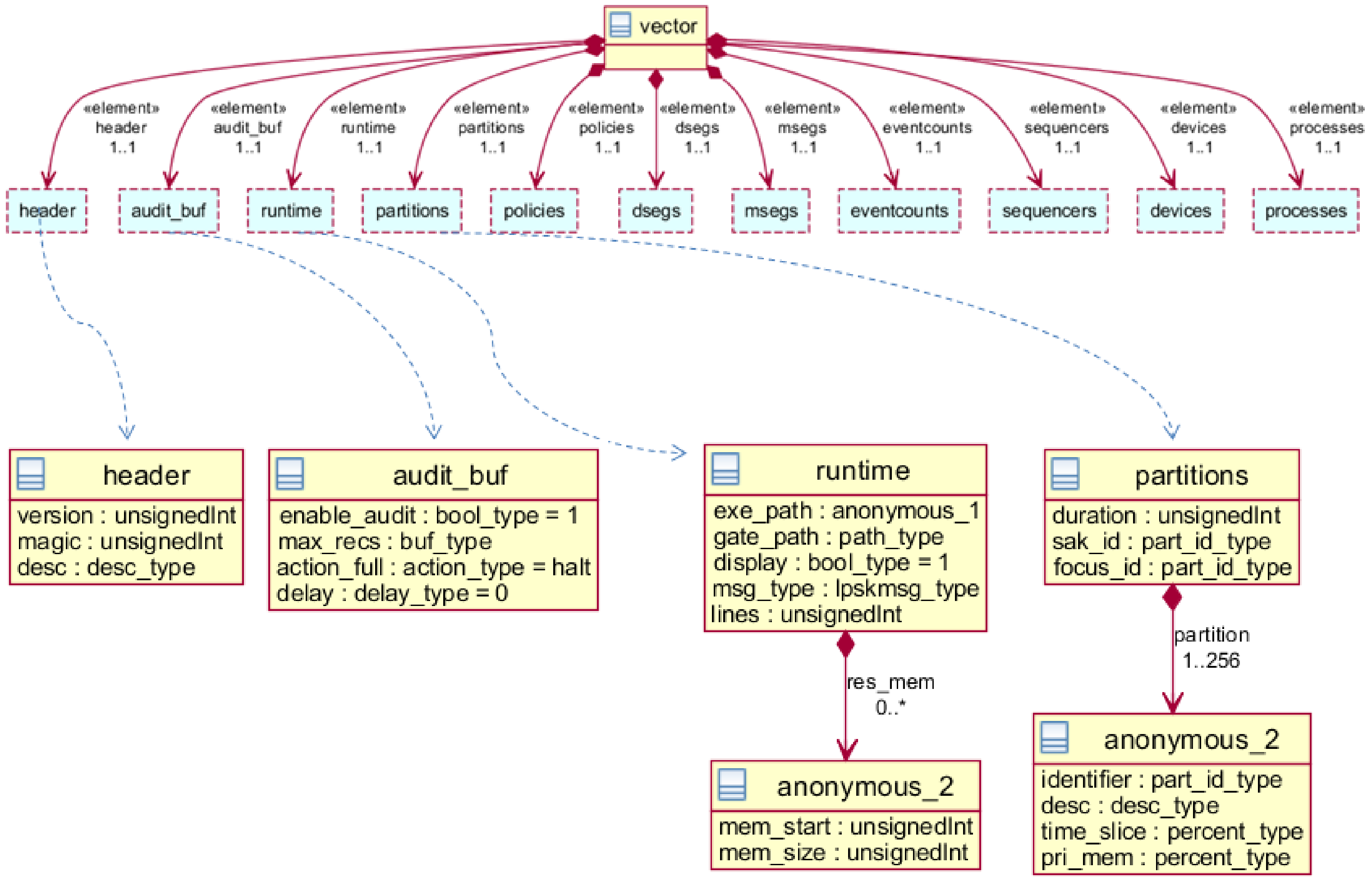


Figure 10. XML Schema with *header*, *audit_buf* (audit buffer configuration), *runtime* and *partitions* details.

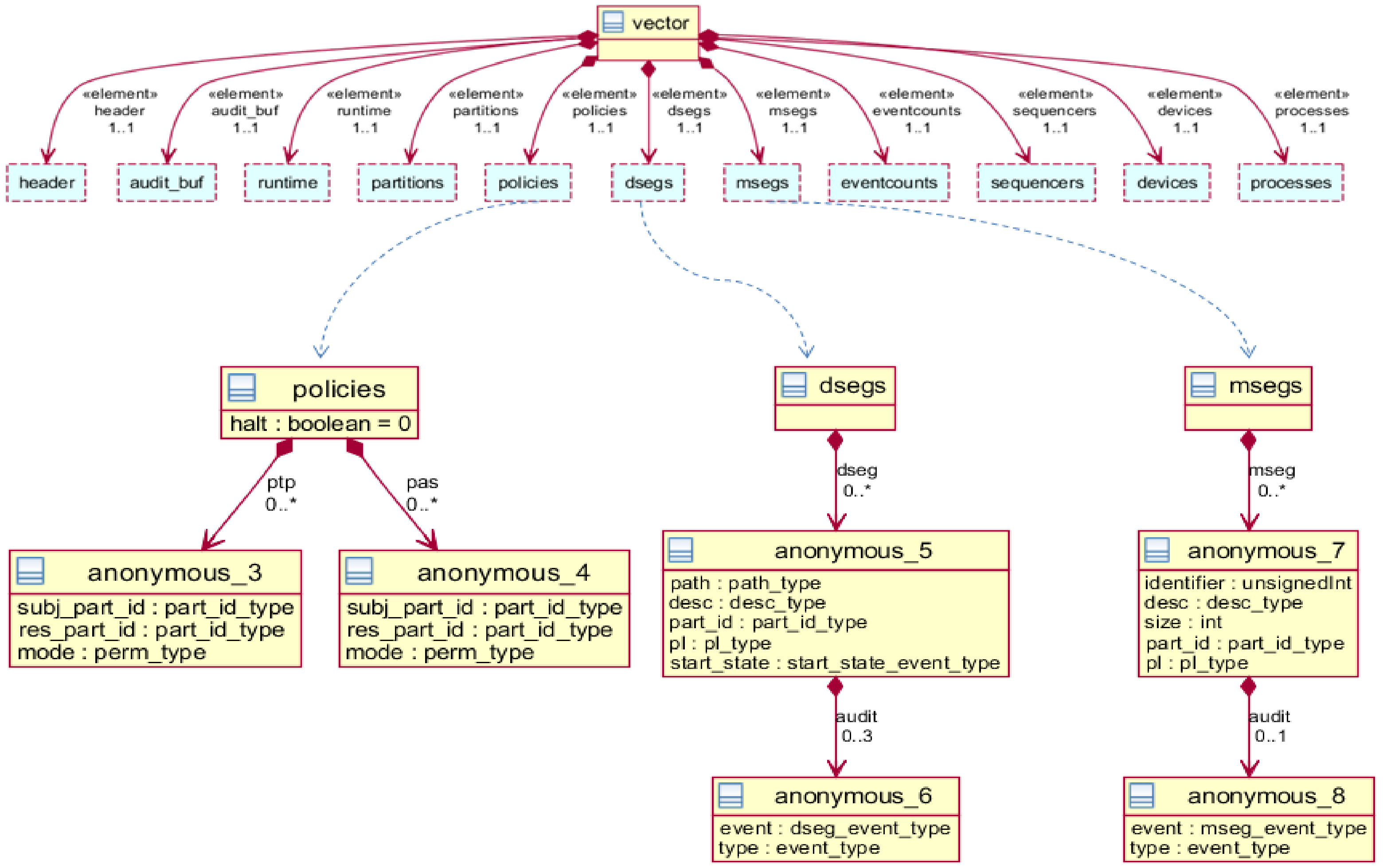


Figure 11. XML Schema with *policy*, *dseg* (data segment) and *mseg* (memory segment) details.

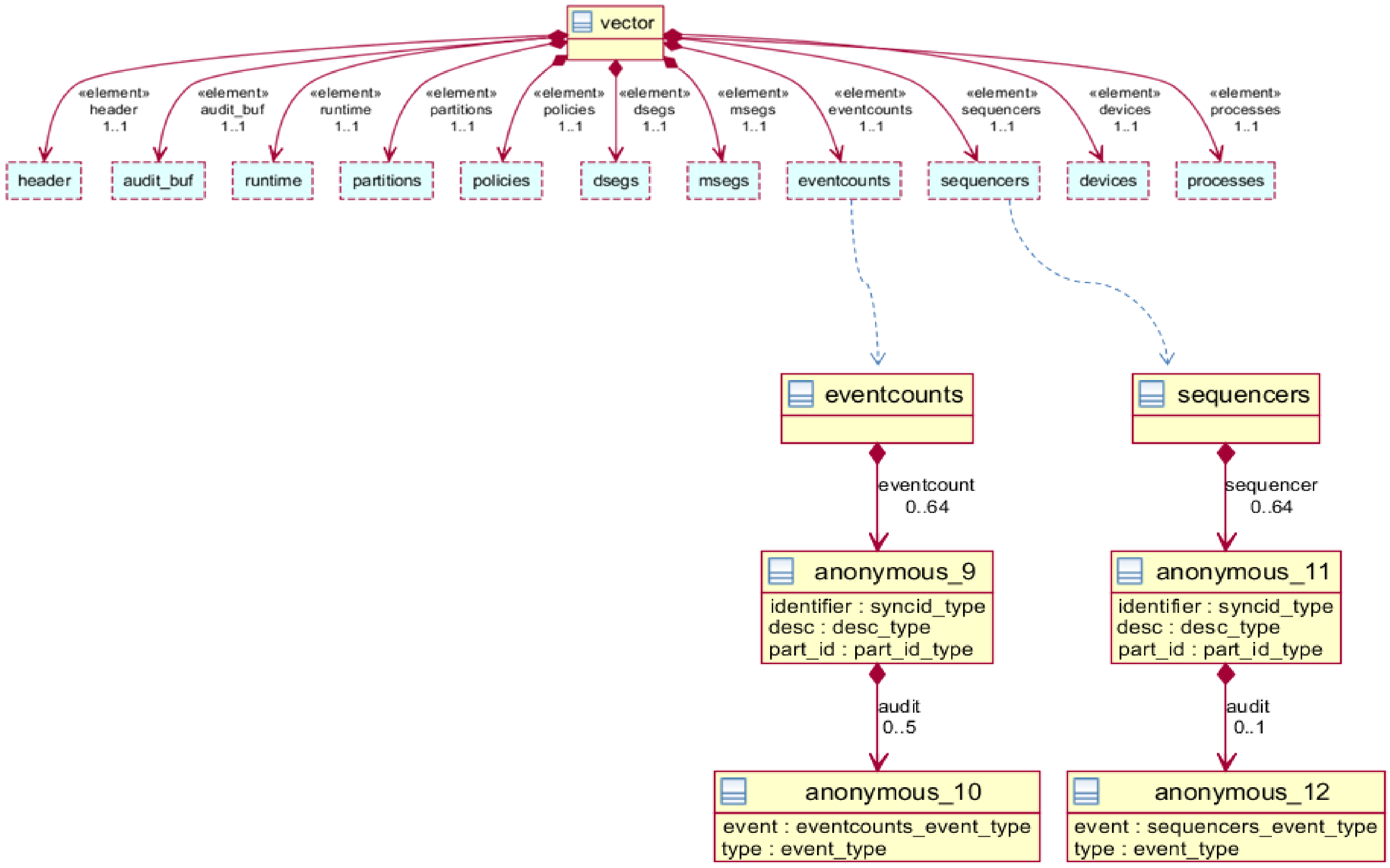


Figure 12. XML Schema with *eventcount* and *sequencer* details.

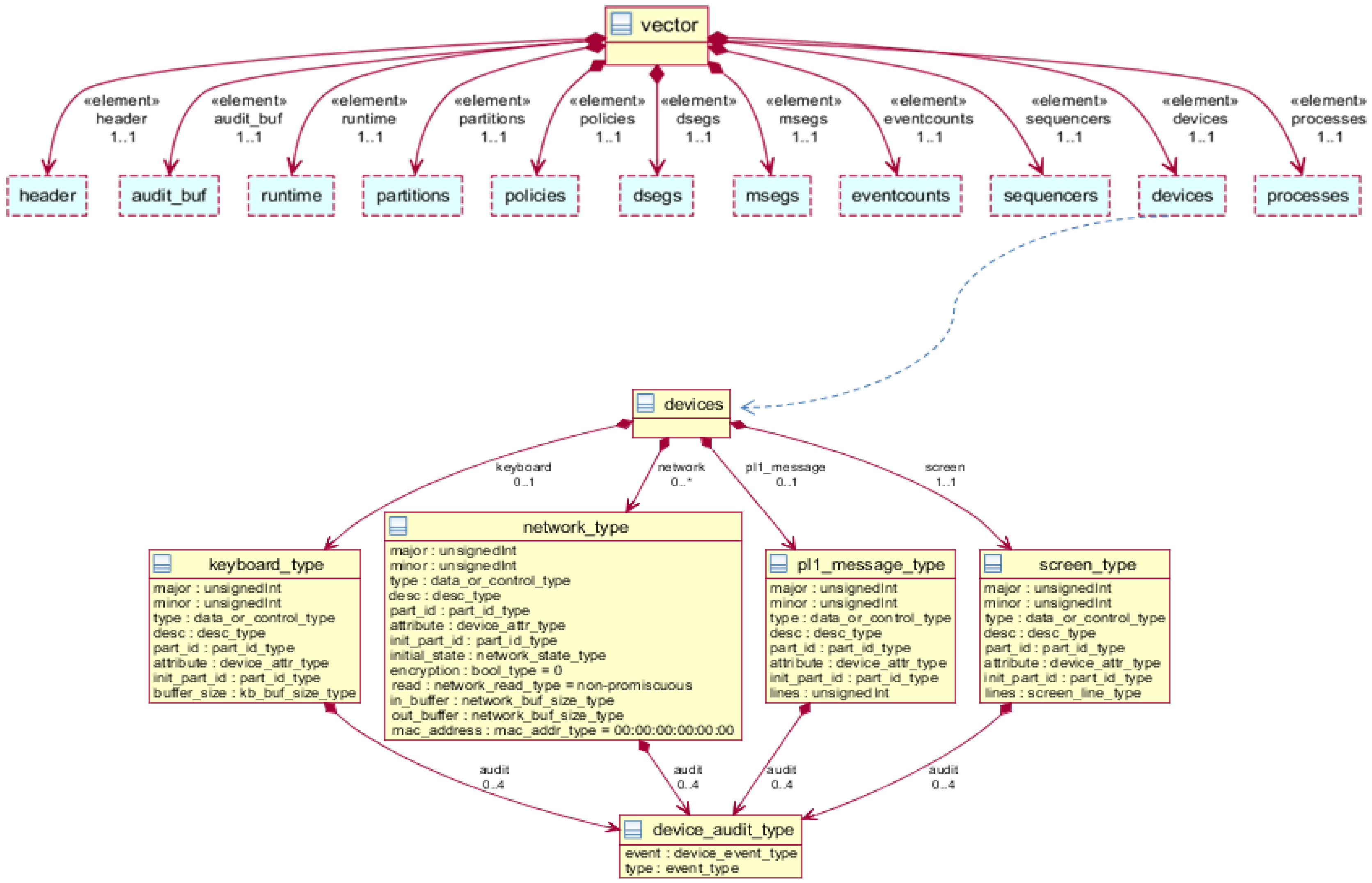


Figure 13. XML Schema with *device* details.

