EM Domain Adapter Approach: Guidance and Best Practices

February 2011
Emergency Management Domain
National Information Exchange Model

Table of Contents

Executive Summary

This document provides Emergency Management (EM) Domain policy with regard to use of Adapters and External standards. It explains the rationale and process in detail for the following basic set of rules:

1. If you use an approved external standard (Emergency Data Exchange Language [EDXL]-Common Alerting Protocol [CAP] for example) for an exchange in its entirety without change, you should continue to use it.
2. If you use a known Information Exchange Package Documentation (IEPD), you are within the rules. Just use it.
3. If you use all of an external standard or part of an external standard (i.e., borrow a concept), along with other standards or concepts (from the National Information Exchange Model [NIEM] where possible) in the same new exchange, you must document the exchange as an IEPD and use adapters for external standard content to provide NIEM conformance.
4. Where adapters are not present, you should create them according to Naming and Design Rules (NDR) precepts. They should be created at the actual structural level of re-use. In other words, if you reuse a substructure of an external standard, you should treat the entire substructure as one concept with one adapter for submission as a NIEM approved concept. You should not build an adapter for each element (unless the only item being reused is the individual element).
5. Required concepts that are not already in the NIEM and are not defined in approved external standards should be documented and submitted as new NIEM elements in accordance with the NDR and IEPD definition guidance without the use of adapters.

Introduction

The Emergency Management (EM) Domain aims to more effectively share emergency management data in emergency situations, and the adapter approach presents the guidance and best practices on how this information exchange incorporates external standards. The approach will focus on the background between the National Information Exchange Model (NIEM) and the Emergency Data Exchange Language (EDXL) messaging standards since EDXL standards are the most common external standards in use in the EM Domain. The best practices and guidance section will explain how the EM Domain will use adapters and what adapters currently exist. In the appendix, a frequency asked questions guide along with the NIEM Naming and Design Rules (NDR), additional web-based resources, further guidance on developing adapters and helpful definitions have been provided.

This document is focused on the adapter approach and does not outline information on the current domain content or future developments in the EM Domain. Additionally, through leveraging expertise and knowledge from both the emergency responder community and technical developer community to implement the best form of data communications; this document is intended for both technical and non-technical audiences.

Background

Effective information sharing is critical to the success of a coordinated emergency response. To support emergency response data communications, the Department of Homeland Security Directorate of Science and Technology and the Federal Emergency Management Agency (FEMA), Information Technology Enterprise Architecture Office (IT-EA) agreed to co-own the EM domain within the NIEM data model. NIEM is a data model designed to develop, disseminate and support enterprise-wide information exchange standards and processes.

The vision of the EM Domain is to enable jurisdictions to effectively share emergency management data daily and in emergency situations. Through increased interoperability, first responders, emergency managers and jurisdictions will be able to communicate more effectively with each other, which will provide all parties with a higher level of situational awareness, more resources for response, and the ability to mobilize the public with an alert or warning and make better decisions.

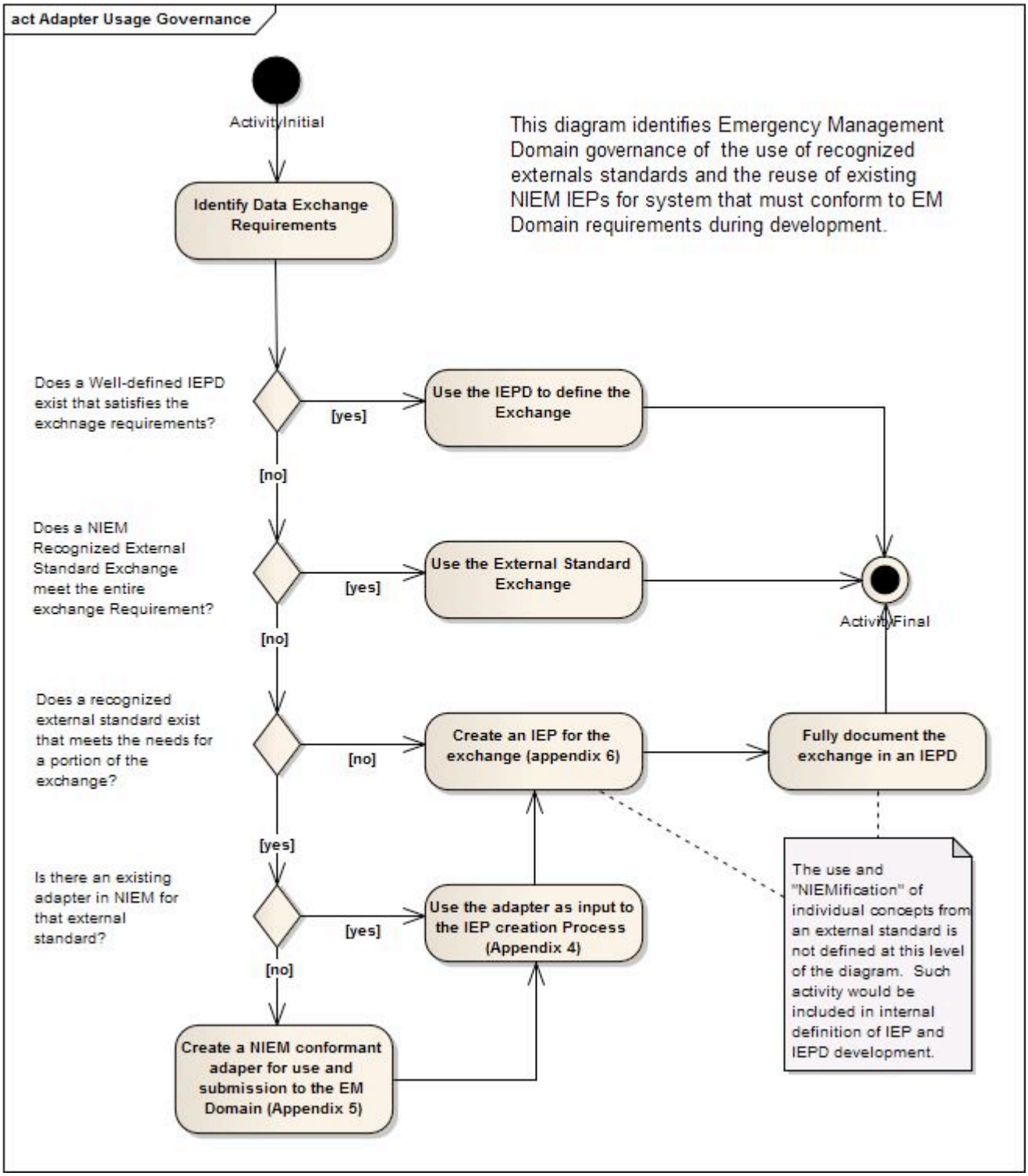
As the EM Domain evolved, the Domain incorporated the Emergency Data Exchange Language (EDXL) message standards. The EDXL standards, an Extensible Markup Language (XML)-based protocol, were developed by the Organization for the Advancement of Structured Information Standards (OASIS) to assist the emergency response community in sharing data seamlessly and securely while responding to an incident. The standards were created through a practitioner-driven, public-private partnership and enable information sharing capabilities between disparate emergency response software applications, systems, and devices. The goal of the EDXL suite of standards is to facilitate emergency information sharing and data exchange across the local, state, tribal, national and non-governmental organizations of different professions that provide emergency response and management services.[1]

Best Practices and Guidelines

The EM Domain leveraged subject matter expertise, NIEM guidance, and stakeholder input to create the best practices and guidelines for using adapter in the EM Domain[2]. Though this approach has many technical questions that need subject matter expertise, the EM Domain adapter approach consists of two main considerations. First, adapters and the use of external standards will continue to exist in the EM Domain. Since many entities already use external standards and the overall goal is to ensure as much interoperability between disparate systems, adapters for external standards will always be available and accepted in the EM Domain. Second, as one uses adapters, the guidelines listed below point to the high-level considerations and best practices that are encouraged in the Domain so concepts are used and created in the appropriate manner. The process image below shows the general guidelines.

