Technical Brief

May 2012

SEARCH
The National Consortium for Justice Information and Statistics

Using Java Tools to Implement the National Data Exchange (N-DEx) Information Exchange Package Documentation (IEPD)

By Yogesh Chawla
SEARCH

What is N-DEx?

The Law Enforcement National Data Exchange (N-DEx) is the FBI's national repository of law enforcement incident and offender data that includes incident and case reports, booking and incarceration data, and parole/probation information. A key capability of N-DEx is the ability to detect relationships between people, vehicles/property, location, and/or crime characteristics contained in these data. It "connects the dots" between data that may not otherwise seem to be related and supports investigative activities across jurisdictional boundaries—enhancing national information sharing, links between regional and state systems, and virtual regional information sharing.[1]

LEXS: The Foundation of the N-DEx IEPD

The N-DEx Information Exchange Package Documentation (IEPD)[2] uses the National Information Exchange Model (NIEM)[3] and a NIEM-based standard called the Logical Entity Exchange Specification Publish Discover (LEXS-PD) IEPD. LEXS-PD is a generic IEPD that defines many commonly used data elements and objects found in justice information exchanges; it also allows the user to define custom content. Two important high-level components are embodied in two key LEXS data structures: the **LEXS digest** and **structured payload(s)**—

[1] Description from N-DEx website: http://www.fbi.gov/about-us/cjis/n-dex

[2] Download the N-DEx IEPD at: http://it.ojp.gov/default.aspx?area=implementationAssistance&page=1017&standard=520

[3] NIEM website: https://www.niem.gov/Pages/default.aspx

- The **LEXS digest** allows implementers to specify people, places, and things using a predefined structure. For a national exchange like N-DEx, this structure is consistent nationally to enable common understanding across disparate agencies. For example, the digest can include subjects, victims, witnesses, home addresses, incident addresses, evidence, and vehicles—each defined as a separate entity. The entities are then associated with each other using XML references. For example, a subject can be associated with an address and/or vehicle. LEXS provides a comprehensive set of possible associations.

- The LEXS IEPD uses the **structured payload** to allow for customized content not contained in the LEXS digest. Implementers may use the structured payload to define any new data or to add additional elements or attributes to a digest entity. For example, an exchange might require a subject's Blood Alcohol Content. The LEXS digest does not contain this element and requires implementers to add these data using the structured payload. To do this, the structured payload would define a person and use a "same as digest" XML element reference to point to the person defined in the LEXS digest. The person entity defined in the LEXS digest now has additional data content contained in the structured payload. LEXS allows multiple structured payloads to accommodate multiple exchanges, if necessary.

The power of LEXS is that the LEXS digest defines much of the information required in any law enforcement exchange. Once data from a given system are mapped to LEXS, it will have a base level of interoperability with any other system that has done the same. This greatly simplifies the implementation process and improves information sharing capabilities, both for N-DEx connectivity and for cross-agency communication.

Strategies to Create an N-DEx Document

This *Technical Brief* addresses how to work with XML in Java for the purpose of implementing the N-DEx IEPD. To fully understand all of the concepts, it is helpful if the reader has a basic understanding of the Java programming language, XML Schema, and Document Object Model (DOM). However, the concepts discussed here are also applicable to other programming languages.

To create an XML document in Java, implementers use three common approaches:

1. Object binding with Java API[4] for XML Binding (JaxB),[5] XMLBeans,[6] or other framework to generate Java code.

   Advantages:
   - Provides a layer of abstraction so developers work with Java code rather than directly with XML.

[4] Application Programming Interface: http://en.wikipedia.org/wiki/Application_programming_interface

[5] Java API for XML Binding: http://jaxb.java.net/

[6] XMLBeans: http://xmlbeans.apache.org/

- Generates code according to XML schema to produce appropriate data types, including enumerated values for codes.

Drawbacks:
- Tightly couples implementation to schema and requires refactoring if schema is updated.
- Marshaling and unmarshaling[7] to and from XML and Java can be resource-intensive.

2. Transforming an XML document from some other format to the N-DEx format, using eXtensible Stylesheet Language Transformation (XSLT).