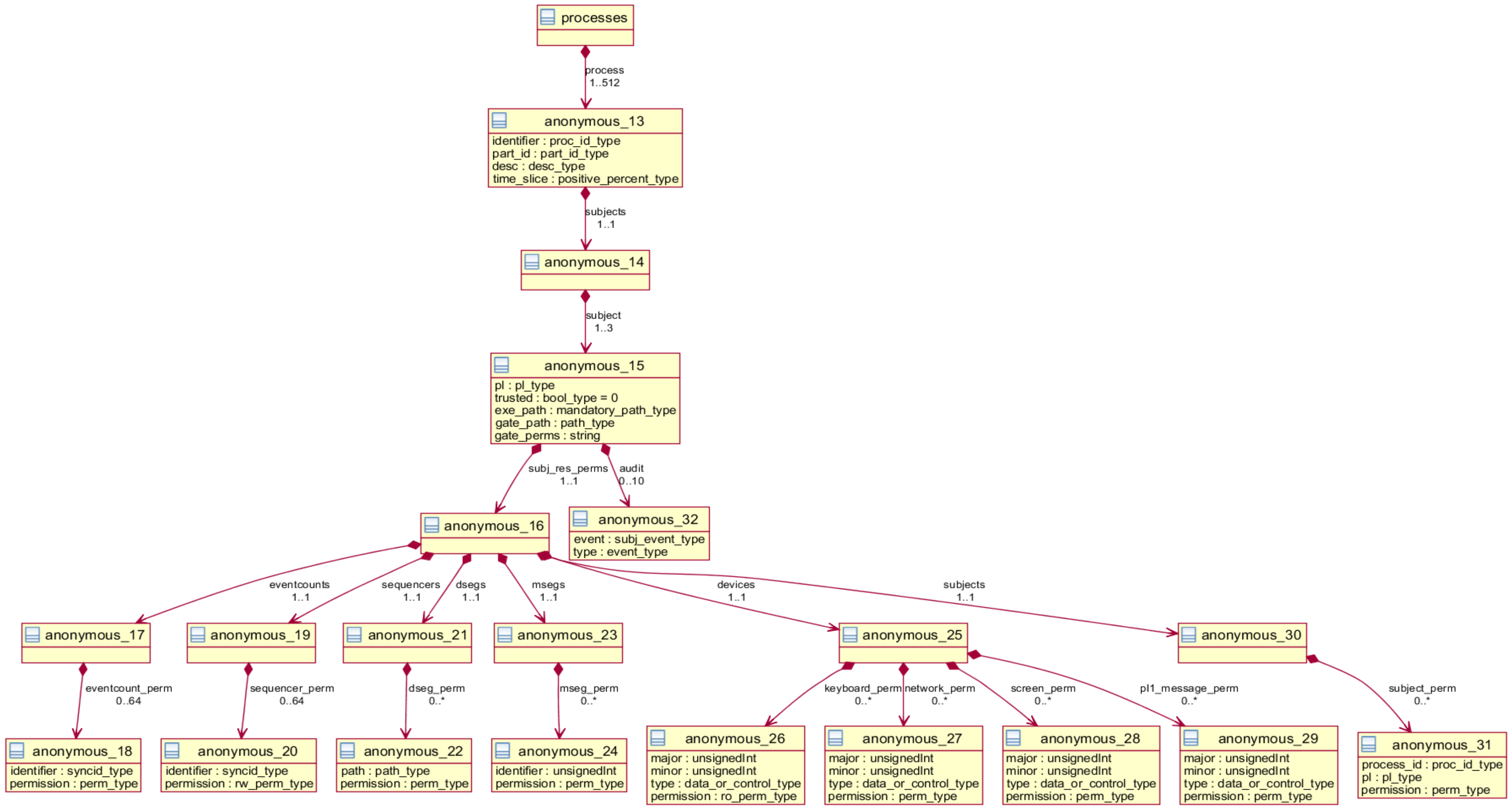


Figure 14. XML Schema with *process*, *subject* and *subj_res_perms* (subject-to-resource permission) details

The LPSK XML Schema uses keys and key references (*keyref*) in XSD 1.0 to define referential constraints between elements in the XML configuration vector, as shown in Table 5. This is similar to the primary-key and foreign-key feature available in relational database systems. These are used by the Configuration Vector Tool to check and maintain the referential integrity of partitions, subjects and resources in a configuration vector.

Table 5. Referential constraints defined in the LPSK XML Schema

***key* Name**	***keyref* Name**	**Description**
`partition_pk`	`ptp_subj_part_id_fk`	The subject partition in a PTP policy is defined in *partitions*.
	`ptp_res_part_id_fk`	The resource partition in a PTP policy is defined in *partitions*.
	`pas_subj_part_id_fk`	The subject partition in a PAS policy is defined in *partitions*.
	`pas_res_part_id_fk`	The resource partition in a PAS policy is defined in *partitions*.
	`dseg_partition_fk`	The home partition of a data segment is defined in *partitions*.
	`mseg_partition_fk`	The home partition of a memory segment is defined in *partitions*.
	`eventcount_partition_fk`	The home partition of an eventcount is defined in *partitions*.
	`sequencer_partition_fk`	The home partition of a sequencer is defined in *partitions*.

***key* Name**	***keyref* Name**	**Description**
	`device_partition_fk`	The home partition of a device is defined in *partitions*.
	`device_initpartition_fk`	The partition of a dedicated device with initial access is defined in *partitions*.
	`process_partition_fk`	The home partition of a process is defined in *partitions*.
`eventcount_pk`	`subj_eventcount_fk`	The eventcount a subject is allowed to access is defined in *eventcounts*.
`sequencer_pk`	`subj_sequencer_fk`	The sequencer a subject is allowed to access (subject-to-resource permissions) is defined in *sequencers*.
`dseg_pk`	`subj_dseg_fk`	The data segment a subject is allowed to access (subject-to-resource permissions) is defined in *dsegs*.
`mseg_pk`	`subj_mseg_fk`	The memory segment a subject is allowed to access (subject-to-resource permissions) is defined in *msegs*.
`device_pk`	`subj_device_fk`	The device a subject is allowed to access (subject-to-resource permissions) is defined in *devices*.

Uniqueness constraints are defined to enforce uniqueness of a value or combinations of values in an element, as shown in Table 6. With respect to subjects-to-resource permissions, the constraints prevent multiple declarations of the same resource for a given subject, but allow the same resource to be accessed by multiple subjects.

Table 6. Uniqueness constraints defined in the LPSK XML Schema

Unique Name	**Description**
`ptp_unique`	The combination of subject partition (*subj_part_id*) and resource partition (*res_part_id*) is unique across PTP policies.
`pas_unique`	The combination of subject partition (*subj_part_id*) and resource partition (*res_part_id*) is unique across PAS policies.
`subj_unique`	The privilege level (*pl*) of a subject is unique across subjects in a process.
`subj_eventcount_unique`	The eventcount a subject is allowed to access (subject-to-resource permissions) is unique within that subject.
`subj_sequencer_unique`	The sequencer a subject is allowed to access (subject-to-resource permissions) is unique within that subject.
`subj_dseg_unique`	The data segment a subject is allowed to access (subject-to-resource permissions) is unique within that subject.
`subj_mseg_unique`	The memory segment a subject is allowed to access (subject-to-resource permissions) is unique within that subject.
`subj_device_unique`	The device a subject is allowed to access (subject-to-resource permissions) is unique within that subject.

Unique Name	Description
`subj_subj_unique`	The subject another subject is allowed to access (subject-to-resource permissions) is unique within that subject.

XSD 1.0 is unable to express some of the validation rules (constraints) required by the LPSK Configuration Vector Tool; additional Java code was written to enforce these rules.

- The total CPU time allocated to all partitions must be exactly 100%.
- A process must belong to an active partition.
- The secure attention key (SAK) partition must be defined when a keyboard device is configured, and it must be an active partition; it cannot be defined when no keyboard device is configured.
- The partition that receives initial KVM focus must be defined when a screen device is configured, and it must be an active partition; it cannot be defined in terms of KVM when no screen device is configured.
- A process must be allocated more than 0% of the CPU time slice.
- The total time slices allocated to all processes in a given partition must be equal to 100% of the CPU time slices allocated to that partition.

2. Binary Representation

The binary representation of the LPSK configuration closely follows the structure of the XML representation, in terms of the ordering of the fields. To support dynamic re-sizing of the file, additional fields are introduced as follows. This allows the configuration tool to determine how many records of each type would follow while it is parsing the binary data.

- Number of reserved memory regions in the Run-time LPSK
- Number of partitions
- Number of PTP policies

- Number of PAS policies
- Number of data segments, and number of audit events for each segment
- Number of memory segments, and number of audit events for each segment
- Number of eventcounts, and number of audit events for each eventcount
- Number of sequencers, and number of audit events for each sequencer
- Number of devices, and number of audit events for each type
- Number of processes
- Number of subjects in each process
- Number of subject-to-resource permissions for each subject
- Number of audit events for each subject

For enumerated fields, the values were encoded as integers as shown in Figure 15 for efficiency reasons.

```
/* Actions to take when audit is full. Used by audit_buf_struct.action_full. */
#define OVERWRITE      0  // Overwrite
#define HALT             1        // Halt the platform
#define SHUTDOWN       2  // Shutdown the platform

/* Types of messages to display. Used by runtime_struct.msg_type. */
#define STATUS           0        // Status
#define PART_FOCUS       1        // Partition with focus
#define BOTH_MSG         2        // Both

/* Permissions and Policies access modes. Used by policy_struct.mode.  */
#define NA                       0        // No Access
#define RO                       1        // Read only
#define RW                       2        // Read and Write
#define WO                       3        // Write only

/* Audit events. Used by audit_struct */
/* Dsegs */
#define DS_SWAPIN        0
#define DS_FLUSH         1
#define DS_SWAPOUT       2
/* Msegs */
#define CREATED          3
/* Eventcounts */
#define ADVANCE          4
#define READ             5
#define AWAIT            6
#define WAKEUP           7
/* Sequencers */
#define TICKET           8
/* Devices */
#define READ_DATA        9
#define WRITE_DATA       10
#define READ_META        11       // read metadata
#define WRITE_META       12       // write metadata
/* Subjects */
#define INTERRUPT        13       // interrupt
#define RCV_SIGNAL       14       // received signal
#define SND_SIGNAL       15       // send signal
#define READ_DEV         16       // read device
#define WRITE_DEV        17       // write device
#define READ_EVCT        18       // read eventcount
#define ADV_EVCT         19       // advance eventcount
#define WAIT_EVCT        20       // wait on eventcount
#define WAKE_EVCT        21       // wakeup on eventcount
#define TICKET_SEQ       22       // ticket sequencer
/* Audit success, failure or both */
#define SUCCESS          23       // success
#define FAILURE          24       // failure
#define BOTH             25       // both

/* Start state. Used by desg_struct.start_state. */
#define SWAPIN_STATE     0
#define SWAPOUT_STATE    1
/* Device data or control type. Used by device_struct.type. */
#define DATA             0
#define CONTROL          1
/* Device attribute. Used by device_struct.attribute. */
#define MULTIPLEXED      0
#define DEDICATED        1
/* Network device initial state */
#define STARTED        0
#define STOPPED          1
/* Network device read */
#define PROMISC          0        // promiscuous mode
#define NON_PROMISC      1        // non-promiscuous mode
```

Figure 15. Binary representation of values of enumerated fields

All Boolean and numerical fields are assigned the *int* data type, although some fields require only *byte* or *short* representations. The rationale for doing so was to avoid the byte alignment issue when using mixed content types in *C struct*, as the configuration vector is processed in the LPSK using *C*.

Since different device categories have different fields, each device declaration in the binary had to be pre-pended with an integer to indicate the type of device that follows, where *0* indicates a keyboard device, *1* indicates a network device, and *2 i*ndicates a PL1 message device or screen device.

In addition, a set of interfaces was implemented to facilitate the traversing of binary configuration vectors by the LPSK, as shown in Table 7. This provides a consistent interface for different programs in the LPSK to access different portions of a configuration vector, and minimizes the impact of future changes to the configuration vector format.

Table 7. Interfaces provided to the LPSK for traversing configuration vectors

Interface	**Description**
`void* get_header(void* ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *header*.
`void* get_audit_buf(void* ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *audit buffer configuration*.
`void* get_runtime(void* ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *runtime*.
`void* get_partitions(void* ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *partitions*.