[1] More information on the current EDXL suite of standards and standards in development can be found in Appendix 1.

[2] The EM Domain has based the adapter approach off of guidance on using external standards from the NIEM NDR. This information is provided in Appendix 2.



The process flow starts with an identification of the information that needs to be exchanged. Much success has been seen in the obtaining requirements for information sharing phase by using an industry best practice requirements methodology. The next step in the process includes checking if an external standard or an Information Exchange Package Documentation (IEPD) already exists that can fulfill the information sharing requirements. If it does exist, the adapter approach recommends using the standard or existing IEPD. If an external standard or

IEPD does not exist, one should either use existing concepts where appropriate or develop new concepts. More details for each of these steps are listed below.

Using Adapters in the EM Domain

If through the adapter approach process flow there are external standards that do exist for the specific information sharing need, the NIEM NDR process lays out the approach for using adapters for reusing external standards. The use of adapters must conform to rules specified in the NIEM NDR. Currently in NIEM, adapters exist for the following EDXL standards:

1. CAP v1.1
2. EDXL DE v1.0
3. EDXL HAVE v1.0

If an adapter does not currently exist, adapters can be created for an entire Standard or for particular concepts (see discussion below) used in the Standard. When a new external standard is incorporated in NIEM, a set of concepts (specific) to the standard must also be created. This will allow reuse. In developing or creating new adapters, the EM Domain does not differentiate between types of external schemas (ex: adapters for content payload schemas will be similar to adapters for distribution/content routing schemas)

Reusing Concepts from external Standards in creating IEPDs

While creating IEPDs, concepts from external standards should be used, if applicable. However, the concepts *should* be used in the same context as its use in the external standard. This requires the development of a concept context model that defines concepts, their meanings and relationships - in other words, it describes the context for using a particular concept.

The concept should define and describe a particular business object or topic. As an example, the concept of 'Emergency Department' is a good example of a concept since it sufficiently describes a topic and includes multiples elements and attributes. A 'bed' as a concept is NOT a good example since it is at the element level and is vague in definition and use. Concepts *should* be at a 'structure' level, rather than at an 'element' level; a complex type rather than a simple type. However in some cases, simple type concepts may be necessary. Additionally, concepts *should* have a high potential for reuse across multiple domains

Conclusion

As the EM Domain continues to develop, the domain managers view the adapter approach as the best way to encourage reuse, interoperability and compatibility in the EM Domain. This approach is a “living document” that will change over time. Moving forward, the intent is for the document to be shared with other domains in NIEM to gather their knowledge, input, and experience in working with other external Standards. As the input from other domains is incorporated, the goal is for the adapter approach to eventually be applied universally across all NIEM domains.

As individuals begin to follow the process flow and guidelines for using adapters or reusing concepts in the other domains, the EM Domain stewards would appreciate any feedback or input as adapters or IEPDs are used and developed. Please contact Denis Gusty at Denis.Gusty@dhs.gov or Scott Shoup at Christopher.Shoup@dhs.gov with any questions or comments.

Appendix 1: Frequently Asked Questions

1. What is the National Information Exchange Model (NIEM)?

The National Information Exchange Model (NIEM) is a joint Department of Justice (DOJ) and DHS program, initiated on February 28, 2005, created to promote standardization of information exchange for cross-jurisdictional information sharing.

As stated in the NIEM User Guide (Volume 1; 19 August, 2008), the primary goal for the NIEM is to develop a common set of reusable, extensible XML data components that can be re-used and combined in documents, transactions, and messages that are consistently structured to support interoperability between systems. NIEM is a data layer "dictionary" and toolset, and intentionally does not address all of the necessary technologies needed for information sharing.

The NIEM provides the tools used across DHS and other agencies for enabling interoperability at the data layer within and across systems supporting information sharing, while preserving investments in current technology and optimizing new technology development. The NIEM Program Management Office (PMO) has three current objectives:

1. Bring stakeholders together to identify information sharing requirements for operational and emergency situations.
2. Maintain a National Data Model and Reference Vocabulary containing common and domain-specific data components that pertain to agency information needs to facilitate development of specific information exchanges.
3. Develop a common vocabulary and an online repository of re-usable exchanges to support information sharing.

The NIEM itself consists of two major components:

- A common XML-based language for information exchange that seeks to provide consistency of data definitions between agencies, and decrease the development time for information exchanges that use similar sets of data.
- A repeatable, reusable process – the IEPD Lifecycle – for business users to document information exchange requirements.

2. What is the Emergency Management Domain (EM Domain)?

NIEM is composed of two structural components:

1. Business "domains" such as Emergency Management, Justice and Immigration broadly reflect the agencies, units of government, operational functions, services, and information systems that are organized or affiliated to meet common objectives
2. "NIEM core" composed of data components agreed upon between two or more domains.

Supporting NIEM's vision, the EM Domain provides reusable, cross-profession emergency response and management data components supporting day to day and emergency information exchange development where an existing open standard is not available. The EM domain also provides NIEM user search and discovery of available EDXL standards for evaluation against exchange requirements, including the ability to extend or adapt EDXL standards using NIEM processes and tools.

3. What is the Emergency Data Exchange Language (EDXL)?

The Emergency Data Exchange Language (EDXL) is a suite of open public standards, free to use and international in scope, driven by cross-profession emergency practitioners through DHS sponsorship. EDXL enables emergency responders in the field and emergency operations personnel to seamlessly share critical information between multiple disparate systems during an emergency. EDXL standards are developed and governed by the international Standards Development Organization, OASIS. Essentially, each EDXL standard acts in place of the "IEPD" because it is agreed-upon and standardized to support a specific operational function. From public alerts and warnings to managing emergency resources, patients, and hospital availability, EDXL standards enable information exchange that is essential for effective response.

The approved EM-Domain external standards in use include:

Standard	**Date Adopted**	**Description**
Common Alerting Protocol (CAP) Version 1.1	October 2005	Enables the exchange of all-hazard emergency alerts, notifications, and public warnings, which can be disseminated simultaneously over many warning systems (e.g., computer systems, wireless systems, alarms, TV, radio, cable TV systems, cell phones, etc.)
Distribution Element (DE) Version 1.0	April 2006	A flexible message "envelope" to wrap and route any EDXL or other emergency information (XML and non-XML). "Address" the package in flexible ways to support intelligent routing by roles, geographic area, or keywords such as agency type (e.g., police, fire)
Hospital AVailability Exchange (HAVE) Version 1.0	November 2008	Enables the exchange of hospital status, capacity, and resource availability between medical and health organizations and emergency decision-makers routing patients to the proper facilities for care.
Resource Messaging (RM) Version 1.0	November 2008	A suite of 16 standard messages for exchange of emergency resource information (equipment, supplies, people, and teams), facilitating their use, deployment, and return in support of emergency preparedness, response, and recovery.