Advantages:
- Allows conversion from any existing XML document to the N-DEx IEPD XML document.
- Has broad support in many programming languages and frameworks.

Drawbacks:
- Requires an existing XML document to convert from.
- XSLT's declarative programming style can be a challenge to inexperienced developers.

3. Creating the document using an XML framework such as Java DOM, DOM4J,[8] or JDOM. These XML frameworks provide an API for creating and modifying XML documents that abstract many of the low-level details from the developer.

Advantages:
- Allows the developer to create XML documents using Java with a great deal of flexibility.

Drawbacks:
- XML documents are created element by element, so the toolkit provides very little support for a specific XML schema (though schema validation is supported).
- The developer has to manually set the correct data type for the XML element, as opposed to generated code that sets the data type according to the XML schema.

Each approach listed here has its advantages and disadvantages, and depending on the project, developer experience, and business requirements, one approach might be favored over another. For example, if an agency that intends to map to N-DEx has an existing Incident Report extract in XML, it would be prudent to explore the XSLT approach. However, if no existing XML extract is available, XSLT might add an additional, unnecessary step. This *Technical Brief* will explore the object binding approach in more

[7] Marshaling and Unmarshaling: http://en.wikipedia.org/wiki/Marshalling_%28computer_science%29

[8] DOM4J: http://dom4j.sourceforge.net/

detail since there are some unique challenges with code generation, which can be time-consuming to develop and requires advanced knowledge of the Java tooling.

Challenges with LEXS/N-DEx and Object Generation

The LEXS digest uses a specific subset of the NIEM. LEXS IEPDs may also contain a structured payload that also uses NIEM—typically a different (but generally overlapping) NIEM subset. When attempting to generate code using the object binding approach, the generated code will typically have package name collisions between the LEXS digest and the structured payload. This is because JaxB uses the schema's target namespace to generate the Java package name. For example, if a LEXS-based IEPD contains two subsets, each of which includes the niem-core namespace, the JaxB tooling will create classes with duplicate package names, causing compilation errors. Implementers may choose to unify the subsets into a single subset, but this approach modifies the original IEPDs and can cause the resulting documents to be invalid.

To work around the namespace collision issue, developers must perform code generation and schema validation in multiple passes. The LEXS digest IEPD code is generated and placed into a Java Archive (JAR) file, and then the N-DEx structured payload IEPD is generated separately and customized to use different package names. A simple approach is to prefix every package in the N-DEx structured payload with **ndex**. This will guarantee that all package names are unique. If there is more than one structured payload in the document, there would be multiple rounds of code generation and different package name prefixes would be selected.

JaxB Maven Code Generation Projects

The Georgia Tech Research Institute (GTRI) has created a Maven[9] JaxB project that generates Java code for LEXS 3.1.4. This project is available for download[10] and contains unit tests that verify that sample LEXS documents can be marshaled and unmarshaled to and from Java classes.

SEARCH worked with code originally developed by the State of Wisconsin to develop a Maven JaxB project that will generate code for the N-DEx Incident Arrest (N-DEx IA) IEPD and provide customizations to avoid package name collisions. Users can take a similar approach with other N-DEx IEPDs.

The resulting artifacts of these two projects are JAR files containing the generated code. With the JAR files in place, users can write a sample implementation that creates a full LEXS/N-DEx document in Java code.

The process of generating code and writing the requisite JaxB code is tedious and distracts from the main focus of the project—mapping data from the data store to the N-DEx format. *Following this approach, much of the code need only be developed once and can be reused multiple times.* If every agency or vendor seeking to connect to N-DEx using Java leverages

[9] Apache Maven: http://maven.apache.org/

[10] Download at http://www.lexsdev.org/content/tools

this sample implementation, they will significantly reduce project cost and time. In addition, the N-DEx implementation process across various state and local agencies should not vary greatly. Thus, once a vendor has written an N-DEx implementation for one customer, they should be able to reuse this capability nationally at minimal or no additional cost.