Interface	**Description**
`void* get_policies(void* ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *policies*.
`void* get_dsegs(void* ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *dsegs*.
`void* get_msegs(void *ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *msegs*.
`void* get_eventcounts(void *ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *eventcounts*.
`void* get_sequencers(void *ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *sequencers*.
`void* get_devices(void *ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *devices*.
`void* get_processes(void *ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *processes*.
`void* get_subjects(void *ptr)`	Takes a reference to the configuration vector in memory and returns a pointer to the start of *subjects*.

B. CONFIGURATION VECTOR TOOL UTILITIES

1. Format Conversion

A command-line format conversion utility was developed for conversion of configuration vectors from its XML form into its binary form, and vice-versa. The syntax of the utility is shown in Figure 16.

```
java -jar vector.jar -bin|-xml|-h [input file] [output file] [schema
file]
      Options:
         -bin     Converts input to binary format.
         -xml     Converts input to XML format.
         -h       Displays help.
      Parameters:
         input file      The input file name.
         output file     The output file name.
         schema file     The XML Schema file name.
```

Figure 16. Format conversion command-line interface

A set of Java classes was developed to manage the conversion of different elements within a configuration vector, as shown in Table 8. The rationale for choosing Java is that the code can run on a wide variety of commodity operating systems. There is also a set of readily available Java-based libraries for parsing and validating XML documents.

Table 8. Java classes of the format conversion utility

Java Class Name	**Description**
`AuditBuffer`	Implements the conversion of the *audit buffer* portion of a configuration vector from XML format to binary format, and vice versa.
`Audits`	Implements the conversion of *audit events* portion of a configuration vector from XML format to binary format, and vice versa.
`Devices`	Implements the conversion of *devices* portion of a configuration vector from XML format to binary format, and vice versa.
`Dsegs`	Implements the conversion of *data segments* portion of a configuration vector from XML format to binary format, and vice versa.
`Eventcounts`	Implements the conversion of *eventcounts* portion of a configuration vector from XML format to binary format, and vice versa.
`Header`	Implements the conversion of *version, magic number* and *description* portion of a configuration vector from XML format to binary format, and vice versa.
`Msegs`	Implements the conversion of *memory segments* portion of a configuration vector from XML format to binary format, and vice versa.
`Partitions`	Implements the conversion of *partitions* portion of a configuration vector from XML format to binary format, and vice versa.

Java Class Name	**Description**
`Policies`	Implements the conversion of *policies* (PTP and PAS) portion of a configuration vector from XML format to binary format, and vice versa.
`Processes`	Implements the conversion of *processes* portion of a configuration vector from XML format to binary format, and vice versa.
`ReservedMemory`	Implements the conversion of *run-time reserved memory* portion of a configuration vector from XML format to binary format, and vice versa.
`Runtime`	Implements the conversion of *run-time LPSK* portion of a configuration vector from XML format to binary format, and vice versa.
`SAXValidation`	Validates the input XML file against the given XML schema, and performs additional consistency checks not covered by the rules in the XML Schema.
`Sequencers`	Implements the conversion of *sequencers* portion of a configuration vector from XML format to binary format, and vice versa.
`Subjects`	Implements the conversion of *subject* portion of a configuration vector from XML format to binary format, and vice versa.
`SubjectToDevices`	Implements the conversion of *subject-to-devices permissions* portion of a configuration vector from XML format to binary format, and vice versa.

Java Class Name	**Description**
`SubjectToDsegs`	Implements the conversion of *subject-to-data segments permissions* portion of a configuration vector from XML format to binary format, and vice versa.
`SubjectToEventcounts`	Implements the conversion of *subject-to-eventcounts permissions* portion of a configuration vector from XML format to binary format, and vice versa.
`SubjectToMsegs`	Implements the conversion of *subject-to-memory segments permissions* portion of a configuration vector from XML format to binary format, and vice versa.
`SubjectToSequencers`	Implements the conversion of *subject-to-sequencers permissions* portion of a configuration vector from XML format to binary format, and vice versa.
`SubjectToSubjects`	Implements the conversion of *subject-to-subjects permissions* portion of a configuration vector from XML format to binary format, and vice versa.
`Utils`	Implements a set of utility functions that handles format conversion between XML and binary.

Java Class Name	**Description**
`Vector`	The main class that accepts the input file name, output file name and XML Schema file name, and converts a configuration vector from XML format to binary format, and vice versa. Before converting from XML to binary format, it first checks the XML against the XML Schema. If the XML is not valid, the program displays an error message and exits. After converting from binary to XML format, it first checks the XML against the XML Schema. If the XML is not valid, the program displays an error message and exits. The output file is not saved.

The dependencies between the classes described in Table 8 are illustrated in Figure 17.

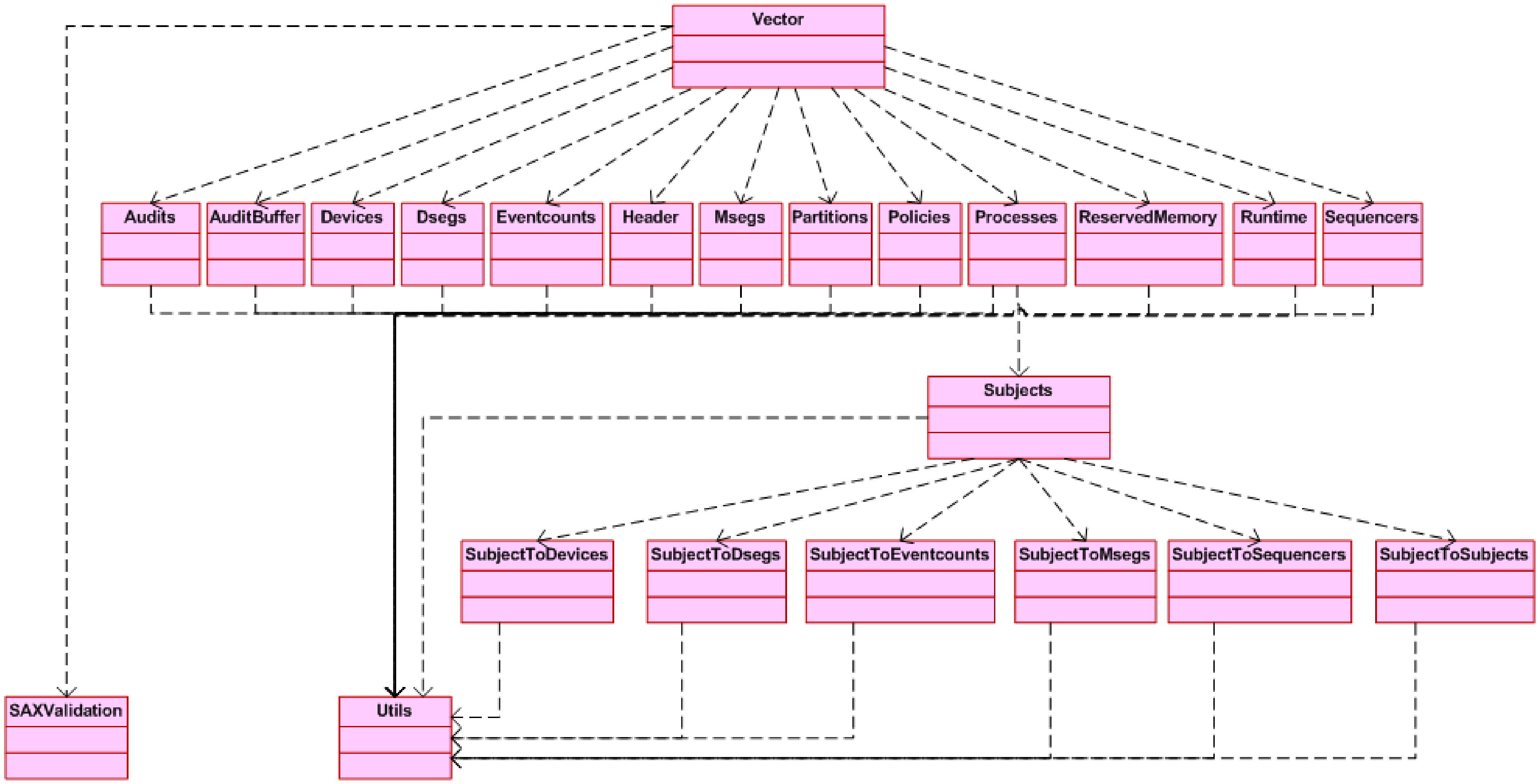


Figure 17. Dependencies between classes in the Format Conversion utility

2. Keyed-Hash Message Authentication Code Generation

The purpose of a message authentication code (MAC) is to verify that the integrity of binary configuration vectors has not been compromised between when they are generated and when they are verified. The MAC generation module receives as inputs a binary configuration vector and a secret password to produce the MAC for the file. To verify the integrity of the configuration vector at a later time, the MAC is recomputed using the same password, and compared with the original MAC. If the two values match, one can be assured that the configuration vector has not been modified.

The ability to generate a HMAC is implemented as a command-line utility, separated from the format conversion utility. A sample interaction with the HMAC utility is shown in Figure 18. The main rationale for providing this feature with a separate tool was to allow it to be readily replaced without an impact to other tools. Doing so also allowed the same utility to be used to hash other LPSK components and files. The cryptographic algorithm used is the SHA-256 algorithm from the Java Cryptography Extension library in Java Platform, Standard Edition 6 [18]. The cryptographic strength of the HMAC largely depends upon the size of the secret key used; a good password should be chosen when generating the HMAC. Otherwise, it makes it easier for an attacker to exploit collisions in the MAC.

```
Enter the key: This is the user password
Enter the file you want to create the HMAC: testconfig.bin
Hash size = 32

6347958ffffffc1ffffffabffffffb8ffffffd153ffffffaa5612ffffff9bffffffb5ff
fffff823fffffff3ffffffedfffffffa71263ffffffbdfffffff6ffffffad5bffffffb2
ffffffffffffffaa30703a
```

Figure 18. HMAC generation command-line interface.

C. SUMMARY

This chapter described design decisions and implementation of the XML Schema, binary configuration vector, format conversion utility and HMAC utility for the LPSK Configuration Vector Tool. The next chapter summarizes the tests conducted on this implementation and the test results.

VI. TESTING

This chapter describes the test cases and test results for the implemented LPSK Configuration Vector Tool. The tests were divided into three categories: functional tests, boundary value tests and consistency checks. The functional tests are designed to verify the functional correctness of the LPSK Configuration Vector Tool with respect to the requirements specification. The boundary values test cases are designed based on maximum and minimum values, typical values and erroneous values of the fields in the LPSK Configuration Vector to verify correct implementation of data validity checks. The consistency checks test cases serve to verify that the rules governing the relationships between XML elements are correctly defined, so that the tool is able to ensure data references in a configuration vector are coherent.

A. TEST CASES

1. Functional Tests

The objective of the following set of tests described in Table 9 is to verify that the Configuration Vector Tool (command line) is able to convert an XML Configuration Vector to its binary format and vice-versa, and contains appropriate error checks on the input parameters.

Table 9. Functional Test Cases

Test #	Purpose	Test Description	Expected Result
Format Conversion utility			
C1.	To verify that the command line Configuration Vector Tool is able to convert an XML Configuration Vector into its binary format.	Run the Configuration Vector Tool, specifying the input XML configuration vector, output binary file and Configuration Vector Schema file.	A message is displayed indicating that the XML file is valid, and is converted and saved to the binary file.
C2.	To verify that the tool is able to handle invalid input XML configuration vectors.	(i) A non-existent XML configuration vector is supplied. (ii) An invalid XML configuration vector is supplied.	(i) & (ii) An error message is displayed indicating that it failed to read the XML file, the tool halts processing and no binary output file is produced.
C3.	To verify that the tool is able to handle invalid input XML Schema (XSD) files.	(i) A non-existent configuration vector XSD is supplied. (ii) An invalid configuration vector XSD is supplied.	(i) & (ii) An error message is displayed indicating that it failed to read the XSD file, the tool halts processing and no binary output file is produced.

Test #	Purpose	Test Description	Expected Result
C4.	To verify that the tool is able to convert a binary Configuration Vector into its XML format.	An input binary configuration vector, output XML file and XSD file is supplied.	A message is displayed indicating that the binary file is saved to the XML file, and the XML file is valid.
C5.	To verify that the tool is able to handle invalid input binary configuration vectors.	An invalid input binary configuration vector is supplied.	An error message is displayed, the tool halts processing and no binary output file is produced.
C6.	To verify that the binary vector can be read by the LPSK.	Load the binary vector and boot up the kernel.	The LPSK boots up according to the configurations in the binary vector.
HMAC utility			
C7.	To verify the HMAC utility is able to generate a 32-byte MAC based on a password string and an input file.	(i) A password and an input filename are supplied to the tool. (ii) No password is provided. (iii) A non-existing filename is supplied.	(i) A 32-byte MAC is displayed. (ii) An error message is displayed indicating that the password cannot be empty. (iii) An error message is displayed indicating that it failed to read the input file.

Test #	Purpose	Test Description	Expected Result
C8.	To verify that the tool generates the same MAC only if the same password and file is used.	(i) The same password and filename are supplied. (ii) The same filename is supplied together with a different password. (iii) The same password is supplied together with a different filename. (iv) A different password and filename is supplied.	(i) The same 32-byte MAC is displayed. (ii) - (iv) A different 32-byte MAC is displayed.

2. Boundary Value Tests

The objective of the following tests described in Table 10 is to ensure that the Configuration Vector Tool checks that the CPU time slices are properly allocated to partitions and processes, and the configurations of the audit buffer, runtime parameters, partitions, subjects and resources are within specified limits. In several cases where a field is constrained to a fixed set of values, only negative test cases are covered in this section because the error messages from inputting invalid values also display the list of valid values. For example, when "the action to take when buffer is full" field contains an invalid value (e.g., "*doshutdown*"), the error message indicates "Value 'doshutdown' is not valid with respect to enumeration '[overwrite, halt, shutdown]', where "[overwrite, halt, shutdown]" is the set of valid values. (Ref. Test C14).