Standards in development include:

Standard	**Date Planned**	**Description**
Situation Reporting (SitRep)	Second Quarter, 2011	Enables the exchange of information about the current situation, the operational picture, and current resources in an actionable form, allowing for decision making before, during, and after emergencies and disasters
Tracking of Emergency Patients	Initiate OASIS process Feb. 2011	Enables the exchange of emergency patient and EMS tracking information from point of encounter through admission at a definitive care facility; to increase the

(TEP)		effectiveness of emergency medical management, patient tracking and care, and family notification
Tracking of Emergency Clients (TEC)	Submit from practitioner process to OASIS First Quarter, 2012	Expands TEP scope for tracking and support across the general population. TEC is aimed at more effective evacuation and services management, evacuee tracking, regulation, family notification, and use of evacuation and shelter assets.

4. What is the difference between EDXL and the EM Domain?

EDXL and NIEM are complementary approaches to meet the needs of the broad interoperability requirements.

At the most basic level, EDXL pre-defines specific message structures (standards), governing information exchange specifically around common, core operational functions (like "resource management", and "alerts and warnings"); functions which are broadly required across professions and jurisdictions that respond to emergencies and disasters Each standard is essentially "defined once" by an international Standards Development Organization (SDO), and intended to be implemented consistently to enable broad interoperability in support of that core function. However, EDXL standards only address a finite set of data exchange functional areas.

Though not an SDO, NIEM with its EM domain on the other hand provides a structured environment of re-usable data components, tools and process allowing each group of exchange partners to define their own message structures for exchange of information. Each set of partners define messages structure to their specific requirements, which may or may not be re-used by others.

As stated earlier, each EDXL standard essentially *provides* the "IEPD" standardized to support the specific operational function or requirement. Because it is an existing open standard message(s), exchange partners coordinate *implementation* of their exchanges applying that standard rather than defining the exchange structure.

In contrast to a pre-defined standard message structure, the NIEM environment, tools and EM Domain provide commonly-defined and reusable data components (or "data dictionary") used to define a message structure or exchange. NIEM and the EM domain are used:

1. To assist NIEM user search and discovery of available EDXL standards for evaluation against exchange requirements
2. To support NIEM user extension of an EDXL standard to meet unique exchange requirements.
3. To define an exchange (an IEPD) between exchange partners where an existing open standard is not available. However, that exchange is only "standard" between those specific exchange partners. It is conceivable that two groups *independently* developing IEPDs for the *same purpose* will create *incompatible* IEPDs.
4. As a potential "data dictionary" of elements re-used within a new or versioned EDXL standard

However, NIEM and the EM domain are not intended to be used to "re-develop" any existing EDXL standard. The result would be incompatible and "stovepipe" versions of the "same" standard, breaking interoperability. The structure of a messaging standard is more important than the semantics of the elements (or "tags") used as placeholders for data. However, in a data

dictionary such as NIEM, emphasis is placed on the specific naming and definition of each data element for use as a "common language" within systems and between users. In addition, NIEM IEPD's may become candidates for open, international standards.

5. Where can I find the EDXL standards?

Normative EDXL standards can be found on the OASIS website at www.oasis-open.org. In addition, a number of other resources are also available to provide access to or assist implementation of EDXL standards (see appendix 3 "Resources").

6. What data elements exist in the EM Domain?

The EM domain of NIEM currently contains data elements and attributes extracted from the following OASIS-published message standards: 1- Common Alerting Protocol (CAP v1.1); 2- Distribution Element (DE v1.0); and 3- Hospital AVailability Exchange (HAVE v1.0). These standards are maintained independent of NIEM, since EDXL functions as a stand-alone suite of messaging standards, and EM domain elements should not be used to "re-develop" any EDXL standard. This will be addressed in future governance documentation.

7. Why aren't the data elements in the EDXL standards NIEM compliant?

The EDXL standards referenced above were published before the NIEM Naming and Design Rules (NDR) and the EM Domain were established. Since EDXL is a practitioner driven standards development process, it follows the OASIS Emergency Management Committee process for the creation and change to all message structures and data elements.

8. Can I use EDXL standards and EM Domain data elements together?

Yes; EM Domain adapters allow you to use an external schema within the domain.

9. Can I use other standards and EM Domain data elements together?

Yes; EM Domain adapters allow you to use an external schema within the domain.

10. Is EDXL going away because of the EM Domain?

No, EDXL standards will not go away and new EDXL standards as well as new versions of existing EDXL standards continue to be developed. The EM domain complements and extends EDXL value by providing a set of reusable elements, specific to the emergency / disaster support domain, which may be used to develop IEPD's where no existing EDXL standard is available – or to extend an EDXL standard to meet unique requirements.

11. How can I use NIEM with EDXL standards?

Anyone can use NIEM and EDXL standards by using an adapter to bring EDXL standards into the EM Domain. NIEM users may search and discover available EDXL standards for evaluation against exchange requirements. An EDXL standard may be applied in whole to meet those requirements, or the EDXL standard may be extended using NIEM adaptor processes and tools.

12. What is an adapter?

An adapter is a NIEM-conformant type that adapts "external components" for use within NIEM. "External components" are defined as any XML compliant structure that was and is not defined within NIEM using the NIEM NDR. External components most frequently include structures from Standards Development Organizations, in whole or in part.

An adapter type creates a new class of object within NIEM that represents a single concept composed of one or more external components. A NIEM-conformant schema defines an adapter type. [Rule 7-63] (REF, EXT) – that is an adaptor type makes a non-conformant schema a NIEM-conformant one. It does this by adding attributes at the adapter level that are needed for NIEM conformance without changing the referenced content in any way. Adapters are housed and maintained in the NIEM core and are not domain specific. An adapter can contain:

a) An entire external schema (assuming it wraps the root element type)
b) A complex component element type from a schema
c) A single type from the schema.

13. What is an Information Exchange Package Documentation (IEPD)?

An IEPD itself is a set of documentation which forms a specification for a data exchange, and defines a particular data exchange between a group of exchange partners. For example, there is an IEPD that defines the information content and structure for an Amber Alert, a bulletin or message sent by law enforcement agencies to announce the suspected abduction of a child. Generally when an IEPD is referred to in documentation, this does not refer to a particular IEPD, but instead to a prescribed template for all IEPDs that define content, structure, format, and packaging of a message structure for information exchange.

14. The adapter I need doesn't exist, how do I develop an adapter?

The EM Domain is a user driven community. Everyone is able to follow the NDR and create their own adapters. Please feel free to reach out to the National Information Sharing Standards Help Desk or refer to Appendix 5 of this document for additional guidance on creating an adapter.

15. Do I need to get approval from someone before developing an adapter?

To ensure IEPDs and adapters are NIEM conformant; they must go through the NIEM approval process. This promotes reuse and interoperability in the emergency management community. Please feel free to reach out to the National Information Sharing Standards Help Desk for additional guidance on the adapter approval process.

