Demonstration of the eNotarization Markup Language (ENML) Version 1.0

Proposal Prepared for
the eNotarization Technical Committee
Legal XML Member Section
Organization for the
Advancement of Structured Information Standards

Western Illinois University
Laurence L. Leff, Ph.D.

This demonstration will allow the notary to prepare an eNotarized document, in ENML of course, given the binary for the document in some format such as XML, Word, PDF, etc. We will prepare a web site and web service so that a person or business having this eNML document can verify that it was signed by the notary. We support a symmetric key cipher. That is the Notary Administration in the Secretary of State in the notary's state **(NA)** will issue the notary a key by which they sign their documents. They will also keep a key copy which will be used in the verification web site and web service.

The input to the first process will be the document to be notarized **(DTBN)** and file containing the notary's key **(NK)**. The output will be the eNotarized document.

Rolly Chambers and myself identified that the highest risk area of this project is the core that encrypts a document with a symmetric cipher to produce a hash representing the notarization process. The first phase of this sponsored project will thus be a command-line exercise and demonstration of this capability. This mini-demo will consist of two command-line programs, corresponding to each process. The first command-line program will take as input the document to be notarized and the file containing the notary's key.
It will produce an XML document containing the following minimal set of tags:

1) `ds:SignatureValue`
2) `enml:SignedDocument` (but without the `mimeType` or comments)

(The three command line arguments will be

1) the file path for the **DTBN**
2) the file path for the **NK**
3) the file path at which to write the minimal XML output file)

The second process and command-line program will take as input, an example of the output as described above and the **NK.** It will extract a copy of the original document and provide an appropriate message if the key does not match. Thus, we will demonstrate tampering with the document and this process detecting that event.

Phase Two of Project

As before, the first program and process is used by the Notary Public to notarize the document and create the ENML file. However, the first process will create a complete ENML document. The notary will enter the necessary information using by a Java Swing application. (The Swing Library is the standard way in which programmers create graphical user interfaces for programs running on the desktop, such as in Windows or an X-windows-based Graphical User Interface for UNIX.) The second process will be implemented as both a web site and a web service.

The first time it is used, it will bring up a Swing screen that will have the following inputs:

a) the Notary's private key, which will be sent from the **NA** office
b) the Notary's name
c) the notary Commission's number
d) when the Commission Expires
e) the Notary Jurisdiction--we will use the United States form for this prototype and will read the County, State and Country.
f) The Notary bond number
Items b through f will be read with standard Java Swing GUI facilities. The notary will be prompted to upload the Notary's private key from a file.

The notary will select the document from a tree view. This is the familiar manner by which individuals choose documents in a tool such as Microsoft Word or the attachment phase of sending an electronic mail. There will be a button for the notary to "inspect" the document. The program will directly display of an ASCII or XML file. It will invoke from standard locations Microsoft Word and Adobe to display those types of documents for inspection. (Note, we will rely on the person setting up the demo to put these programs in specific locations on the computer being used for the demo.)
The system will allow the notary to enter:
1) the `SignerIdentificationMethod`

2) the date (which will override the current date if entered)
3) the location (which will override the default of the Notary Commisson Information in `enotary.in`)
4) The Notary Identification Type (`Acknowledgement`, `Jurat`, etc. as defined in the `NotaryIdentificationType` element)
5) the Notarization address (where this notarization occurred, Section 4.46).

It will then ask the signer to enter in the:
a) statutory text for the signature if any
b) the signer's name
c) the signer's title (optional)
d) the signer's prefix (such as Judge or Mr.)
e) the signer's address (we will use the United States form for the prototype, `SignerUSAAdress`)
f) `SignerIdentificationMethod`
g) the signer's address
h) `SignerSignature`, e. g. `/s= Mr. Ying Liu`

(For our prototype, we will simply represent these by passing the keyboard from the notary to the signer.) All of these functions will be done using standard Graphical User Interface displays developed using the Java Swing package.

It will produce the ENML, electronically signing the document using the private key. It will also set the the `ds:RetrievalMethod` so as to point to the web site of the **NA.**

The second process will validate an ENML document. It will be implemented two ways:
1) a web service
2) a web site

We will implement a client program to demonstrate the web service. That program will allow the user to specify using a Directory Tree View, the notarized file to be updated. It will go the web site indicated in the `ds:RetrievalMethod`, and verify that it was validly signed and eNotarized. Assuming it was, it would display information about the document such as the notary's commission. It would also reverse the BASE64 encoding of the document and ask the user where to save the file. Again, the user will have a standard GUI tree view, similar to saving a file in Microsoft Word or when downloading a file from the web.