Table 10. Boundary Value Test Cases

Test #	**Purpose**	**Test Description**	**Expected Result**
C9.	To ensure the total of CPU time allocated to all partitions is exactly 100%.	(i) The total *time_slices* of all partitions is equal to 100%. (ii) The total *time_slices* of all partitions is less than 100%. (iii) The total *time_slices* of all partitions exceeds 100%.	(i) No error message is displayed. (ii) & (iii) An error message is displayed indicating that the total time slices must be equal to 100%.
C10.	To ensure that a process is allocated a time slice greater than 0%.	(i) The *time_slice* of a process is set to 1%. (ii) The *time_slice* of a process is set to be less than or equal to 0%. (iii) The home partition of the process is given a *time_slice* of 0%.	(i) No error message is displayed. (ii) An error message is displayed indicating that a process time slice must be greater than 0%. (iii) An error message is displayed indicating that a process cannot reside in a passive partition.

Test #	Purpose	Test Description	Expected Result
C11.	To ensure the sum of time slices allocated to all processes in a partition equal to 100% of the time slices allocated to that partition.	(i) The sum of *time_slices* of all processes of a part_id is equal to 100%. (ii) The sum of *time_slices* of all processes of a part_id is less than 100%. (iii) The sum of *time_slices* of all processes of a part_id exceeds 100%.	(i) No error message is displayed. (ii) & (iii) An error message is displayed indicating that the sum of time slices must be equal to 100% of the time slices allocated to that partition.
Header			
C12.	To ensure the vector description does not exceed 32 characters.	(i) The description contains no characters. (ii) The description contains between 1 and 32 characters. (iii) The description contains 33 characters.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the value is greater than the max length.
Audit Buffer Configuration			
C13.	To ensure the *enable_audit* can only be set to either 0 (disable audit) or 1 (enable audit).	(i) The field is set to 0. (ii) The field is set to 1. (iii) The field is set to other arbitrary invalid values, e.g., false	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the value is invalid.

Test #	Purpose	Test Description	Expected Result
C14.	To ensure only one of the actions {*overwrite, halt, shutdown*} are taken when the audit buffer is full.	The *action_full* field is set to other arbitrary invalid values.	An error message is displayed indicating that the value does not match the defined enumeration.
C15.	To ensure the number of audit records that can be buffered is between 1 and 65535.	(i) The *max_recs* field is set to 1. (ii) The *max_recs* field is set to 65535. (iii) The *max_recs* field is set to 0. (iv) The *max_recs* field is set to 65536.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the element is below the minimum range ("minInclusive" facet-value is violated[1]). (iv) An error message is displayed indicating that the element exceeds the maximum range ("maxInclusive" facet-value is violated[2]).

[1] A *facet* restricts the values of an element to a specific range or length. This message indicates the minimum value constraint has been violated.

[2] This message indicates the maximum value constraint has been violated.

Test #	Purpose	Test Description	Expected Result
C16.	To ensure the maximum delay before shutdown when the audit buffer is full is between 0 and 60 (seconds).	(i) The *delay* field is set to 0. (ii) The *delay* field is set to 60. (iii) The *delay* field is set to -1. (iv) The *delay* field is set to 61.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the element is below the minimum range. (“minInclusive” facet-value is violated). (iv) An error message is displayed indicating that the element exceeds the maximum range. (“maxInclusive” facet-value is violated).
Runtime			
C17.	To ensure the runtime executable is not null and the length of the executable path does not exceed 64 characters.	(i) The executable path contains between 1 and 64 characters. (ii) The executable path is empty. (iii) The executable path contains 65 characters.	(i) No error message is displayed. (ii) An error message is displayed indicating that the value is less than the minimum length. (iii) An error message is displayed indicating that the value is greater than the max length.

Test #	Purpose	Test Description	Expected Result
C18.	To ensure the gate path is not null and length of the gate path does not exceed 64 characters.	(i) The gate path contains between 1 and 64 characters. (ii) The gate path is empty. (iii) The gate path contains 65 characters.	(i) No error message is displayed. (ii) An error message is displayed indicating that the value is less than the minimum length. (iii) An error message is displayed indicating that the value is greater than the max length.
C19.	To ensure the *display* field can only be set to either 0 (do not display) or 1 (display).	(i) The *display* field is set to 0. (ii) The *display* field is set to 1. (iii) The *display* field is set to other invalid arbitrary values.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the value does not match the defined enumeration.
C20.	To ensure the message type field can only be one of {*Status*, *Partition with focus*, *Both*}.	The *msg_type* field is set to other arbitrary invalid values.	An error message is displayed indicating that the value does not match the defined enumeration.
Partitions			

Test #	Purpose	Test Description	Expected Result
C21.	To ensure the duration for one round-robin scheduling of all processes in all partitions is greater than 0.	(i) The duration is set to 1 or more. (ii) The duration is set to 0 or less.	(i) No error message is displayed. (ii) An error message is displayed indicating that the element is below the minimum range.
C22.	To ensure the total number of partitions is between 1 and 256.	(i) 1 partition is specified. (ii) 256 partitions are specified. (iii) There are no partitions specified. (iv) There are 257 partitions specified.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that a partition tag is expected. (iv) An error message is displayed indicating that the partition ID ‘257’ exceeds the maximum range.
C23.	To ensure the partition description does not exceed 32 characters.	(i) The description contains no characters. (ii) The description contains between 1 and 32 characters. (iii) The description contains 33 characters.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the value is greater than the max length.

Test #	Purpose	Test Description	Expected Result
C24.	To ensure that there is at least one active partition defined when a secure attention key (SAK) partition is defined.	(i) No active partition is specified (time slice equals 0). (ii) 1 partition with time slice greater than 0 is defined.	(i) An error message is displayed indicating that at least 1 active partition is required. (*focus_id* cannot refer to a passive partition.) (ii) No error message is displayed.
C25.	To ensure a partition can contain between 0 and 512 processes.	(i) A partition contains no process. (ii) A partition contains 512 processes. (iii) A partition contains 513 processes.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the partition exceeded the maximum number of processes allowed.
Policies			
C26.	To ensure the partition to partition (PTP) mode can only be one of {*NA*, *RO*, *RW*, *WO*}.	The *mode* field is set to other arbitrary invalid values.	An error message is displayed indicating that the value does not match the defined enumeration.
C27.	To ensure the Partitioned Information Flow Policy acyclic subset (PAS) mode can only be one of {*NA*, *RO*, *RW*, *WO*}.	The *mode* field is set to other arbitrary invalid values.	An error message is displayed indicating that the value does not match the defined enumeration.
Data Segments			

Test #	Purpose	Test Description	Expected Result
C28.	To ensure a data segment path is defined and the length of the path does not exceed 64 characters.	(i) The path contains between 1 and 64 characters. (ii) The path contains no characters. (iii) The path contains 65 characters.	(i) No error message is displayed. (ii) An error message is displayed indicating that the value is less than the min length. (iii) An error message is displayed indicating that the value is greater than the max length.
C29.	To ensure a data segment description does not exceed 32 characters.	(i) The description contains no characters. (ii) The description contains between 1 and 32 characters. (iii) The description contains 33 characters.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the value is greater than the max length.

Test #	Purpose	Test Description	Expected Result
C30.	To ensure the privilege level of data segments is between 0 and 3.	A data segment is given a *pl* value of (i) 0 (ii) 3 (iii) -1 (iv) 4	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the element is below the minimum range (“minInclusive” facet-value is violated). (vi) An error message is displayed indicating that the element exceeds the maximum range (“maxInclusive” facet-value is violated).
C31.	To ensure that the start state of a data segment is one of {*swapin, swapout*}.	The *start_state* field is set to other arbitrary invalid values.	An error message is displayed indicating that the value does not match the defined enumeration.
C32.	To ensure that the audit events for data segments are only one of *{swapin, flush, swapout}* X *{success, failure, both}*.	The *event* and *type* field is set to other arbitrary invalid values.	An error message is displayed indicating that the value does not match the defined enumeration.

Test #	Purpose	Test Description	Expected Result
Memory Segments			
C33.	To ensure a memory segment description does not exceed 32 characters.	(i) The description contains no characters. (ii) The description contains between 1 and 32 characters (iii) The description contains 33 characters.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the value is greater than the max length.
C34.	To ensure the privilege level of memory segments are between 0 and 3.	A memory segment is given a *pl* value of (i) 0 (ii) 3 (iii) -1 (iv) 4	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the element is below the minimum range (“minInclusive” facet-value is violated). (vi) An error message is displayed indicating that the element exceeds the maximum range (“maxInclusive” facet-value is violated).
C35.	To ensure the audit events for memory segments are only one of *{created}* X *{success, failure, both}*.	The *event* and *type* field is set to other arbitrary values.	An error message is displayed indicating that the value does not match the defined enumeration.

Test #	Purpose	Test Description	Expected Result
Eventcounts and Sequencers			
C36.	To ensure the total number of eventcounts is less than 64.	(i) No eventcount is specified. (ii) There are 64 eventcounts specified. (iii) There are 65 eventcounts specified.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the synchronization ID '64' exceeds the maximum range.
C37.	To ensure an eventcount description is does not exceed 32 characters.	(i) The description contains no characters. (ii) The description contains between 1 and 32 characters. (iii) The description contains 33 characters.	(i) & (ii) No error message is displayed. (ii) An error message is displayed indicating that the value is greater than the max length.
C38.	To ensure that the audit events for eventcounts are only one of *{advance, read, await, wakeup}* X *{success, failure, both}*.	The *event* and *type* field is set to other arbitrary values.	An error message is displayed indicating that the value does not match the defined enumeration.

Test #	Purpose	Test Description	Expected Result
C39.	To ensure the total number of sequencers is less than 64.	(i) No sequencer is specified. (ii) There are 64 sequencer specified. (iii) There are 65 sequencer specified.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the synchronization ID '64' exceeds the maximum range.
C40.	To ensure a sequencer description is does not exceed 32 characters.	(i) The description contains no characters. (ii) The description contains 32 characters. (iii) The description contains 33 characters.	(i) & (ii) No error message is displayed. (ii) An error message is displayed indicating that the value is greater than the max length.
C41.	To ensure that the audit events for sequencers are only one of *{ticket}* X *{success, failure, both}*.	The *event* and *type* field is set to other arbitrary values.	An error message is displayed indicating that the value does not match the defined enumeration.
Devices			
C42.	To ensure the type field of any device is one of {*data, control*}.	The *type* field is set to other arbitrary values.	An error message is displayed indicating that the value does not match the defined enumeration.

Test #	Purpose	Test Description	Expected Result
C43.	To ensure the description of any device does not exceed 32 characters.	(i) The description contains no characters. (ii) The description contains 32 characters. (iii) The description contains 33 characters.	(i) & (ii) No error message is displayed. (ii) An error message is displayed indicating that the value is greater than the max length.
C44.	To ensure devices can be configured to {*multiplexed*, *dedicated*}.	The *attribute* field is set to other arbitrary values.	An error message is displayed indicating that the value does not match the defined enumeration.
C45.	To ensure the total number of exported keyboard devices is no more than 1. (It is possible to have no connected keyboard in order to support embedded systems)	(i) There is no keyboard device and no SAK partition specified. (ii) There is no keyboard device, but the SAK partition is specified. (iii) There is 1 keyboard device specified. (iv) There are 2 keyboard devices specified.	(i) No error message is displayed. (ii) An error message is displayed indicating that the SAK partition is defined, but no keyboard is configured. (iii) No error message is displayed. (iv) An error message is displayed indicating that the 2nd element is not expected.

Test #	Purpose	Test Description	Expected Result
C46.	To ensure the incoming scan code buffer size of keyboard devices is between 32 and 1024.	(i) The keyboard *buffer_size* is set to 32. (ii) The keyboard *buffer_size* is set to 1024. (iii) The keyboard *buffer_size* is set to 31. (iv) The keyboard *buffer_size* is set to 1025.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the element is below the minimum range (“minInclusive” facet-value is violated). (iv) An error message is displayed indicating that the element exceeds the maximum range (“maxInclusive” facet-value is violated).
C47.	To ensure the total number of PL1 message devices is between 0 and 1.	(i) No PL1 message device is specified. (ii) 1 PL1 message device is specified. (iii) 2 PL1 message devices are specified.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the element is not expected.

Test #	Purpose	Test Description	Expected Result
C48.	To ensure that the number of buffered incoming frames for network devices is between 32 and 256.	(i) A network device is *in_buffer* is set to 32. (ii) A network device is *in_buffer* is set to 256. (iii) A network device is *in_buffer* is set to 31. (iv) A network device is *in_buffer* is set to 257.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the element is below the minimum range ("minInclusive" facet-value is violated). (iv) An error message is displayed indicating that the element exceeds the maximum range ("maxInclusive" facet-value is violated).

Test #	Purpose	Test Description	Expected Result
C49.	To ensure that the number of buffered outgoing frames for network devices is between 32 and 256.	(i) A network device is *out_buffer* is set to 32. (ii) A network device is *out_buffer* is set to 256. (iii) A network device is *out_buffer* is set to 31. (iv) A network device is *out_buffer* is set to 257.	(i) & (ii) No error message is displayed (iii) An error message is displayed indicating that the element is below the minimum range (“minInclusive” facet-value is violated). (iv) An error message is displayed indicating that the element exceeds the maximum range (“maxInclusive” facet-value is violated).
C50.	To ensure the initial states of network devices are one of {*started*, *stopped*}.	The *initial_state* field is set to other arbitrary values.	An error message is displayed indicating that the value does not match the defined enumeration.
C51.	To ensure that network devices can be set to encrypted (1) or unencrypted (0).	(i) The *encryption* field is set to 0. (ii) The *encryption* field is set to 1. (iii) The *encryption* field is set to arbitrary invalid values.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the value does not match the defined data type.