16. Who do I contact with comments on the EM Domain?

The EM Domain stewards would appreciate any feedback or input as adapters or IEPDs are used and developed. Please contact Denis Gusty at Denis.Gusty@dhs.gov or Scott Shoup at Christopher.Shoup@dhs.gov with any questions or comments.

Appendix 2: NIEM Naming and Design Rules

The EM Domain leveraged subject matter expertise, NIEM guidance, and stakeholder input to create the best practices and guidelines for using adapter in the EM Domain. The EM Domain has based the adapter approach off of guidance on using external standards from the NIEM Naming and Design Rules (NDR) as referenced in the NIEM NDR citations below.

7.7 Using External Schemas
There are a variety of commonly used standards that are represented in XML Schema. Such schemas are generally not NIEM-conformant. NIEM-conformant schemas may reference components defined by these external schemas. NIEM-conformant components may be constructed from schema components that are not NIEM-conformant.
[Definition: external schema]
An **external schema** is any schema that is not a supporting schema and that is not NIEM-conformant.
Note that the supporting schemas structures and appinfo are nonconformant because they define the fundamental framework on which NIEM is built. However, they are not considered external schemas because of their supporting nature and are thus excluded from this definition.
NIEM-conformant schemas may work with external schemas by creating external adapter types. A single method is used to integrate external components into NIEM-conformant schemas: NIEM-conformant types are constructed from the external components.

Figure 7-10: Use of external components to create a NIEM-conformant type
Components defined by external schemas are called *external components*. A NIEM-conformant type may use external components in a specific way: to construct a NIEM-conformant type from external components. The goal in this method is to preserve as a single unit a set of data that embodies a single *concept* from an external standard.
For example, a NIEM-conformant type may be created to represent a bibliographic reference from an external standard. Such an object may be composed of multiple elements and types from the external standard. These pieces are put together to form a single NIEM-conformant type. For example, an element representing an author, a book, and a publisher may be included in a single bibliographic entry. A NIEM-conformant type built from these components may be used as any other NIEM-conformant type. That is, elements may be constructed from such a type, and those elements are fully NIEM-conformant.
To construct such a component, a NIEM-conformant schema must first import an external schema.
[Rule 7-61] (REF, EXT)
Within the schema, an element xsd:import that imports a namespace defined by an external schema MUST have the application information appinfo:ConformantIndicator, with a value of false.
Rationale
Knowledge of the conformance of an imported schema allows processors to understand the semantics of referenced components, without additional processing. Namespaces imported into NIEM-conformant schemas are assumed to be conformant unless otherwise indicated.
[Rule 7-62] (REF, EXT)
Within the schema, an element xsd:import that imports a namespace defined by an external schema MUST be a documented component.
Rationale
A NIEM-conformant schema has well-known documentation points. Therefore, a schema that imports a NIEM-conformant namespace need not provide additional documentation. However, when an external schema is imported, appropriate documentation must be provided at the point

of import because documentation associated with external schemas is undefined and variable. In this particular case, documentation of external schemas is required at their point of use in NIEM.

[Definition: adapter type]

An **adapter type** is a NIEM-conformant type that adapts external components for use within NIEM. An adapter type creates a new class of object that embodies a single concept composed of external components. A NIEM-conformant schema defines an adapter type.

[Rule 7-63] (REF, EXT)

Within the schema, an adapter type MUST have application information appinfo:ExternalAdapterTypeIndicator with a value of true. A type that is not an adapter type SHALL NOT contain that indicator.

Rationale

This rule flags as external adapters those types that may contain external content. This allows for easier processing.

[Rule 7-64] (REF, SUB, EXT)

Within the schema, an adapter type MUST be an immediate extension of type structures:ComplexObjectType.

Rationale

The adapter type must contain the content defined for any NIEM component. The type structures:ComplexObjectType provides such content

[Rule 7-65] (REF, SUB, EXT)

Within the schema, an adapter type MUST be composed of only elements and attributes from an external standard.

Rationale

An adapter type should contain the information from an external standard to express a complete concept. This expression should be composed of content entirely from an external schema. Most likely, the external schema will be based on an external standard with its own legacy support. In the case of an external expression that is in the form of model groups, attribute groups, or types, additional elements and type components may be created in an external schema, and the adapter type may use those components.

[Rule 7-66] (REF, EXT)

Within the schema, an element reference used in an adapter type definition MUST be a documented component.

[Rule 7-67] (REF, EXT)

Within the schema, an attribute reference used in an adapter type definition MUST be a documented component.

Rationale

In normal (conformant) type definition, a reference to an attribute or element is a reference to a documented component. Within an adapter type, the references to the attributes and elements being adapted are references to undocumented components. These components must be documented to provide comprehensibility and interoperability. Since documentation made available by nonconformant schemas is undefined and variable, documentation of these components is required at their point of use, within the conformant schema.

[Rule 7-68] (REF, SUB, EXT)

Within the schema, an adapter type MUST NOT be extended or restricted.

Rationale

Adapter types are meant to stand alone; each type expresses a single concept from an external schema, and adapter types are maintained in separate schemas that only contain adapter types. In this way, processors may easily switch modes, processing NIEM-conformant content in one way, and external content in another.

Appendix 3: Additional Resources

Integrated Justice Information Systems (IJIS)
IJIS brings together industry and government in an effort to improve national security and promote effective information sharing across all levels of the justice, public safety, and homeland security communities.
http://www.ijis.org/

Naming and Design Rules (NDR)
The NDR document specifies XML Schema documents for use with the National Information Exchange Model (NIEM).
http://www.niem.gov/pdf/NIEM-NDR-1-3.pdf

National Information Exchange Model (NIEM) Website
The website includes information and updates related to NIEM and its domains.
www.niem.gov

> *List of NIEM 2.1 EM Domain Schemas*
> http://www.niem.gov/niem/NIEM-2.1-schemas.html

National Information Sharing Standards Help Desk
The DOJ Office of Justice Programs and the Global Training and Technical Assistance Committee deployed the National Information Sharing Standards Help Desk, which includes a knowledgebase and live technical support. This new interactive resource provides an enhanced level of help to individuals and agencies implementing the Global Justice XML Data Model and NIEM.
http://it.ojp.gov/default.aspx?area=implementationAssistance&page=1117

Organization for the Advancement of Structured Information Standards (OASIS) Website
OASIS is a not-for-profit consortium that drives the development, convergence and adoption of open standards for the global information society.
http://www.oasis-open.org/

Approved standards include:

> *Common Alerting Protocol (CAP) Version 1.1*
> http://www.oasis-open.org/committees/download.php/15135/emergency-CAPv1.1-Corrected_DOM.pdf
>
> *Distribution Element (DE) Version 1.0*
> http://docs.oasis-open.org/emergency/EDXL-DE/V1.0
>
> *Hospital Availability Exchange (HAVE) Version 1.0*
> http://docs.oasis-open.org/emergency/edxl-have/pr05/emergency_edxl_have-1.0-spec-pr05.pdf
>
> *Resource Messaging (RM) Version 1.0*
> http://docs.oasis-open.org/emergency/edxl-rm/v1.0/cd01/EDXL-RM-SPEC-V1.0.pdf

www.edxlsharp.codeplex.com

http://openedxlcapde.sourceforge.net/

Appendix 4: How to Create an IEP with an Adapter from the EM Domain

NIEM Adapters are accepted by NIEM as "approved external standards." So you do not violate the NIEM requirements by using them, provided you use them as-is, in their entirety. If you use individual elements (or a subset of elements) from an Adapter schema in a way that does not validate against the adapter schema, you are actually violating both the external adapter and NIEM unless you document the use of those elements using the formal NIEM Information Exchange Package Documentation (IEPD) methodology as defined at niem.gov.