Creating the LEXS/N-DEx IA Document in Java

The FBI N-DEx Program Office[11] distributes many sample LEXS/N-DEx instance documents with the N-DEx IEPD. These examples are detailed and thorough, and address most of the LEXS/N-DEx mappings. In order to make a useful Java example, SEARCH and the State of Wisconsin have written a J-Unit[12] test that recreates the Burglary Incident Arrest example[13] that N-DEx distributes. This J-Unit test creates both the LEXS container and the N-DEx IA structured payload. A developer can look at the sample Burglary Incident Arrest instance document and see the corresponding Java code that creates the XML. *With this template code complete, static values in the code can be replaced with actual data from a database or other data store*.

Java Projects

There are three Java projects required to create the LEXS/N-DEx IA document:

1. **lexs-jaxb** – LEXS JaxB Maven project distributed at http://www.lexsdev.org/content/tools

2. **NdexAutoGen** – N-DEx IA JaxB Maven project distributed by SEARCH and the State of Wisconsin. Please contact SEARCH for the project source code.[14]

3. **NDExSubmissionConnector** – Contains the J-Unit test that creates the N-DEx burglary incident arrest sample document. Please contact SEARCH for the project source code.

Document Validation and N-DEx Tools

There are a variety of tools to assist in mapping data and creating valid N-DEx exchanges. Although many rules can be enforced in XML Schema, others cannot and must be validated outside the schema. The FBI has developed an online tool—the Conformance Testing Assistant (ConTesA)[15]—to address and validate many of these business rules. It produces a detailed report indicating what changes are necessary to create an accurate and complete N-DEx submission.

[11] See http://www.fbi.gov/about-us/cjis/n-dex/ndex_overview

[12] J-Unit: http://junit.sourceforge.net/

[13] The example document is named **burglary-incident-w-arrest-basic.xml** and is located in the "xml" folder of the N-DEx IA IEPD.

[14] Submit your request via our online form at http://www.search.org/about/contact/#form

[15] See https://contesa.ittl.gtri.org/contesa/

N-DEx also provides many helpful documents with the N-DEx IEPD download, including the Component Mapping Template (CMT) spreadsheet, the master document, and the code tables spreadsheet. These documents provide a starting point for mapping data from an internal data model to N-DEx IEPDs. When beginning an N-DEx project, it is best to contact the N-DEx Program Office[16] to get more information on how to use these tools to facilitate data mapping.

Next Steps

Once developers create the N-DEx IEPD, they will invoke a Web Service to submit the document to the FBI. N-DEx security requirements dictate that partners submit data over an SSL connection with X509 mutual certificate authentication.[17] The service specification also contains a WS-Security[18] policy that requires digitally signed messages. The open source community provides several projects that support this functionality:

- **Apache CXF** – Open source Web Services framework that implements most WS-* standards and provides a number of data bindings, including JaxB. See http://cxf.apache.org/

- **Apache WSS4J** – Open source project that provides implementation of the WS-Security standard. This integrates seamlessly with Apache CXF. See http://ws.apache.org/wss4j/

- **Apache Camel** – Open source implementation of Enterprise Integration Patterns that allow for routing and mediation of messages between a vast array of components in a simple Java or XML syntax.

Summary

This *Technical Brief* shows how to develop N-DEx documents using Object Binding with JaxB. Future publications will focus on implementing information exchanges using Apache Camel, Apache CXF, and Apache WSS4J.

[16] Contact the office via email at ndex@leo.gov

[17] X509 Mutual Certificate Authentication: http://fusesource.com/docs/broker/5.3/security/SSL-UseCerts.html

[18] WS-Security: http://en.wikipedia.org/wiki/WS-Security

This project was supported by Grant No. 2009-D1-BX-K007 awarded by the Bureau of Justice Assistance. The Bureau of Justice Assistance is a component of the Office of Justice Programs, which also includes the Bureau of Justice Statistics, the National Institute of Justice, the Office of Juvenile Justice and Delinquency Prevention, the SMART Office, and the Office for Victims of Crime. Points of view or opinions in this document are those of the author and do not represent the official position or policies of the United States Department of Justice.

Captain Thomas W. Turner
Chairman

Ronald P. Hawley
Executive Director

Scott M. Came
Deputy Executive Director

SEARCH
7311 Greenhaven Drive, Suite 270 • Sacramento, CA 95831
(916) 392-2550 • (916) 392-8440 (fax) • www.search.org