Test #	Purpose	Test Description	Expected Result
C52.	To ensure network devices can be configured to {*promiscuous, non-promiscuous*}.	The *read* field is set to other arbitrary invalid values.	An error message is displayed indicating that the value does not match the defined enumeration.
C53.	To ensure the MAC address format is XX:XX:XX:XX:XX, where X is a hexadecimal value between ‘0’ and ‘F’.	The *mac_address* field is set to other invalid formats.	An error message is displayed indicating that the element does not match the pattern “[0-F]{2}:[0-F]{2}:[0-F]{2}:[0-F]{2}:[0-F]{2}:[0-F]{2}.”
C54.	To ensure the total number of exported screen devices is no more than 1. (It is possible to have no connected screen in order to support embedded systems.)	(i) There is no screen device and no initial focus partition specified. (ii) There is no screen device, but the initial focus partition is specified. (iii) There is 1 screen device specified. (iv) There are 2 screen devices specified.	(i) No error message is displayed. (ii) An error message is displayed indicating that the initial focus partition is defined, but no screen is configured. (iii) No error message is displayed. (iv) An error message is displayed indicating that the element is not expected.

Test #	Purpose	Test Description	Expected Result
C55.	To ensure that the audit events for devices are only one of *{read, data, write data, read metadata, write metadata}* X *{success, failure, both}*.	The *event* and *type* field is set to other arbitrary values.	An error message is displayed indicating that the value does not match the defined enumeration.
Processes and Subjects			
C56.	To ensure the total number of processes in the system in the configuration vector is between 1 and 512, and consequently that there is at least 1 active partition.	(i) There is one process defined. (ii) There are 512 processes defined. (iii) There are no processes defined. (iv) There are 513 processes specified.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the process element is expected. (iv) An error message is displayed indicating that the process *identifier* exceeds the maximum range (because process identifiers are unique).

Test #	Purpose	Test Description	Expected Result
C57.	To ensure the total number of subjects within a process (excluding PL0 subject) is between 1 and 3.	(i) 1 subject is specified within a process.. (ii) 3 subjects are specified within a process. (iii) No subjects are specified within a process. (iv) 4 subjects are specified within a process.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that a subject element is expected. (iv) An error message is displayed indicating that the subject element is not expected.
C58.	To ensure the privilege levels of subjects is between 0 and 3.	A subject is given a *pl* value of (i) 0 (ii) 3 (iii) -1 (iv) 4	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the element is below the minimum range (“minInclusive” facet-value is violated). (vi) An error message is displayed indicating that the element exceeds the maximum range (“maxInclusive” facet-value is violated).

Test #	Purpose	Test Description	Expected Result
C59.	To ensure subjects can be configured to be trusted (1) or un-trusted (0).	(i) The *trusted* field is set to 0. (ii) The *trusted* field is set to 1. (iii) The *trusted* field is set to other arbitrary values.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the value does not match the defined enumeration.
C60.	To ensure the length of the executable path does not exceed 64 characters.	(i) The *exe_path* contains 64 characters. (ii) The *exe_path* is empty. (iii) The *exe_path* contains 65 characters.	(i) No error message is displayed. (ii) An error message is displayed indicating that the value is less than the min length. (iii) An error message is displayed indicating that the value is greater than the max length.
C61.	To ensure the length of the gate path does not exceed 64 characters.	(i) The *gate_path* contains no characters. (ii) The *gate_path* contains 64 characters. (iii) The *gate_path* contains 65 characters.	(i) & (ii) No error message is displayed. (iii) An error message is displayed indicating that the value is greater than the max length.

Test #	Purpose	Test Description	Expected Result
C62.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to eventcounts.	A subject is assigned permission to an eventcount that is not in one of these values.	An error message is displayed indicating that the value does not match the defined enumeration.
C63.	To ensure subjects can only have {*NA*, *RW*} permissions to sequencers.	A subject is assigned permission to a sequencer that is not in one of these values.	An error message is displayed indicating that the value does not match the defined enumeration.
C64.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to data segments.	A subject is assigned permission to a data segment that is not in one of these values.	An error message is displayed indicating that the value does not match the defined enumeration.
C65.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to memory segments.	A subject is assigned permission to a memory segment that is not in one of these values.	An error message is displayed indicating that the value does not match the defined enumeration.
C66.	To ensure subjects can only have {*NA*, *RO*} permissions to keyboards.	A subject is assigned permission to a keyboard device that is not in one of these values.	An error message is displayed indicating that the value does not match the defined enumeration.
C67.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to network devices.	A subject is assigned permission to a network device that is not in one of these values.	An error message is displayed indicating that the value does not match the defined enumeration.

Test #	Purpose	Test Description	Expected Result
C68.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to screen devices.	A subject is assigned permission to a screen device that is not in one of these values.	An error message is displayed indicating that the value does not match the defined enumeration.
C69.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to PL1 message devices.	A subject is assigned permission to a PL1 message device that is not in one of these values.	An error message is displayed indicating that the value does not match the defined enumeration.
C70.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to other subjects.	A subject is assigned permission to a subject that is not in one of these values.	An error message is displayed indicating that the value does not match the defined enumeration.
C71.	To ensure the audit events for subjects are only one of {*interrupt*, *received signal*, *send signal*, *read device*, *write device*, *read eventcount*, *advance eventcount*, *wait eventcount*, *wakeup eventcount*, *ticket sequencer}* X *{success*, *failure*, *both}*.	The *event* and *type* field is set to other arbitrary values.	An error message is displayed indicating that the value does not match the defined enumeration.

3. Consistency Checking Tests

The objective of the tests described in Table 11 is to ensure that the Configuration Vector Tool checks and maintains the referential integrity of partitions, subjects and resources in the configuration vector. Since the correct configuration is already verified in C1, only negative test cases are covered in this section.

Table 11. Consistency Checking Tests

Test #	Purpose	Test Description	Expected Result
Partitions			
C72.	To ensure that partitions are uniquely identified by their *identifier* field.	There is more than one partition with the same identifier.	An error message is displayed indicating that the *partition_pk* identity constraint is violated.
C73.	To ensure the secure attention key (SAK) partition is an active partition, and a keyboard is also configured.	(i) The *partitions sak_id* does not refer to any of the defined partitions. (ii) The *partitions sak_id* refers to a passive partition identifier.	(i) & (ii) An error message is displayed indicating that *sak_id* cannot refer to an invalid or passive partition.
C74.	To ensure the partition that received the initial focus is an active partition.	(i) The *partitions focus_id* does not refer to any of the defined partitions. (ii) The *partitions focus_id* refers to a passive partition.	(i) & (ii) An error message is displayed indicating that *focus_id* cannot refer to an invalid or passive partition.

Test #	Purpose	Test Description	Expected Result
C75.	To ensure the subject partition identifier under partition-to-partition (PTP) policy is an active partition.	(i) The PTP *subj_part_id* does not refer to any of the defined partitions. (ii) The PTP *subj_part_id* refers to a partition identifier that is passive.	(i) An error message is displayed indicating that *subj_part_id* cannot refer to an invalid or passive partition. (ii) An error message is displayed indicating that *subj_part_id* cannot refer to an invalid or passive partition.
C76.	To ensure the resource partition identifier under PTP policy is a valid partition.	(i) The PTP *res_part_id* does not refer to any of the defined partitions. (ii) The PTP *subj_part_id* refers to a partition identifier that is passive.	(i) An error message is displayed indicating that the *res_part_id_fk* identity constraint is violated. (ii) No error message is displayed.
C77.	To ensure the subject partition identifier under Partitioned Information Flow Policy acyclic subset (PAS) is an active partition.	(i) The PAS *subj_part_id* does not refer to any of the defined partitions. (ii) The PAS *subj_part_id* refers to a passive partition.	(i) An error message is displayed indicating that *subj_part_id* cannot refer to an invalid or passive partition. (ii) An error message is displayed indicating that *subj_part_id* cannot refer to an invalid or passive partition.

Test #	Purpose	Test Description	Expected Result
C78.	To ensure the resource partition identifier under PAS is a valid partition.	(i) The PAS *res_part_id* does not refer to any of the defined partitions. (ii) The PAS *res_part_id* is a passive partition.	(i) An error message is displayed indicating that the *res_part_id_fk* identity constraint is violated. (ii) No error message is displayed.
C79.	To ensure that the home partitions of data segments are valid partitions.	The dseg *part_id* does not refer to any of the defined partitions.	An error message is displayed indicating that the *dseg_partition_fk* identity constraint is violated.
C80.	To ensure that the home partitions of memory segments are valid partitions.	The mseg *part_id* does not refer to any of the defined partitions.	An error message is displayed indicating that the *mseg_partition_fk* identity constraint is violated.
C81.	To ensure that the home partitions of eventcounts are valid partitions.	The eventcount *part_id* does not refer to any of the defined partitions.	An error message is displayed indicating that the *eventcount_partition_fk* identity constraint is violated.

Test #	Purpose	Test Description	Expected Result
C82.	To ensure that the home partitions of sequencers are valid partitions.	The sequencer *part_id* does not refer to any of the defined partitions.	An error message is displayed indicating that the *sequencer_partition_fk* identity constraint is violated.
C83.	To ensure that the home partitions of devices are valid partitions.	The device *part_id* does not refer to any of the defined partitions.	An error message is displayed indicating that the *device_partition_fk* identity constraint is violated.
C84.	To ensure that the initial partition of devices are valid partitions.	The device *init_part_id* does not refer to any of the defined partitions.	An error message is displayed indicating that the *device_initpartition_fk* identity constraint is violated.

Test #	Purpose	Test Description	Expected Result
C85.	To ensure that processes are mapped to partitions that have been allocated time slices greater than 0%.	(i) The process *part_id* does not refer to any of the defined partition identifier. (ii) The process *part_id* value refers to a partition with no time slice allocated.	(i) An error message is displayed indicating that the *process_partition_fk* identity constraint is violated. (ii) An error message is displayed indicating that *part_id* cannot refer to an invalid or partition with no time slices allocated.
Subject-Resource-Permissions to Resources			
C86.	To ensure that eventcounts are uniquely identified by their *identifier* field.	There is more than one eventcount with the same identifier.	An error message is displayed indicating that the *eventcount_pk* identity constraint is violated.
C87.	To ensure eventcount permissions refer to valid eventcounts.	The eventcount *identifier* does not refer to any of the defined eventcounts.	An error message is displayed indicating that the *subj_eventcount_fk* identity constraint is violated.
C88.	To ensure that sequencers are uniquely identified by their *identifier* field.	There is more than one sequencer with the same identifier.	An error message is displayed indicating that the *sequencer_pk* identity constraint is violated.

Test #	Purpose	Test Description	Expected Result
C89.	To ensure sequencer permissions refer to valid sequencers.	The *identifier* value does not refer to any of the defined sequencers.	An error message is displayed indicating that the *subj_sequencer_fk* identity constraint is violated.
C90.	To ensure that data segments are uniquely identified by their *path*.	There is more than one data segment with the same path.	An error message is displayed indicating that the *dseg_pk* identity constraint is violated.
C91.	To ensure data segment permissions refer to valid data segments.	The *path* value does not refer to any of the defined dsegs.	An error message is displayed indicating that the *subj_dseg_fk* identity constraint is violated.
C92.	To ensure that memory segments are uniquely identified by their *identifier*.	There is more than one memory segment with the same identifier.	An error message is displayed indicating that the *mseg_pk* identity constraint is violated.
C93.	To ensure memory segment permissions refer to valid memory segments.	The *identifier* value does not refer to any of the defined memory segments.	An error message is displayed indicating that the *subj_mseg_fk* identity constraint is violated.
C94.	To ensure that devices are uniquely identified by the *<major, minor, type>* triple.	There is more than one device with the same *<major, minor, type>*.	An error message is displayed indicating that the *device_pk* identity constraint is violated.

Test #	Purpose	Test Description	Expected Result
C95.	To ensure device permissions refer to valid devices.	The <*major, minor, type*> triple does not refer to any of the defined devices.	An error message is displayed indicating that the *subj_device_fk* identity constraint is violated.
C96.	To ensure that processes are uniquely identified by their *identifier* field.	There is more than one process with the same identifier.	An error message is displayed indicating that the *process_pk* identity constraint is violated.
C97.	To ensure that subjects are uniquely identified by the <*process_id, pl*> tuple.	There is more than one subject with the same <*process_id, pl*>.	An error message is displayed indicating that the *subj_unique* identity constraint is violated.
C98.	To ensure subject permissions refer to valid subjects.	The <*process_id, pl*> triple does not refer to any of the defined subjects.	An error message is displayed indicating that the *subj_subj_fk* identity constraint is violated.

B. TEST RESULTS

The tests described in Section A were automated using the JUnit testing framework and pre-generated configuration vectors. Automating the tests enabled quick verification that an update introduced into the XML Schema would not cause other tests to fail. The test setup and procedures are documented in Appendix B. All the tests completed successfully. The test results are summarized in Tables 12 to 14.