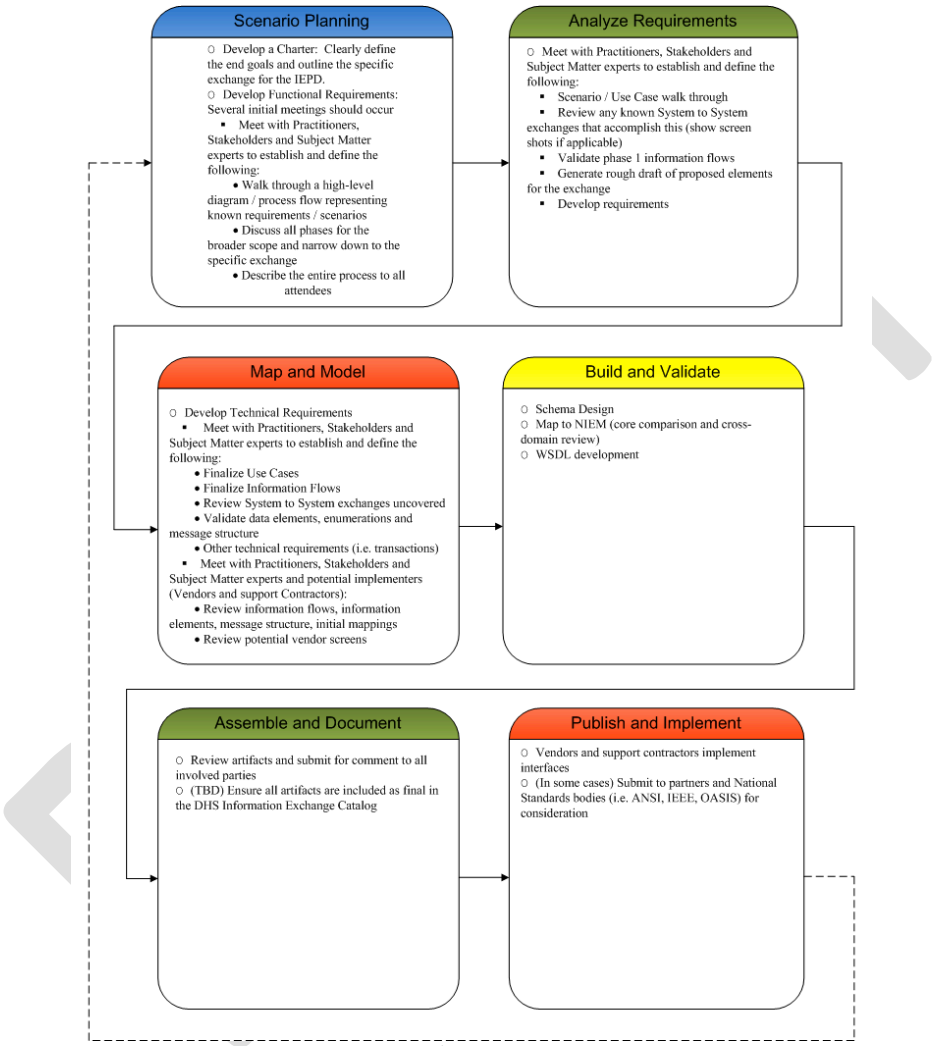


Figure 1

In order to use a system that uses EDXL- CAP, EDXL-DE, EDXL-RM and/or EDXL-HAVE, go ahead. You are within the terms of the grant language. But if you modify (aka "improve") the standards in any way, you must go through a formal IEPD process.

The following basic guidelines apply to all adapters in the NIEM EM Domain.

- Follow the process of Figure 3 for IEPD development

- At "Build and Validate" section of IEPD development create the IEP (either XML message or Schema for messages) and include the desired adapter in your namespace definition.
    - Example: *xmlns:h=niem2.1:niem:edxl-have:2.1*
- Determine if you need to use all elements and attributes of the external standard in your message.
    - You must use all required elements and attributes.
    - Extensions and modifications to elements or attributes must validate against the adapter schema.
    - Additional elements and attributes that are desired must be outside of the adapter call and defined with either a NIEM namespace or developed utilizing the NIEM Naming and Design Rules and should be:
        - Submitted to the EM Domain in the form of an Additional Content Request if the element is internally deemed to be of potential use to multiple message structures.
        - Documented in the form of an IEPD to include long and short definitions.

In addition, you may wish to further define your adapter call within the IEP by "profiling" the adapter to restrict usage of optional elements and attributes or defining specific content for list usage within the adapter. The use of the internal profile must be defined in the IEPD and must adhere to the following guidelines:

- The Adapter Profile **must**
    - Comply with the original technical specifications of the Adapter
    - Validate against the adapter schema
    - Use all required elements and attributes defined by the Adapter
    - Not change attributes for required elements.
- The Adapter Profile **may**
    - Further restrain the adapter (may be thought of as a "*constraint schema"* against the adapter)
    - Add to required element definitions (*only to extend or interpret the definition*)
    - Limit size of required elements
    - Exclude optional elements
    - Use optional elements in a specific way – as defined within the IEPD

Appendix 5: How to Create and Submit an Adapter for the EM Domain

The use of adapters is described in section 7-7 of the NIEM NDR[3]. Serious users of NIEM should read the NDR in its entirety. The final point on this discussion is very simple. You do not incorporate an external structure into NIEM, but you can define a NIEM conformant adapter in your IEPD as a container for that external structure that will allow you to import the external structure "as-is."

In order to submit an adapter to be reviewed for inclusion in the NIEM EM Domain you will need the complete IEPD[4] including to IEP artifacts; the Adapter and the Schema representing the external content desired. This information should be submitted to the NIEM EM Domain Governance Board. The instructions for the creation of these artifacts are defined below:

The NIEM NDR contains the following:

> *There are a variety of commonly-used standards that are represented in XML Schema. Such schemas are generally not NIEM-conformant. NIEM-conformant schemas may reference components defined by these external schemas. NIEM-conformant components may be constructed from non-NIEM schema components.*
>
> ***[Definition: external schema]***
>
> > *An **external schema** is any non-supporting schema that is not NIEM-conformant. Note that the supporting schemas structures and appinfo are non-conformant because they define the fundamental framework on which NIEM is built. However, they are not considered external schemas because of their supporting nature, and are thus excluded from this definition.*
>
> *NIEM-conformant schemas may work with external schemas by creating external adapter types.*
>
> *A single method is used to integrate external components into NIEM-conformant schemas: NIEM-conformant types are constructed from the external components.*
> ***Use of external components to create a NIEM-conformant type***
>
> *Components defined by external schemas are called external components. External components may be used by a NIEM-conformant type in a specific way: to construct a NIEM-conformant type from external components. The goal in this method is to preserve as a single unit a set of data that embodies a single concept from an external standard.*
>
> *For example, a NIEM-conformant type may be created to represent a bibliographic reference from an external standard. Such an object may be composed of multiple elements and types from the external standard. These pieces are put together to form a single NIEM-conformant type. For example, an element representing an author, a book, and a publisher may be included in a single bibliographic entry. A NIEM-conformant type built from these components may be used as any other NIEM conformant type. That is, elements may be constructed from such a type, and those elements are fully NIEM-conformant.*

