Evolution of Emergency Data Exchange Language (EDXLL): A Framework/Toolkit for Developers

Presentation of the OASIS Emergency Management Technical Committee
September 22, 2017
Introduction

Organization for the Advancement of Structured Information Standards (OASIS)

Rex Brooks,
- Secretary, OASIS Emergency Management Technical Committee (EM-TC)
  rexb@starbourne.com
  +1 510-898-0670

Elysa Jones,
- Chair, OASIS Emergency Management Technical Committee (EM-TC)

Jeff Waters,
- Co-Chair, OASIS EM Reference Information Model Subcommittee (EM-RIM-SC)
The Beginning...

- Stimulated by 9/11/2001
- Improved Emergency Response...
  - Requires rapid information sharing -- Interoperability
    - Across Diverse Emergency Management Systems
    - Across Organization and Jurisdictional Boundaries

... and Now
OASIS Emergency Data Exchange Language

- Provides a Suite of Standardized Message Formats
  - Started with Common Alerting Protocol (CAP) for Alerting Messages
    - v1.0 March 2004
    - v1.1 October 2005
    - V1.2 July 2010
  - Followed by the EDXL-Distribution Element (EDXL-DE) for Standardized Routing for Emergency Messages and other digital resources like image, audio and video files
    - v1.0 May 2006
    - v2.0 June 2013
OASIS Emergency Data Exchange Language

- Provides a Suite of Standardized Message Formats
  - Then came EDXL-Resource Messaging (EDXL-RM) for the variety of messages involved with handling emergency Logistics
    - v1.0 Nov. 2008
  - Simultaneously with EDXL Hospital Availability Exchange (EDXL-HAVE) v1.0 for Reporting the Availability of Hospital Resources
    - v1.0 Nov. 2008
OASIS Emergency Data Exchange Language

- Provides a Suite of Standardized Message Formats
  - Then came EDXL Situation Reporting (EDXL-SitRep) for Situational Awareness and Decision Support messages
    - v1.0 Nov. 2012
  - Followed by EDXL Tracking Emergency Patients (EDXL-TEP) for the continuum of patient information from emergency site through hospital admission/transfer
    - v1.0 Jan. 2014
    - v1.1 Jan. 2016
  - Simultaneously with Tracking Emergency Clients (EDXL-TEC) Registry
    - v1.0 Jun. 2014
Challenge Now: Improve Support and Encourage Adoption

- With EDXL TEP we finished the base set of EDXL Specifications
  - With no new specifications in the pipeline, we entered a new phase of development
  - Began to refine the set as a whole entity by completing the Reference Information Model (RIM)
  - Continuation of the process started with Supporting Elements: edxl-ct, edxl-gsf, edxl-ciq, extensions and profiles
  - Discovered minor differences, so we’re looking at revisions of each specification to achieve unification and internal consistency.
Challenge Now: Improve Support and Encourage Adoption

- Primary Goal: Increase Adoption of EDXL by Attracting Developers to use EDXL
  - Not enough developers involved with EDXL
  - Lack of supporting materials available for development
  - Need to increase outreach to developers, other stakeholders to drive up demand
  - New ideas, stakeholders, participants, developers welcome
EDXL Framework/Toolkit

- A Software Library
  - Provides uniform, easy-to-use common functions
    - e.g. Read/Write/Validate/Store/Send/Convert Messages
  - Java and C# Class Libraries for all EDXL Specifications
    - Class Libraries generated from EDXL-RIM through Enterprise Architect
    - .Net C# Class Libraries available with ‘EDXL Sharp’ developed by MITRE
      https://edxlsharp.codeplex.com/
EDXL Framework/Toolkit

- Example CAP Library

http://github.com/google/cap-library/

---------

The CAP Library is a collection of code and tools to work with public alerting messages in the [Common Alerting Protocol](http://en.wikipedia.org/wiki/Common_Alerting_Protocol) format.

Namely, a well-tested and easy-to-use Java library that supports:

- creation and parsing of feeds in the CAP format,
- validating of feeds against common CAP profiles.

Moreover, it includes a simple [web application](http://cap-validator.appspot.com/) to validate the correctness of CAP messages.

About

The CAP Library is designed to support CAP versions 1.0, 1.1, and 1.2. There are classes that can parse XML CAP messages as well as easily create new messages and write them to XML, JSON, (soon) ASN.1, and (soon) KML.

The main data structures are auto-generated from a Google protocol buffer implementation of the CAP spec in proto/cap.proto. Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data – think XML, but smaller, faster, and simpler.