Table 12. Functional Test Results

Test #	Purpose	Test Result
Format conversion utility		
C1.	To verify that the command line Configuration Vector Tool is able to convert an XML Configuration Vector into its binary format.	Passed
C2.	To verify that the tool is able to handle invalid input XML files.	Passed
C3.	To verify that the tool is able to handle invalid input XML Schema (XSD) files.	Passed
C4.	To verify that the tool is able to convert a binary Configuration Vector into its XML format.	
C5.	To verify that the tool is able to handle invalid input binary configuration vectors.	Passed
C6.	To verify that the binary vector can be read by the LPSK.	Passed
HMAC utility		
C7.	To verify the HMAC utility is able to generate a 32-byte MAC based on a password string and an input file.	Passed
C8.	To verify that the tool generates the same MAC only if the same password and file is used.	Passed

Table 13. Boundary Value Test Results

Test #	**Purpose**	**Test Result**
C9.	To ensure the total of CPU time allocated to all partitions is exactly 100%.	Passed
C10.	To ensure that a process is allocated a time slice greater than 0%.	Passed
C11.	To ensure the sum of time slices allocated to all processes in a partition equal to 100% of the time slices allocated to that partition.	Passed
Header		
C12.	To ensure the vector description does not exceed 32 characters.	Passed
Audit Buffer Configuration		
C13.	To ensure the *enable_audit* can only be set to either 0 (disable audit) or 1 (enable audit).	Passed
C14.	To ensure only one of the actions {*overwrite, halt, shutdown*} are taken when the audit buffer is full.	Passed
C15.	To ensure the number of audit records that can be buffered is between 1 and 65535.	Passed
C16.	To ensure the maximum delay before shutdown when the audit buffer is full is between 0 and 60 (seconds).	Passed
C17.	To ensure the runtime executable is not null and the length of the executable path does not exceed 64 characters.	Passed
C18.	To ensure the gate path is not null and length of the gate path does not exceed 64 characters.	Passed

Test #	Purpose	Test Result
C19.	To ensure the *display* field can only be set to either 0 (do not display) or 1 (display).	Passed
C20.	To ensure the message type field can only be one of {*Status*, *Partition with focus*, *Both*}.	Passed
Partitions		
C21.	To ensure the duration for one round-robin scheduling of all processes in all partitions is greater than 0.	Passed
C22.	To ensure the total number of partitions is between 1 and 256.	Passed
C23.	To ensure the partition description does not exceed 32 characters.	Passed
C24.	To ensure that there is at least one active partition defined when a secure attention key (SAK) partition is defined.	Passed
C25.	To ensure a partition can contain between 0 and 512 processes.	Passed
Policies		
C26.	To ensure the partition to partition (PTP) mode can only be one of {*NA*, *RO*, *RW*, *WO*}.	Passed
C27.	To ensure the Partitioned Information Flow Policy acyclic subset (PAS) mode can only be one of {*NA*, *RO*, *RW*, *WO*}.	Passed
Data Segments		
C28.	To ensure a data segment path is defined and the length of the path does not exceed 64 characters.	Passed
C29.	To ensure a data segment description does not exceed 32 characters.	Passed
C30.	To ensure the privilege level of data segments is between 0 and 3.	Passed

Test #	Purpose	Test Result
C31.	To ensure that the start state of a data segment is one of {*swapin, swapout*}.	Passed
C32.	To ensure that the audit events for data segments are only one of *{swapin, flush, swapout}* X *{success, failure, both}*.	Passed
C33.	To ensure a memory segment description does not exceed 32 characters.	Passed
C34.	To ensure the privilege level of memory segments are between 0 and 3.	Passed
C35.	To ensure that the audit events for memory segments are only one of *{created}* X *{success, failure, both}*.	Passed
Eventcounts and Sequencers		
C36.	To ensure the total number of eventcounts is less than 64.	Passed
C37.	To ensure an eventcount description is does not exceed 32 characters.	Passed
C38.	To ensure that the audit events for eventcounts are only one of *{advance, read, await, wakeup}* X *{success, failure, both}*.	Passed
C39.	To ensure the total number of sequencers is less than 64.	Passed
C40.	To ensure a sequencer description is does not exceed 32 characters.	Passed
C41.	To that the audit events for ensure sequencers are only one of *{ticket}* X *{success, failure, both}*.	Passed
Devices		
C42.	To ensure the type field of any device is one of {*data, control*}.	Passed
C43.	To ensure the description of any device does not exceed 32 characters.	Passed
C44.	To ensure devices can be configured to {*multiplexed, dedicated*}.	Passed

Test #	**Purpose**	**Test Result**
C45.	To ensure the total number of exported keyboard devices is no more than 1. (It is possible to have no connected keyboard in order to support embedded systems)	Passed
C46.	To ensure the incoming scan code buffer size of keyboard devices is between 32 and 1024.	Passed
C47.	To ensure the total number of PL1 message devices is between 0 and 1.	Passed
C48.	To ensure that the number of buffered incoming frames for network devices is between 32 and 256.	Passed
C49.	To ensure that the number of buffered outgoing frames for network devices is between 32 and 256.	Passed
C50.	To ensure the initial states of network devices are one of {*started*, *stopped*}.	Passed
C51.	To ensure that network devices can be set to encrypted (1) or unencrypted (0).	Passed
C52.	To ensure network devices can be configured to {*promiscuous*, *non-promiscuous*}.	Passed
C53.	To ensure the MAC address format is XX:XX:XX:XX:XX, where X is a hexadecimal value between ‘0’ and ‘F’.	Passed
C54.	To ensure the total number of exported screen devices is no more than 1. (It is possible to have no connected screen in order to support embedded systems.)	Passed
C55.	To ensure that the audit events for devices are only one of *{read, data, write data, read metadata, write metadata}* X *{success, failure, both}*.	Passed

Test #	Purpose	Test Result
Processes and Subjects		
C56.	To ensure the total number of processes in the system in the configuration vector is between 1 and 512, and consequently that there is at least 1 active partition.	Passed
C57.	To ensure the total number of subjects within a process (excluding PL0 subject) is between 1 and 3.	Passed
C58.	To ensure the privilege levels of subjects is between 0 and 3.	Passed
C59.	To ensure subjects can be configured to be trusted (1) or un-trusted (0).	Passed
C60.	To ensure the length of the executable path does not exceed 64 characters.	Passed
C61.	To ensure the length of the gate path does not exceed 64 characters.	Passed
C62.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to eventcounts.	Passed
C63.	To ensure subjects can only have {*NA*, *RW*} permissions to sequencers.	Passed
C64.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to data segments.	Passed
C65.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to memory segments.	Passed
C66.	To ensure subjects can only have {*NA*, *RO*} permissions to keyboards.	Passed
C67.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to network devices.	Passed
C68.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to screen devices.	Passed

Test #	**Purpose**	**Test Result**
C69.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to PL1 message devices.	Passed
C70.	To ensure subjects can have {*NA*, *RO*, *RW*, *WO*} permissions to other subjects.	Passed
C71.	To ensure that the audit events for subjects are only one of {*interrupt, received signal*, *send signal, read device*, *write device*, *read eventcount, advance eventcount, wait eventcount*, *wakeup eventcount*, *ticket sequencer}* X *{success*, *failure*, *both}*.	Passed

Table 14. Consistency Checking Test Results

Test #	**Purpose**	**Test Result**
Partitions		
C72.	To ensure that partitions are uniquely identified by their *identifier* field.	Passed
C73.	To ensure the secure attention key (SAK) partition is an active partition, and a keyboard is also configured.	Passed
C74.	To ensure the partition that received the initial focus is an active partition.	Passed
C75.	To ensure the subject partition identifier under partition-to-partition (PTP) policy is an active partition.	Passed
C76.	To ensure the resource partition identifier under PTP policy is a valid partition.	Passed
C77.	To ensure the subject partition identifier under Partitioned Information Flow Policy acyclic subset (PAS) is an active partition.	Passed

Test #	Purpose	Test Result
C78.	To ensure the resource partition identifier under PAS is a valid partition.	Passed
C79.	To ensure that the home partitions of data segments are valid partitions.	Passed
C80.	To ensure that the home partitions of memory segments are valid partitions.	Passed
C81.	To ensure that the home partitions of eventcounts are valid partitions.	Passed
C82.	To ensure that the home partitions of sequencers are valid partitions.	Passed
C83.	To ensure that the home partitions of devices are valid partitions.	Passed
C84.	To ensure that the initial partition of devices are valid partitions.	Passed
C85.	To ensure that processes are mapped to partitions that have been allocated time slices greater than 0%.	Passed
C86.	To ensure that eventcounts are uniquely identified by their *identifier* field.	Passed
C87.	To ensure eventcount permissions refer to valid eventcounts.	Passed
C88.	To ensure that sequencers are uniquely identified by their *identifier* field.	Passed
C89.	To ensure sequencer permissions refer to valid sequencers.	Passed
C90.	To ensure that data segments are uniquely identified by their *path*.	Passed
C91.	To ensure data segment permissions refer to valid data segments.	Passed
C92.	To ensure that memory segments are uniquely identified by their *identifier*.	Passed
C93.	To ensure memory segment permissions refer to valid memory segments.	Passed

Test #	Purpose	Test Result
C94.	To ensure that devices are uniquely identified by the <*major, minor, type*> triple.	Passed
C95.	To ensure device permissions refer to valid devices.	Passed
C96.	To ensure that processes are uniquely identified by their *identifier* field.	Passed
C97.	To ensure that subjects are uniquely identified by the <*process_id, pl*> tuple.	Passed
C98.	To ensure subject permissions refer to valid subjects.	Passed

C. SUMMARY

This chapter outlined the tests performed on the implemented LPSK Configuration Vector Tool, and summarized the test results. All the test results were consistent with their expected results. The next chapter concludes this thesis and provides recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSION

A. RESULTS

This work resulted in significant contributions. A new configuration vector format was defined, in XML and binary representations, to support new features and enhancements to the LPSK that impose additional configuration requirements. A XML-based representation of the LPSK configuration, including its security policies, proved to be viable. It enforced a structure on the configuration vector, with tags that describe the content. The LPSK Configuration Vector XML format was formalized through the definition of its XML Schema using XSD 1.0. XSD 1.0 has a rich set of data types, and the identity constraint definition (using *key, keyref* and *unique*) feature was useful. It allowed most of the validation rules to be expressed in the XML Schema, such that configuration vectors could be validated by standard XML parsers and editors. It also allowed users to use the auto-complete feature in some XML editors to suggest valid values. Nonetheless, the XSD 1.0 grammar is limited and could not express all the required configuration vector validation rules that were required. Instead some of the rules had to be coded into the format conversion utility.

The new LPSK configuration vector format has many new fields, making editing a complex task. Several GUI-based XML editors were evaluated based on functional, usability and other requirements. The ability to validate XML configuration vectors against the XML Schema developed is a pertinent feature. Other features that enhanced usability included good visualization, auto-completion of data based on the XML Schema, informative error messages that indicate corrective actions required, and help for users to locate sources of errors easily and quickly. These features serve to remove many of the underlying complexities of creating configuration vectors from users.

A suite of utilities that are components of the LPSK Configuration Vector Tool was developed that validates configuration vectors against the XML Schema, converts them from XML format to binary format and vice-versa, and generates HMAC for configuration vectors to provide assurance of their integrity. These utilities underwent extensive functional tests, including boundary and consistency checking tests, to provide confidence that the implementation is sound; it passed all the tests. Manually testing all the fields in the configuration vector was extremely time-consuming. Therefore, the tests were automated using the JUnit testing framework, enabling rapid verification that updates introduced into the XML Schema and other code changes would not cause other tests to fail.

B RELATED WORK

There are existing commercial-off-the-shelf implementations of high assurance separation kernels. Wind River provides the VxWorks MILS platform that is intended to meet real-time operating system requirements for high assurance (EAL6+) MLS embedded systems [19]. An XML configuration tool suite, which also includes a GUI, is provisioned to allow configuration of the system parameters as well as application-specific and middleware-specific run-time parameters. It also includes an XML-to-binary compiler that translates configuration data into binary format. This configuration tool suite appears to be similar to the LPSK Configuration Tool utilities that were implemented.

Lynuxworks offers LynxSecure, an embedded hypervisor and separation kernel that is being designed to be certifiable to EAL7 and to satisfy the SKPP [20]. The configuration is defined and maintained by an XML definition file and system tools. Green Hills Software is another embedded systems software company; their INTEGRITY 178B separation kernel is certified to EAL6+ and satisfies the SKPP requirements [21]. The details concerning the level of granularity of information flow controls and the configuration tools that are available for both kernels are not publicly available.

C. RECOMMENDATIONS FOR FUTURE WORK

The LPSK is still in its infancy; many features defined in the functional specifications are either undergoing development or have not been developed. Hence, the new configuration vector could not be fully tested in the LPSK. The LPSK was modified to read in the new binary vector and translate the configuration into the format that the

LPSK currently recognizes; new configuration settings were not utilized. The LPSK requires extensive changes before the new configuration vector format can be fully tested. An incremental approach is proposed. Currently, some of the LPSK configuration settings (e.g., device configurations) are hardcoded. An audit subsystem was concurrently being developed in another project, and many of the new settings are audit-related. Going forward, enhancements would be to enable the LPSK to read these settings from the new configuration vector and to use the new configuration vector format with the new audit mechanism.

Currently, any change to the configuration vector requires a re-initialization of the LPSK. A potential enhancement is to allow a subset of the configuration settings to be dynamically changed (i.e., during run-time without re-initialization). For example, memory segments are currently statically allocated during initialization, and cannot be re-sized without re-booting the LPSK. Providing for dynamic reconfiguration gives the LPSK more flexibility in managing memory resources. This is also a valid operational requirement for emergency response systems. However, this has to be carefully analyzed to ensure it does not introduce policy enforcement inconsistencies into the LPSK, and without violating any requirements in the SKPP.

The W3C XML Schema Definition Language (XSD) 1.1 specification [22] addresses some of the limitations with XSD 1.0, using assertions and rules for evaluation using XPath 2.0 [23]. However, the specification is still in the draft stage, with limited tool support. None of the XML editors evaluated in this thesis supports the XSD 1.1 specifications. There are other non-W3C specifications, such as RELAX NG [24] and Schematron [25]; these were not explored. These languages can potentially be used to overcome the limitations of XSD 1.0 in expressing the more complex configuration vector validation. If all the validation rules are captured in a single place (i.e., in the XML schema), the design will be more elegant and it will be easier to manage future changes to the configuration vector format. An in-depth study into the features of these languages is recommended.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. INSTALLATION GUIDE

This appendix describes the installation procedure for the LPSK Configuration Vector Tool utilities.