[3] See Appendix 3 for link to the NIEM NDR
[4] See Appendix 3 for a link to the NIEM IEPD Lifecycle (http://www.niem.gov/iepdLifecycle.php)

NIEM NDR Rules and Rational

NIEM has established the following rules for incorporating Adapters in the NIEM NDR:

NIEM NDR Rule	Rational
7-61: Within a NIEM-conformant schema, an element xsd:import that imports a namespace defined by an external schema MUST have the application information appinfo:ConformantIndicator, with a value of false.	Knowledge of the conformance of an imported schema allows processors to understand the semantics of referenced components, without additional processing. Namespaces imported into NIEM-conformant schemas are assumed to be conformant, unless otherwise indicated.
7-62: Within the schema, an element xsd:import that imports a namespace defined by an external schema MUST be a documented component.	A NIEM-conformant schema has well-known documentation points. Therefore, a schema that imports a NIEM-conformant namespace need not provide additional documentation. However, when an external schema is imported, appropriate documentation must be provided at the point of import, because documentation associated with external schemas is undefined and variable. In this particular case, documentation of external schemas is required at their point of use in NIEM.
7-63: Within the schema, an adapter type MUST have application information appinfo:ExternalAdapterTypeIndicator with a value of true. A type that is not an adapter type SHALL NOT contain that indicator.	This rule flags as external adapters those types that may contain external content. This allows for easier processing.
7-64: Within the schema, an adapter type MUST be an immediate extension of type structures:ComplexObjectType.	The adapter type must contain the content defined for any NIEM component. The type structures:ComplexObjectType provides such content.
7-65: Within the schema, an adapter type MUST be composed of only elements and attributes from an external standard.	An adapter type should contain the information from an external standard to express a complete concept. This expression should be composed of content entirely from an external schema. Most likely, the external schema will be based on an external standard with its own legacy support. In the case of an external expression that is in the form of model groups, attribute groups, or types, additional elements and type components may be created in an external schema, and the adapter type may use those components.

NIEM NDR Rule	Rational
7-66: Within the schema, an element reference used in an adapter type definition MUST be a documented component. **7-67:** Within the schema, an attribute reference used in an adapter type definition MUST be a documented component.	In normal (conformant) type definition, a reference to an attribute or element is a reference to a documented component. Within an adapter type, the references to the attributes and elements being adapted are references to undocumented components. These components must be documented to provide comprehensibility and interoperability. Since documentation made available by nonconformant schemas is undefined and variable, documentation of these components is required at their point of use, within the conformant schema.
7-68: Within the schema, an adapter type MUST NOT be extended or restricted.	Adapter types are meant to stand alone; each type expresses a single concept from an external schema, and adapter types are maintained in separate schemas that only contain adapter types. In this way, processors may easily switch modes, processing NIEM-conformant content in one way, and external content in another.

To create an adapter to be considered for inclusion in the NIEM EM Domain you must provide an IEPD as defined by NIEM[5]. In addition you will need 2 IEP artifacts as defined below:

1. **The Adapter** – The actual adapter will be placed in the EM domain model under the /niem/*adapter name*/*niem version*/*adapter name.xsd*. In our example below the EXL-DE is placed in /niem/edxl-de/2.1/edxl-de.xsd

   Note the compliance with Rules:
   7-61 (xsd:import and i:ConformantIndicator),
   7-62 (<xsd:import schemaLocation="../../external/de/1.0/edxl-de.xsd" namespace="urn:oasis:names:tc:emergency:EDXL:DE:1.0">),
   7-63 (xsd:appinfo),
   7-64 (xsd:extension base="s:ComplexObjectType") and
   7-66 and 7-67 (xsd:documentation).
   It does not break the rules of 7-65 and 7-68.

Adapter example (EDXL-DE)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://niem.gov/niem/edxl-de/2.1" version="1"
xmlns:s="http://niem.gov/niem/structures/2.0"
xmlns:de="urn:oasis:names:tc:emergency:EDXL:DE:1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:edxl-
de="http://niem.gov/niem/edxl-de/2.1" xmlns:i="http://niem.gov/niem/appinfo/2.0">
  <xsd:annotation>
    <xsd:documentation>Distribution Element</xsd:documentation>
```

[5] See Appendix 3 for a link to the NIEM IEPD Lifecycle (http://www.niem.gov/iepdLifecycle.php)