This interaction will be implemented as a web service on the **NA** site in AXIS. Validation will be an RPC method with attachments sent back and forth between the **NA** site and the verifier application that will consume the service. Firms receiving many notarized documents such as those in the Mortgage Industry could put this web service invocation in their normal workflow processing.

We will also implement an alternative, a web site where users can easily verify single or a few documents. The web site will have a place for the user to upload the ENML document. It will display via the browser a message as to whether the ENML document was valid and also allow the user to download the processed document.

For the purposes of the demonstration, we are not not exercising witnessed documents, documents where the user as opposed to the notary was using any kind of cryptographic key to sign, public key or asymmetric key technology, ENML documents where the notary electronically notarized several documents at once or documents with multiple signatures. Nor are we demonstrating apostille's.

Cryptographic Algorithm

John Messing raised the issue of which symmetric cryptoalgorithm we should use. At that point, I had only identified the DES algorithm as it was well known. Other possibilities are the built in Java Cryptography Architecture (JCA) which is apparently part of Java. Nippon, Camelia Source Code, put out a Java version in public domain under both the GNU Public License and BSD license.

Limitations

As we discussed, this set of programs although functional is not commercial-grade or complete. This is for two reasons. It saves the Legal XML group funds. But more importantly, it is not the role of OASIS technical committees to compete with the software vendors. (Note, the committee would need to decide the disposition of the software; developers would find it helpful to extract the functionality and then build their business interface around it.)

A particular limitation is **security.** The TC has decided to use symmetric keys. Thus the **NA** will have to keep a copy of each symmetric key for each of the notaries that have been commissioned. Should the computer containing these be compromised, then the **NA** would have to prepare a new key for each of its notaries, and documents that are notarized with the old keys. That is because any person possessing who downloaded the set of cryptographic keys could electronically impersonate any of the notaries.

We will not verify that the date returned from the system date functions is in fact the true date. That means that a person compromising the computer containing the keys from the **NA,** can potentially electronically enotarize a document so that it appears that it was done before the date of the breach. We also note that a person attempting to create a falsified ENML document could change the date on their computer or simply create their own program. We believe these security issues would be a problem with any enotarization system based upon symmetric keys.

We will use and test one recent version of the Java SDK and Apache Software including the web server. However, we do not anticipate any incompatibilities. Similarly, we will use and test with only one of the major browsers (Internet Explorer, Mozilla FireFox or Safari). That being said, we do not anticipate that there will be incompatibilities and we believe the software should work in all recent versions of the Java software and the browsers.

Phases, Structure of Schedule for Deliverables and Payments

This work will be done in four phases. Upon accepting the contract, we anticipate that the Organization for the Advancement of Structured Information Standards will send remittance for the first $a + m$ hours at p ~~dollars per hour. These will be expended as Mr. Yeluri works on this effort.~~ Receipt by Western Illinois of University of the acceptance of the contract by OASIS will be the **ACCEPTANCE TIME.**

The first phase will be for a hours, and done in a short number of days b. In addition to the software itself in Java, we will provide a script log of the computer running these command-line programs. And we will provide a written explanation that could be used as a script in a demonstration at a conference.

The TC will review this and quickly decide if it wanted to proceed with phase two. This is implementation of the complete eNotarization Markup Language (ENML) Graphical User Interface for the notaries and the verification web site. For convenience, we will do this on the Computer Science Department LINUX computer, toolman.wiu.edu. (Note that this computer is currently behind a firewall.)

This will take m hours and n additional days. There will also be two bonus deadlines. The clock for these will start at the **ACCEPTANCE TIME**. m will be a shorter time period than n and hence the bonus for meeting it will be larger. Upon completion of this, the Technical Committee will determine whether the work done meets this information and will be accepted. Such acceptance shall only be witheld if the TC, can state in its reasonable professional opinion that the software prepared Western Illinois University does not do what is specified in this document.

While the TC is reviewing the work prior to acceptance, Western Illinois University will move the software to the server of their choice. Upon acceptance and installation on this server, OASIS shall pay a completion bonus to Western Illinois University.

Mr. Chaitanya Yeluri and Dr. Laurence L. Leff of Western Illinois University will go to conferences as needed to speak and to help demo the software. Of course both individuals are limited on the number of times during the semester. This is because their main responsibility are their studies and teaching respectively.

This will be invoiced at *q* dollars per hour for Mr. Yeluri's time. OASIS will also pay travel expenses for Dr. Laurence Leff and Chaitanya Yeluri or just the former if Mr. Yeluri is not available. If Dr. Laurence Leff travels on behalf of this project to conferences with Mr. Yeluri or he travels during the times when Western Illinois University classes are not in session, his time in travelling to support the outreach mission of the eNotarization Technical committee will be free.