A. SYSTEM REQUIREMENTS

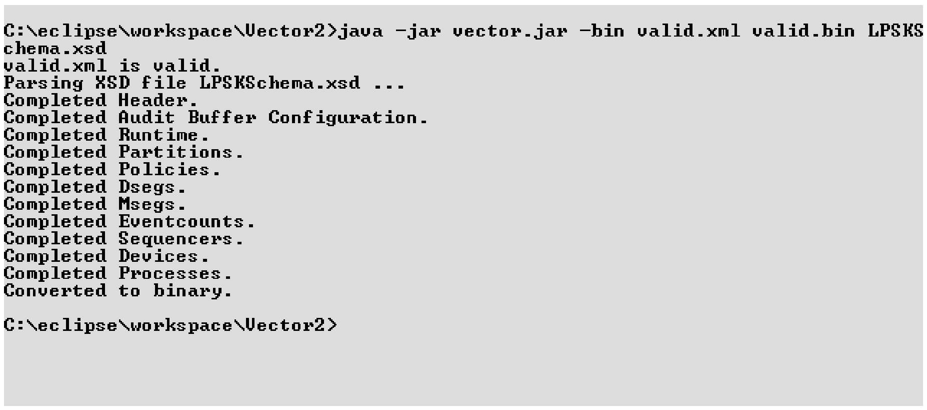
The tool requires a commodity operating system (e.g., Windows) with the Java Standard Edition Virtual Machine (JVM) installed. The tool has been successfully tested on *jre 1.6* (Java Runtime Environment). The XML editor, XMLSpy, runs natively on Windows platform. The Altova Web site states that virtualization software, Parallels for Mac and Wine, is required for running it on MacOS and RedHat Linux respectively.

B. PROCEDURES

1. The LPSK Configuration Vector Tool distribution contains the following files. Create and copy them into a working directory (e.g., *c:\config_vector*).

    - *hmac.jar* - the HMAC utility
    - *vector.jar* - the Format Conversion utility
    - *LPSKSchema.xsd* - the LPSK XML Schema. This does not have to be in the same directory as the jar files.
    - *valid.xml* - a sample XML configuration vector that conforms to *LPSKSchema.xsd*. This does not have to be in the same directory as the jar files.

2. To verify the installation, execute the command "*java -jar vector.jar -bin valid.xml valid.bin LPSKSchema.xsd*", where *valid.xml* is the input XML configuration vector, *valid.bin* is the output binary configuration vector and *LPSKSchema.xsd* is the XML Schema file. The full path to these files must be specified if they do not reside in the working directory. Figure 19 shows the expected output.

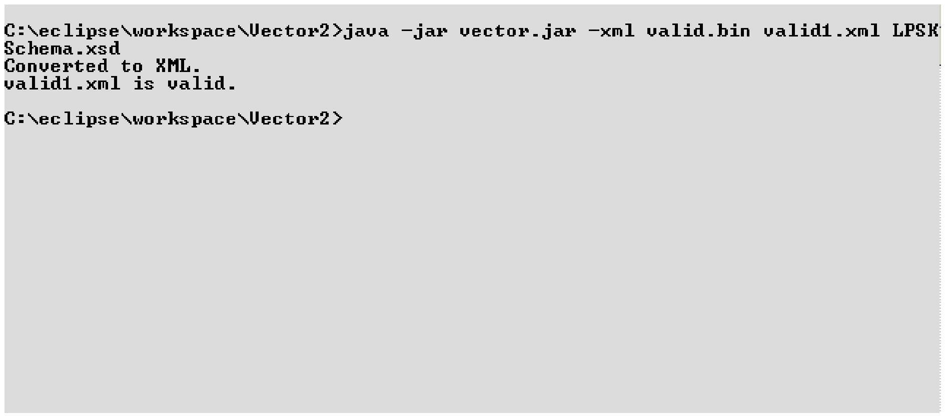


```
C:\eclipse\workspace\Vector2>java -jar vector.jar -bin valid.xml valid.bin LPSKS
chema.xsd
valid.xml is valid.
Parsing XSD file LPSKSchema.xsd ...
Completed Header.
Completed Audit Buffer Configuration.
Completed Runtime.
Completed Partitions.
Completed Policies.
Completed Dsegs.
Completed Msegs.
Completed Eventcounts.
Completed Sequencers.
Completed Devices.
Completed Processes.
Converted to binary.

C:\eclipse\workspace\Vector2>
```

Figure 19. Expected output of converting XML configuration vector to binary

3. Execute the command *"java -jar vector.jar -bin valid.xml valid.bin LPSKSchema.xsd"*, where *valid.xml* is the input XML configuration vector, *valid.bin* is the output binary configuration vector and *LPSKSchema.xsd* is the XML Schema file. The full path to these files must be specified if they do not reside in the working directory. Figure 20 shows the expected output.



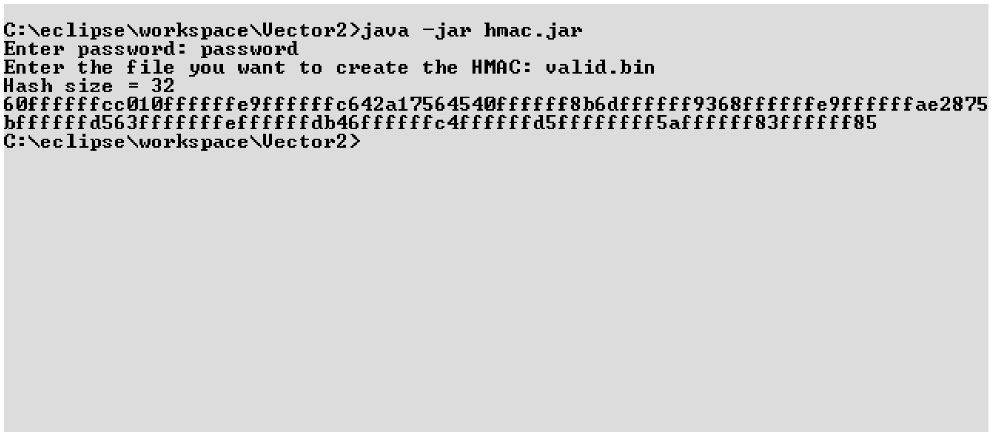
```
C:\eclipse\workspace\Vector2>java -jar vector.jar -xml valid.bin valid1.xml LPSK
Schema.xsd
Converted to XML.
valid1.xml is valid.

C:\eclipse\workspace\Vector2>
```

Figure 20. Expected output of converting binary configuration vector to XML

4. Execute the command "*java -jar hmac.jar*."
5. Enter the password when prompted. A strong password should be chosen; otherwise it makes it easier for an attacker to exploit collisions in the MAC.

6. Enter the name of the file to create the HMAC when prompted. Figure 21 shows the expected output.



```
C:\eclipse\workspace\Vector2>java -jar hmac.jar
Enter password: password
Enter the file you want to create the HMAC: valid.bin
Hash size = 32
60ffffffcc010ffffffe9ffffffc642a17564540ffffff8b6dffffff9368ffffffe9ffffffae2875
bffffffd563fffffffeffffffdb46ffffffc4ffffffd5ffffffff5affffff83ffffff85
C:\eclipse\workspace\Vector2>
```

Figure 21. Expected output from the HMAC utility

7. To install XMLSpy (XML editor), download the installer from the Altova Web site (http://www.altova.com/download-current.html) and follow the installation instructions. Enter the key-code provided by Altova to unlock the software. An alternative XML editor may also be used if desired.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. TEST PROCEDURES

This appendix describes the test procedures to execute the test cases outlined in Chapter VI. Most of the tests take pre-generated XML configuration vectors as inputs to the program, where the procedures were automated using JUnit 3 test suites. It is assumed that the installation disk that contains the test suites is available to the tester.

A. SETUP

The setup requires the following:

- The run time LPSK environment is required by Test C3.
- The format conversion utility (`vector.jar`) and the HMAC conversion utility (`hmac.jar`).
- A terminal running Windows (e.g., Windows XP Professional SP3), and the *eclipse.zip* file, which contains the Eclipse IDE and a JUnit project named *Vector_Tests*. *Vector_Tests* contains the following:
    - `src\tests` - This directory contains the source code for the test cases.
    - `tests` - This directory contains the XML configuration vectors pre-generated for each test case.
    - `LPSKSchema.xsd` - This is the LPSK XML Schema file used by the test cases. The file name is hard coded into each test class, and must be replaced with the version to be tested.
    - `.classpath` - This file contains entries to classes and external libraries used.

The following procedure sets up the Eclipse IDE for running the JUnit tests. If a different operating system is used, an Eclipse IDE for the target operating system must be installed and the test project imported into the Eclipse workspace.

- Open the *Vector_Tests* project in the Eclipse IDE.
- If the *Vector_Tests* project is not available in *Package Explorer*, load the project into the workspace as follows:
    - Select *File* > *Import*. A dialog box is displayed.
    - From the dialog box, under "*Select an import source*", select "*Existing Projects into Workspace*" and click "*Next*."
    - From the *Import Projects* dialog box, select "*Select Root Directory*" option, and click the "*Browse*" button. Browse to the *Vector_Tests* project directory.
    - Select the *Vector_Tests* project that appears in *Projects*. Check "*Copy projects into workspace*." The *Vector_Tests* project contains references to the Format Conversion utility (`vector.jar`) and the HMAC utility (`hmac.jar`). The locations of these files are listed in `.classpath`.
    - Click "*Finish*" to import the project. The project now appears in *Package Explorer*.
- From *Package Explorer*, open the *Vector_Tests* project and expand *src* > *tests*. A list of JUnit test classes is displayed.

B. PROCEDURES

1. Functional Tests

Table 15 outlines the procedure for executing the boundary value test cases described in Chapter VI Section A2.

Table 15. Functional Test Procedures

Test #	Procedure	Expected Result
Format Conversion utility		
C1	From the Windows terminal, open a command line window. Navigate to the directory where the format conversion utility is installed. Run "`java -jar vector.jar -bin <Vector_Tests directory>\tests\functional\valid.xml out.bin <Vector_Tests directory>\LPSKSchema.xsd`." <*Vector_Tests directory*> is the location where the *Vector_Tests* project is installed.	A message is displayed indicating that the XML file is valid, and is converted and saved to the binary file.

Test #	Procedure	Expected Result
C2	(i) Run “`java -jar vector.jar -bin <Vector_Tests directory>\tests\functional\invalid.xml out.bin <Vector_Tests directory>\LPSKSchema.xsd`”, where `invalid.xml` is an invalid XML configuration vector. (ii) Run “`java -jar vector.jar -bin <Vector_Tests directory>\tests\functional\missing.xml o.bin <Vector_Tests directory>\LPSKSchema.xsd`”, where `missing.xml` is a non-existing file. (iii) Run “`java -jar vector.jar -bin <Vector_Tests directory>\tests\functional\empty.xml out.bin <Vector_Tests directory>\LPSKSchema.xsd`”, where `empty.xml` is an empty XML configuration vector.	(i) An error message is displayed, the tool halts processing and no binary output file is produced. (ii) An error message is displayed indicating that it failed to read the input file, the tool halts processing and no binary output file is produced. (iii) An error message is displayed indicating the input file is empty, the tool halts processing and no binary output file is produced.

Test #	Procedure	Expected Result
C3	(i) Run "`java -jar vector.jar -bin <Vector_Tests directory>\tests\functional\valid.xml valid.bin missing.xsd`", where `missing.xsd` is a non-existing file. (ii) Run "`java -jar vector.jar -bin <Vector_Tests directory>\tests\functional\valid.xml out.bin invalid.xsd`", where `invalid.xsd` is an invalid XSD.	(i) & (ii) An error message is displayed indicating that it failed to read the XSD file, the tool halts processing and no binary output file is produced.
C4	From the command line, run "`java -jar vector.jar -xml <Vector_Tests directory>\tests\functional\valid.bin out.xml <Vector_Tests directory>\LPSKSchema.xsd`."	A message is displayed indicating that the binary file is converted to XML, and that the XML file is valid.

Test #	Procedure	Expected Result
C5	(i) Run "`java -jar vector.jar -xml <Vector_Tests directory>\tests\functional\invalid.bin valid.xml <Vector_Tests directory>\LPSKSchema.xsd`", where is `invalid.bin` an invalid binary configuration vector. (ii) Run "`java -jar vector.jar -xml <Vector_Tests directory>\tests\functional\missing.bin valid.xml <Vector_Tests directory>\LPSKSchema.xsd`", where is `missing.bin` a non-existing binary configuration vector. (iii) Run "`java -jar vector.jar -xml <Vector_Tests directory>\tests\functional\empty.bin valid.xml <Vector_Tests directory>\LPSKSchema.xsd`", where `empty.bin` is an empty binary file.	(i) An error message is displayed, the tool halts processing and no binary output file is produced. (ii) An error message is displayed indicating that it failed to read the input file, the tool halts processing and no binary output file is produced. (iii) An error message is displayed indicating the file is empty, the tool halts processing and no binary output file is produced.
C6	This test requires the LPSK runtime environment. Boot up the LPSK with the binary configuration vector generated in Test C1.	The LPSK boots up according to the configurations specified in the binary configuration vector.