```
    <xsd:appinfo>
      <i:ConformantIndicator>true</i:ConformantIndicator>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:import schemaLocation="../../structures/2.0/structures.xsd"
namespace="http://niem.gov/niem/structures/2.0"/>
  <xsd:import schemaLocation="../../appinfo/2.0/appinfo.xsd"
namespace="http://niem.gov/niem/appinfo/2.0"/>
  <xsd:import schemaLocation="../../external/de/1.0/edxl-de.xsd"
namespace="urn:oasis:names:tc:emergency:EDXL:DE:1.0">
    <xsd:annotation>
      <xsd:documentation>This Distribution Element specification describes a standard message
distribution framework for data sharing among emergency information systems using the XML-
based Emergency Data Exchange Language (EDXL). This format may be used over any data
transmission system, including but not limited to the SOAP HTTP
binding.</xsd:documentation>
      <xsd:appinfo>
        <i:ConformantIndicator>false</i:ConformantIndicator>
      </xsd:appinfo>
    </xsd:annotation>
  </xsd:import>
  <xsd:complexType name="DistributionElementAdapterType">
    <xsd:annotation>
      <xsd:documentation>A data type for a standard message distribution framework for data
sharing among emergency information systems using the XML-based Emergency Data
Exchange Language (EDXL).</xsd:documentation>
      <xsd:appinfo>
        <i:Base i:namespace="http://niem.gov/niem/structures/2.0" i:name="Object"/>
        <i:ExternalAdapterTypeIndicator>true</i:ExternalAdapterTypeIndicator>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="s:ComplexObjectType">
        <xsd:sequence>
          <xsd:element ref="de:EDXLDistribution" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="DistributionElementAdapter" type="edxl
de:DistributionElementAdapterType" nillable="true">
    <xsd:annotation>
      <xsd:documentation>A standard message distribution framework for data sharing among
emergency information systems using the XML-based Emergency Data Exchange Language
(EDXL).</xsd:documentation>
    </xsd:annotation>
  </xsd:element>
</xsd:schema>
```

2. The Schema of the standard

Although you have namespaced the schema in the adapter (in our example with a

urn to OASIS) a local reference was provided to ease message/exchange development and validation (<xsd:import schemaLocation="../../external/de/1.0/edxl-de.xsd" namespace="urn:oasis:names:tc:emergency:EDXL:DE:1.0">).

The Local Schema must be identical to the referenced standard and will be placed in /niem/*external/ adapter name/standard version/adapter name.xsd*

External Schema example - EDXL DE

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="urn:oasis:names:tc:emergency:EDXL:DE:1.0"
targetNamespace="urn:oasis:names:tc:emergency:EDXL:DE:1.0"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0CD">
  <xsd:element name="EDXLDistribution">
      <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="distributionID" type="xsd:string"/>
                <xsd:element name="senderID" type="xsd:string"/>
                <xsd:element name="dateTimeSent" type="xsd:dateTime"/>
                <xsd:element name="distributionStatus" type="statusValues"/>
                <xsd:element name="distributionType" type="typeValues"/>
                <xsd:element name="combinedConfidentiality" type="xsd:string"/>
                <xsd:element name="language" type="xsd:string" minOccurs="0"/>
                <xsd:element name="senderRole" type="valueListType" minOccurs="0"
                  maxOccurs="unbounded"/>
                <xsd:element name="recipientRole" type="valueListType" minOccurs="0"
                  maxOccurs="unbounded"/>
                <xsd:element name="keyword" type="valueListType" minOccurs="0"
                  maxOccurs="unbounded"/>
                 <xsd:element name="distributionReference" type="xsd:string"
                  minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="explicitAddress" type="valueSchemeType"
                  minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="targetArea" type="targetAreaType" minOccurs="0"
                  maxOccurs="unbounded"/>
                <xsd:element name="contentObject" type="contentObjectType"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>
  </xsd:element>
   <xsd:annotation/>
   <xsd:annotation/>
  <xsd:complexType name="contentObjectType">
      <xsd:sequence>
            <xsd:element name="contentDescription" type="xsd:string" minOccurs="0"/>
            <xsd:element name="contentKeyword" type="valueListType" minOccurs="0"
              maxOccurs="unbounded"/>
            <xsd:element name="incidentID" type="xsd:string" minOccurs="0"/>
            <xsd:element name="incidentDescription" type="xsd:string" minOccurs="0"/>
            <xsd:element name="originatorRole" type="valueListType" minOccurs="0"
```

```
                maxOccurs="unbounded"/>
            <xsd:element name="consumerRole" type="valueListType" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="confidentiality" type="xsd:string" minOccurs="0"/>
            <xsd:choice>
                <xsd:element name="nonXMLContent" type="nonXMLContentType"/>
                <xsd:element name="xmlContent" type="xmlContentType"/>
            </xsd:choice>
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
                maxOccurs="unbounded" />
    </xsd:sequence>
</xsd:complexType>
 <xsd:complexType name="nonXMLContentType">
    <xsd:sequence>
            <xsd:element name="mimeType" type="xsd:string"/>
            <xsd:element name="size" type="xsd:integer" minOccurs="0"/>
            <xsd:element name="digest" type="xsd:string" minOccurs="0"/>
            <xsd:element name="uri" type="xsd:anyURI" minOccurs="0"/>
            <xsd:element name="contentData" type="xsd:base64Binary" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
 <xsd:complexType name="xmlContentType">
    <xsd:sequence>
            <xsd:element name="keyXMLContent" type="anyXMLType" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="embeddedXMLContent" type="anyXMLType"
                minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="anyXMLType">
    <xsd:sequence>
            <xsd:any namespace="##other" processContents="lax"
                maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
<xsd:complexType name="valueListType">
    <xsd:sequence>
            <xsd:element name="valueListUrn" type="xsd:string" />
            <xsd:element name="value" type="xsd:string" maxOccurs="unbounded"/>
            </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="valueSchemeType">
    <xsd:sequence>
            <xsd:element name="explicitAddressScheme" type="xsd:string"/>
            <xsd:element name="explicitAddressValue" type="xsd:string"
                maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="targetAreaType">
    <xsd:sequence>
```

```
            <xsd:element name="circle" type="xsd:string" minOccurs="0"
              maxOccurs="unbounded"/>
            <xsd:element name="polygon" type="xsd:string" minOccurs="0"
              maxOccurs="unbounded"/>
            <xsd:element name="country" type="xsd:string" minOccurs="0"
              maxOccurs="unbounded"/>
            <xsd:element name="subdivision" type="xsd:string" minOccurs="0"
              maxOccurs="unbounded"/>
            <xsd:element name="locCodeUN" type="xsd:string" minOccurs="0"
              maxOccurs="unbounded"/>
      </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="statusValues">
      <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="Actual"/>
      <xsd:enumeration value="Exercise"/>
      <xsd:enumeration value="System"/>
      <xsd:enumeration value="Test"/>
      </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="typeValues">
      <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="Report"/>
            <xsd:enumeration value="Update"/>
            <xsd:enumeration value="Cancel"/>
            <xsd:enumeration value="Request"/>
            <xsd:enumeration value="Response"/>
            <xsd:enumeration value="Dispatch"/>
            <xsd:enumeration value="Ack"/>
            <xsd:enumeration value="Error"/>
            <xsd:enumeration value="SensorConfiguration"/>
            <xsd:enumeration value="SensorControl"/>
            <xsd:enumeration value="SensorStatus"/>
            <xsd:enumeration value="SensorDetection"/>
      </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Appendix 6: How to Create an IEP without an Adapter from the EM Domain

This appendix provides guidance on how to reuse XML components from XML schemas defined by External Standards that are not NIEM-conformant. The following diagram provides a process diagram, which is described and explained by the paragraphs following the diagram.

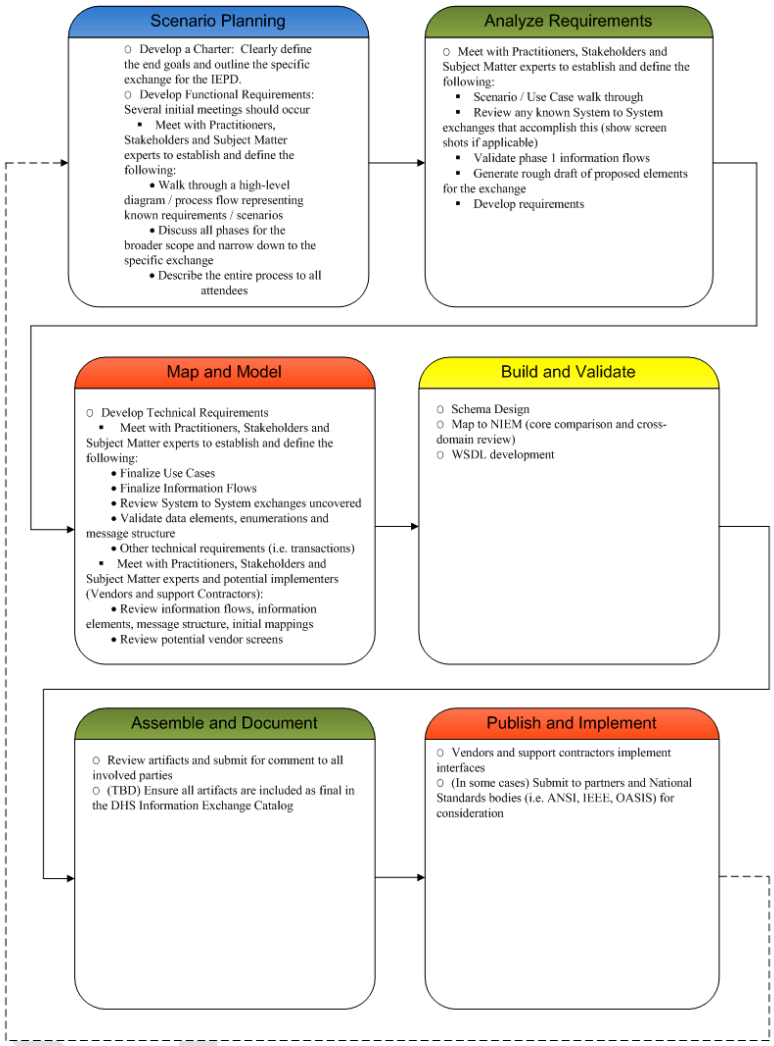


Figure 3. External Standard XML Component Reuse Process

Process starting conditions:

- A NIEM-conformant IEPD is to be developed or enhanced.
- One or more new components of the IEPD cannot be found in NIEM Core or Domains for reuse.
- There are one or more External Standards that have one or more components that can be reused to meet the requirements of the IEPD.

- It is desired to maintain the identity and correspondence between the component(s) reused from the External Standard(s) and the component(s) as it/they appear in the IEPD.
- The task now is to incorporate the components into the IEPD.

Process Start:

Do for each Component:

Step 1: Determine Component Scope

The scope will be one of:

- A single attribute
- A single element of atomic data type
- A single type with no subtrees
- A subtree collection of types
- A codelist

Step 2: Branch to Component Scope

Branch to the subprocess for the selected Component Scope

Step 3: Process Attribute Component

In this case the attribute will be converted into a NIEM-conformant attribute without changing its semantics.

Step 3.1: NIEMify the Name

Using the NIEM NDR rules, make the minimum necessary change to the attribute name to make it NIEM-conformant.

Step 3.2: Create Description

Copy any description text from the attribute declaration in the External Standard. Add text to note the External Standard from which the attribute came, and its name in the External Standard. Add any additional description text needed to clarify any potential ambiguities due to inadequate/incomplete description text in the External Standard.

Step 3.3: Declare Data Type

- If the attribute is declared in the External Standard as a standard W3c atomic type other than ID, IDREF, or IDREFS, use the same declaration.
- If the attribute is declared in the External Standard as an:
  - xs:ID, then declare it as s:id
  - xs:IDREF, then declare it as s:ref
  - xs:IDREFS, then declare it as s:metadata
- If the attribute is declared in the External Standard as a codelist, then, follow the process defined in Step 7 to convert the codelist type, and then declare the attribute as this codelist type.

Step 4: Process Element Component

In this case the element will be converted into a NIEM-conformant element without changing its semantics.

Step 4.1: NIEMify the Name

Using the NIEM NDR rules, make the minimum necessary change to the element name to make it NIEM-conformant.

Step 4.2: Create Description

Copy any description text from the element declaration in the External Standard. Add text to note the External Standard from which the element came, and its name in the External Standard. Add any additional description text needed to clarify any potential ambiguities due to inadequate/incomplete description text in the External Standard.

Step 4.3: Declare Data Type

- If the element is declared in the External Standard as a standard W3c atomic type, use the same declaration.
- If the element is declared in the External Standard as a Simple Type or Complex Type of simple content that is based on a standard W3c atomic type (for example by restriction or list), then use this same declaration.
- If the element is declared in the External Standard as a Complex Type, the follow the process defined in Step 5.
- If the element is declared in the External Standard as a codelist element, then follow the process defined in Step 7 to convert the codelist type, and then declare the element as this codelist type.
- If the element is declared in the External Standard by reference to a global element in the standard's namespace, then apply the applicable rule above to the global element.
- If the element is declared in the External Standard by reference to a global element in another schema, and that schema is not NIEM conformant, then apply the applicable rule above to the external element. If the external schema is NIEM conformant, then declare the element using the same reference.

Step 5: Process Type Component

In this case the Type will be converted into a NIEM-conformant Type without changing its semantics. The child elements and attributes will also be converted. Note in this case there are no child elements whose declarations are complex types.

Step 4.1: NIEMify the Name

Using the NIEM NDR rules, make the minimum necessary change to the Type name to make it NIEM-conformant.

Step 4.2: Create Description

Copy any description text from the Type declaration in the External Standard. Add text to note the External Standard from which the component came, and its name in the

External Standard. Add any additional description text needed to clarify any potential ambiguities due to inadequate/incomplete description text in the External Standard.

Step 4.3: Include appinfo

NIEM rules require the following to be included in Type definitions.