The generated classes offer a clean API for creating and manipulating alert objects. The alert data structures are
EDXL Framework/Toolkit

- An Online Validator for each EDXL Specification
  - Example: https://cap-validator.appspot.com/
EDXL Framework/Toolkit

- Basic Graphical Interface Components for all EDXL Specifications
  - e.g. a Validator
  - e.g. a Form Builder to view/create/update individual messages
  - e.g. a Map View to geographically view messages
  - e.g. a Table View to view lists of elements in messages and lists of messages
EDXL Framework/Toolkit

- JSON Schema & Form Builder Example
  - JSON Schema: [http://json-schema.org](http://json-schema.org)
  - Form Builder: [https://mozilla-services.github.io/react-jsonschema-form/](https://mozilla-services.github.io/react-jsonschema-form/)
EDXL Framework/Toolkit

- CAP Validator MapView Example
EDXL Framework/Toolkit

- A set of Reference Implementations for all EDXL Specifications
  - Free proof-of-concept software implementations
  - Demonstrating effective and proper use of the specifications
  - Using the free Developer Tools
EDXL Framework/Toolkit

- Reference Implementations starting with CAP-DE
EDXL Framework/Toolkit

- CAP-DE Reference Implementation Start Page
EDXL Framework/Toolkit

- CAP-DE Ref. Imp. Alert Category Page
EDXL Framework/Toolkit

- CAP-DE Ref. Imp. Assemble Message Page
EDXL Framework/Toolkit

- CAP-DE Ref. Imp. Review Message Page
EDXL Framework/Toolkit

- CAP-DE Ref. Imp. Approve Message Page
Wrapping EDXL-CAP with EDXL-DE-v1.0

We will insert CAP Message (Payload) in Content Category
Wrapping EDXL-CAP with EDXL-DE-v1.0
Wrapping EDXL-CAP with EDXL-DE-v1.0

Insert CAP Message into <embeddedXMLContent> element
Wrapping EDXL-CAP with EDXL-DE-v1.0

CAP Message Starts Here—Inserted into body of <embeddedXMLContent> element

EDXL-DE-v1.0 Wrapper/Header Starts Here
Wrapping EDXL-CAP with EDXL-DE-v1.0

CAP Message Ends here

EDXL-DE-v1.0 Wrapper-Header Concludes here
Wrapping EDXL-CAP with EDXL-DE-v2.0

3 Data Categories in EDXL-DE-v2.0

We will insert CAP Message (Payload) in Content Category
Wrapping EDXL-CAP with EDXL-DE-v2.0
Wrapping EDXL-CAP with EDXL-DE-v2.0

Insert CAP Message into `<embeddedXMLContent>` element
Wrapping EDXL-CAP with EDXL-DE-v2.0

CAP Message Starts Here—Inserted into body of <embeddedXMLContent> element

EDXL-DE-v2.0 Wrapper/Header Starts Here
Wrapping EDXL-CAP with EDXL-DE-v2.0
EDXL Framework/Toolkit

- Tutorials for all EDXL Specification
  - No Samples available yet
- Examples for all EDXL Specifications
  - In addition to those included with EDXL Specifications
  - No new Samples available yet
EDXL Framework/Toolkit

- Automated JSON Schema Conversions of Normative XML Schema for all EDXL Specifications
EDXL Framework/Toolkit

- Custom JSON Schema Conversions of Normative XML Schema for all EDXL Specifications
  - To be constructed as we revise each specification
  - For use with the Form Builder
    - JSON Schema: http://json-schema.org

- JSON-LD Representations for all EDXL Specifications
  - For uses of Linked Data in web applications
  - To be constructed as revise each specification
EDXL Framework/Toolkit

- Relational (SQL) Database Schema for all EDXL Specifications in DDL format
  - Useful for Cross-Enterprise, Enterprise-wide SOA Applications
  - Basic SQL From Enterprise Architect based on XML Schema
EDXL Framework/Toolkit

- NoSQL (MongoDB) Document Store Database Schema
- Fits JSON High-Performance, High-Availability Applications
EDXL Framework/Toolkit

- Guidance for Emerging Technologies
  - Microservices Architecture using Docker-type Containers
    - Particularly effective for JSON-based Microservices using NoSQL Datastores
  - Protocol Buffers for programming language-agnostic inter-application messaging
    - Software developers use a variety of programming languages for specific components in modular patterns so a language-agnostic means of communication between components is needed
Conclusion

- What would Developers Actually Use?
  - If we build it, will they come?
  - What are we missing?
  - We need your help

- Can we find a Host for persisting the EDXL Framework/Toolkit?
  - OASIS supports the use of Public Open-Source GitHub Repositories and Subversion
  - Other Options?
Questions

Rex Brooks rexb@starbourne.com
Elysa Jones elysajones@yahoo.com
Jeff Waters jeffrywaters@gmail.com