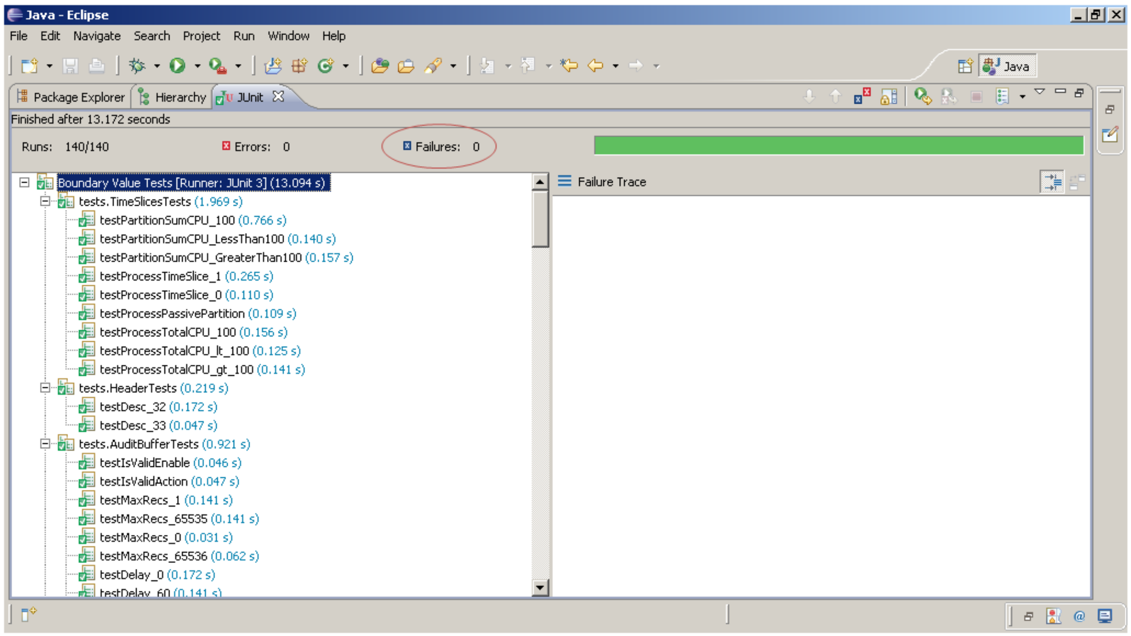
Test #	Procedure	Expected Result
HMAC utility		
C7	(i) From the command line, run "`java -jar hmac.jar`." Enter the password and the path of the file to create the HMAC when prompted. Note the value of the MAC generated. (ii) Repeat Test C6 without supplying any passwords. (iii) Repeat Test C6, supplying a non-existing filename.	(i) A 32-byte MAC is displayed. (ii) An error message is displayed indicating that the password cannot be empty. (iii) An error message is displayed indicating that it failed to read the input file.
C8	(i) Repeat Test C6, using the same password and file. Compare MAC generated with that displayed in C6. (ii) Repeat Test C6, using a different password and the same file. Compare MAC generated with that displayed in C6. (iii) Repeat Test C6, using the same password, but a different file. Compare MAC generated with that displayed in C6. (iv) Repeat Test C6 with a different password and filename. Compare MAC generated with that displayed in C6.	(i) The same 32-byte MAC is displayed. (ii) - (iv) A different 32-byte MAC is displayed.

2. Boundary Value Tests

Table 16 outlines the procedure for executing the boundary value test cases described in Chapter VI Section A2.

Table 16. Boundary Value Test Procedures

Test #	Procedure	Expected Result
C9 - C71	**Run the JUnit boundary value test suite** Select and right-click on the file *BoundaryValueTestSuite.java*, then select *Run As > JUnit Test.*	JUnit reports there are no failures (Figure 22). The expected error messages are displayed in the console view.



```
Finished after 13.172 seconds
Runs:   140/140        Errors:   0        Failures:   0

Boundary Value Tests [Runner: JUnit 3] (13.094 s)
  tests.TimeSlicesTests (1.969 s)
    testPartitionSumCPU_100 (0.766 s)
    testPartitionSumCPU_LessThan100 (0.140 s)
    testPartitionSumCPU_GreaterThan100 (0.157 s)
    testProcessTimeSlice_1 (0.265 s)
    testProcessTimeSlice_0 (0.110 s)
    testProcessPassivePartition (0.109 s)
    testProcessTotalCPU_100 (0.156 s)
    testProcessTotalCPU_lt_100 (0.125 s)
    testProcessTotalCPU_gt_100 (0.141 s)
  tests.HeaderTests (0.219 s)
    testDesc_32 (0.172 s)
    testDesc_33 (0.047 s)
  tests.AuditBufferTests (0.921 s)
    testIsValidEnable (0.046 s)
    testIsValidAction (0.047 s)
    testMaxRecs_1 (0.141 s)
    testMaxRecs_65535 (0.141 s)
    testMaxRecs_0 (0.031 s)
    testMaxRecs_65536 (0.062 s)
    testDelay_0 (0.172 s)
    testDelay_60 (0.141 s)
```

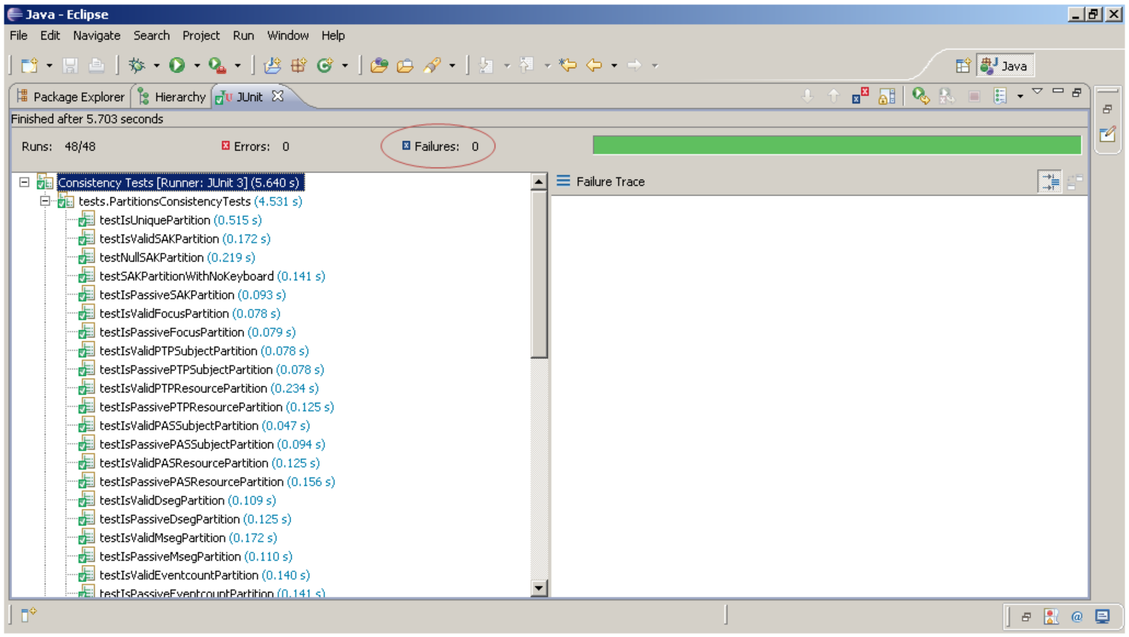
Figure 22. Expected results of the boundary value tests

3. Consistency Checking Tests

Table 17 outlines the procedure for executing the consistency checking test cases described in Chapter VI Section A3.

Table 17. Consistency Checking Test Procedures

Test #	Procedure	Expected Result
C72 - C98	**<u>Run the JUnit consistency test suite</u>** Select and right-click on the file *ConsistencyTestSuite.java*. Then select *Run As > JUnit Test*.	JUnit reports there are no failures (Figure 23). The expected error messages are displayed in the console view.



```
Finished after 5.703 seconds
Runs:   48/48          Errors:   0          Failures:   0

Consistency Tests [Runner: JUnit 3] (5.640 s)
  tests.PartitionsConsistencyTests (4.531 s)
    testIsUniquePartition (0.515 s)
    testIsValidSAKPartition (0.172 s)
    testNullSAKPartition (0.219 s)
    testSAKPartitionWithNoKeyboard (0.141 s)
    testIsPassiveSAKPartition (0.093 s)
    testIsValidFocusPartition (0.078 s)
    testIsPassiveFocusPartition (0.079 s)
    testIsValidPTPSubjectPartition (0.078 s)
    testIsPassivePTPSubjectPartition (0.078 s)
    testIsValidPTPResourcePartition (0.234 s)
    testIsPassivePTPResourcePartition (0.125 s)
    testIsValidPASSubjectPartition (0.047 s)
    testIsPassivePASSubjectPartition (0.094 s)
    testIsValidPASResourcePartition (0.125 s)
    testIsPassivePASResourcePartition (0.156 s)
    testIsValidDsegPartition (0.109 s)
    testIsPassiveDsegPartition (0.125 s)
    testIsValidMsegPartition (0.172 s)
    testIsPassiveMsegPartition (0.110 s)
    testIsValidEventcountPartition (0.140 s)
    testIsPassiveEventcountPartition (0.141 s)
```

Figure 23. Expected results of the consistency checking tests.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] T. D. Nguyen, T. E. Levin and C. E. Irvine, “TCX Project: High Assurance for Secure Embedded Systems,” *Proceedings of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 21–25, March 2005.

[2] P.C. Clark, D. J. Shifflett, C. E. Irvine, T. D. Nguyen and T. E. Levin, “Trusted Computing Exemplar (TCX) Least Privilege Separation Kernel (LPSK) Product Functional Specification Volume I High Level Description,” 6 May 2010.

[3] World Wide Web Consortium, “Extensible Markup Language (XML) 1.0 (Fifth Edition),” W3C Recommendation 26 November 2008, http://www.w3.org/TR/REC-xml, Last Accessed: September 2010.

[4] J. Rushby, “The Design and Verification of Secure Systems,” 8th ACM Symposium on Operating System Principles, *ACM Operating Systems Review*, Vol. 15, No. 5 pp. 12–21.

[5] National Security Agency. “U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness,” Version 1.03, 29 June 2007.

[6] J. H. Saltzer and M. D. Schroeder, “The Protection of Information in Operating Systems,” *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[7] T. M. Welliver, “Configuration Tool Prototype for the Trusted Computing Exemplar Project,” Master’s Thesis, Naval Postgraduate School, December 2009.

[8] World Wide Web Consortium, “Web Services Policy 1.5 – Framework,” W3C Recommendation 04 September 2007, http://www.w3.org/TR/ws-policy, Last Accessed: September 2010.

[9] Giovanni Della-Libera et al., “Web Services Security Policy Language Version 1.1,” July 2005, http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf, Last Accessed: September 2010.

[10] World Wide Web Consortium, “XML Schema Part 0: Primer Second Edition,” W3C Recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-0-20041028, Last Accessed: September 2010.

[11] World Wide Web Consortium, “XML Schema Part 1: Structures Second Edition,” W3C Recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028, Last Accessed: September 2010.

[12] World Wide Web Consortium, “XML Schema Part 2: Datatypes Second Edition,” W3C Recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028, Last Accessed: September 2010.

[13] World Wide Web Consortium, “Guide to the W3C XML Specification ("XMLspec") DTD, Version 2.1,” http://www.w3.org/XML/1998/06/xmlspec-report.htm, Last Accessed: September 2010.

[14] World Wide Web Consortium, “Document Object Model (DOM) Level 1 Specification Version 1.0,” W3C Recommendation, 1 October, 1998, http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001, Last Accessed: September 2010

[15] SourceForge.Net, “About SAX,” http://www.saxproject.org/about.html, Last Accessed: September 2010.

[16] National Institute of Standards and Technology, “NIST’s Policy on Hash Functions,” http://csrc.nist.gov/groups/ST/hash/policy.html, 15 March 2006, Last Accessed: October 2010.

[17] B. Doyle, “XML Editors Review,” June 13, 2006, http://www.cmsreview.com/XML/Editors, Last Accessed: September 2010.

[18] Oracle, “Java ^{TM} Cryptography Architecture API Specification & Reference,” http://download.oracle.com/javase/1.5.0/docs/guide/security/CryptoSpec.html, 25 July 2004, Last Accessed: September 2010.

[19] Wind River, “Wind River VxWorks MILS Platform Features,” http://cdn.windriver.com/products/platforms/vxworks-mils/features.html, Last Accessed: November 2010.

[20] Lynuxworks, “LynxSecure Embedded Hypervisor and Separation Kernel,” http://www.lynuxworks.com/virtualization/hypervisor.php, Last Accessed: November 2010.

[21] Green Hills Software, “W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures,” W3C Working Draft 3 December 2009, Last Accessed: September 2010.

[22] World Wide Web Consortium, “W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures,” W3C Working Draft 3 December 2009, Last Accessed: September 2010.

[23] World Wide Web Consortium, “XML Path Language (XPath) 2.0,” W3C Recommendation 23 January 2007, Last Accessed: September 2010.

[24] International Organization for Standardization, “Information technology – Document Schema Definition Languages (DSDL) Part 2: Regular-grammar-based validation - RELAX NG,” ISO/IEC 19757-2, 1 June 2006.

[25] International Organization for Standardization, “Information technology - Document Schema Definition Languages (DSDL) Part 3: Rule-based validation – Schematron,” ISO/IEC 19757-3, 1 June 2006.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Kris Britton
   National Security Agency
   Fort Meade, MD

4. John Campbell
   National Security Agency
   Fort Meade, MD

5. Deborah Cooper
   DC Associates, LLC
   Reston, VA

6. Grace Crowder
   NSA
   Fort Meade, MD

7. Louise Davidson
   National Geospatial Agency
   Bethesda, MD

8. Vincent J. DiMaria
   National Security Agency
   Fort Meade, MD

9. Rob Dobry
   NSA
   Fort Meade, MD

10. Jennifer Guild
    SPAWAR
    Charleston, SC

11. CDR Scott Heller
    SPAWAR
    Charleston, SC

12. Dr. Steven King
ODUSD
Washington, DC

13. Steve LaFountain
NSA
Fort Meade, MD

14. Dr. Greg Larson
IDA
Alexandria, VA

15. Dr. Carl Landwehr
National Science Foundation
Arlington, VA

16. Dr. John Monastra
Aerospace Corporation
Chantilly, VA

17. John Mildner
SPAWAR
Charleston, SC

18. Dr. Victor Piotrowski
National Science Foundation
Arlington Virginia

19. Jim Roberts
Central Intelligence Agency
Reston, VA

20. Ed Schneider
IDA
Alexandria, VA

21. Mark Schneider
NSA
Fort Meade, MD

22. Keith Schwalm
Good Harbor Consulting, LLC
Washington, DC

23. Ken Shotting
NSA
Fort Meade, MD

24. Dr. Ralph Wachter
ONR
Arlington, VA

25. Dr. Cynthia E. Irvine
Naval Postgraduate School
Monterey, CA

26. Paul C. Clark
Naval Postgraduate School
Monterey, CA

27. Yeo Tat Soon
National University of Singapore
Singapore

28. Tan Lai Poh
National University of Singapore
Singapore

29. Quek Chee Luan
Defence Science & Technology Agency
Singapore