```
<xsd:appinfo>
  <i:Base i:namespace="http://niem.gov/niem/structures/2.0"
i:name="Object"/>
</xsd:appinfo>
```

Step 4.4: Process the Child Components

In this case, each child component will be either an attribute or an element declared as a simple type.

If component is an attribute, follow step 3.

If component is an element, follow step 4.

Step 6: Process Subtree Component

In this case only the component at the head of the subtree needs to be reusable and NIEM-conformant, and the components below the head of the subtree do not need to be NIEM conformant. The semantics of the subtree will be unchanged. In this case use the NIEM adapter type by applying it to the subtree head and wrap the whole substree.

Step 7: Process Codelist Component

In this case the component is a Simple Type with a restricted value set defined by a set of enumerations that defines the codelist.

Step 7.1: Process name and description

Follow steps 4.1, 4.2 and 4.3.

Step 7.2: Select Enumerations

Copy over the enumerations required, either all or a subset.

Step 7.3: Complete Enumeration Documentation

NIEM rules require that each enumeration include xsd:documentation. If this is missing from the source codelist, add the appropriate documentation.

Step 7.4: Add Complex Type

NIEM rules require that each codelist simple type have a companion complex type defined as s:ComplexObjectType. Add this type and provide the appropriate name and documentation text in accordance with the relevant NIEM rules. Insert the following into the type definition.

```
 <xsd:appinfo>
  <i:Base i:namespace="http://niem.gov/niem/structures/2.0" i:name="Object"/>
 </xsd:appinfo>
</xsd:annotation>
<xsd:simpleContent>
```

```
        <xsd:extension base="[insert qualified name of the codelist simple type]">
          <xsd:attributeGroup ref="s:SimpleObjectAttributeGroup"/>
        </xsd:extension>
      </xsd:simpleContent>
END Process
```


Appendix 7: Definition of Terms and Acronyms

recognized external standard – developed by a Standards Development Organization (e.g., IEEE, ANSI, OASIS, ISO, HL7, NFPA, and OGC)

structure – can stand alone (e.g., GML Where and CIQ)

CAP	Common Alerting Protocol
DE	Distribution Element
DHS	Department of Homeland Security
DOJ	Department of Justice
EDXL	Emergency Data Exchange Language
EM Domain	Emergency Management Domain
FEMA	Federal Emergency Management Agency
HAVE	Hospital Availability Exchange
IEP	Information Exchange Package
IEPD	Information Exchange Package Documentation
IT-EA	Information Technology Enterprise Architecture
NDR	Naming and Design Rules
NIEM	National Information Exchange Model
OASIS	Organization for the Advancement of Structured Information Standards
OIC	Office for Interoperability and Compatibility
PMO	Project Management Office
RM	Resource Messaging
S&T	Science and Technology Directorate
SDO	Standards Development Organization
SitRep	Situation Reporting
TEC	Tracking of Emergency Clients
TEP	Tracking of Emergency Patients
XML	Extensible Markup Language