August 8, 2011

MODEL PACKAGE DESCRIPTION SPECIFICATION

VERSION 1.0

URI: http://reference.niem.gov/niem/specification/model-package-description/1.0/

Change History

No.	Date	Reference: All, Page, Table, Figure, Paragraph	A = Add. M = Mod. D = Del.	Revised By	Change Description
1.0	8/8/2011	All	A	NTAC	Initial release

Contents

Figures

Tables

1 Introduction

This specification assumes familiarity with the National Information Exchange Model (NIEM), its basic concepts, architecture, processes, design rules, and general conformance rules. For novices, the recommended reading list includes:

- Introduction to the National Information Exchange Model **[NIEM-Intro]**
- NIEM Concept of Operations **[NIEM-ConOps]**
- NIEM Naming and Design Rules **[NIEM-NDR]**
- NIEM High-Level Version Architecture **[NIEM-HLVA]**
- NIEM High-Level Tool Architecture **[NIEM-HLTA]**
- NIEM Conformance **[NIEM-Conformance]**
- NIEM User Guide **[NIEM-UserGuide]**
- NIEM Business Information Exchange Components (BIEC) **[NIEM-BIEC]**
- NIEM Implementation Guidelines **[NIEM-Implementation]**

The foregoing NIEM documents are available at http://reference.niem.gov/niem/. See **[NIEM-Implementation]** for NIEM Implementation Guidelines.

Those knowledgeable of NIEM should be familiar with the **[NIEM-NDR], [NIEM-HLVA], [NIEM-Conformance],** and **[NIEM-BIEC]**.

This specification uses and is a peer to the *NIEM Naming and Design Rules (NDR)* **[NIEM-NDR]** and supersedes IEPD guidance previously published in *Requirements for a NIEM IEPD* **[NIEM-IEPD]** and the *NIEM User Guide* **[NIEM-UserGuide]**. The NIEM User Guide remains a good source for understanding the process of building Information Exchange Package Documentation (IEPD).

1.1 Background

Many fundamental concepts, processes, and products in the NIEM generally involve aggregating electronic files into logical sets that serve a specific purpose. Examples of such sets include, but in the future may not necessarily be limited to, a NIEM release, domain update, Information Exchange Package Documentation (IEPD), and Enterprise Information Exchange Model (EIEM). Each of these examples is a NIEM *Model Package Description (MPD).* Each MPD contains one and only one class of an *Information Exchange Model* (*IEM)*. An IEM is a set of defining XML schemas and the core of an MPD. Because these terms are closely related (one contains the other) and used in this specification, they are both formally defined now:

> Definition: **Information Exchange Model (IEM)** – One or more NIEM-conforming XML schemas that together specify the structure, semantics, and relationships of XML objects. These objects are consistent XML representations of information. Currently, five IEM classes exist in NIEM:
> (1) release, (2) core update, (3) domain update, (4) Information Exchange Package Documentation (IEPD), and (5) Enterprise Information Exchange Model (EIEM).

> Definition: **Model Package Description (MPD)** – An organized set of files that contains one and only one of the five classes of NIEM IEM, as well as supporting documentation and other artifacts. An MPD is self-documenting and provides sufficient normative and non-normative information to allow technical personnel to understand how to use and/or implement the IEM it contains. An MPD is packaged as a ZIP **[PK-ZIP]** file.

Another key NIEM concept that is used throughout this specification is *data component*.

> Definition: **data component** – An XML Schema type or attribute group definition; or an XML Schema element or attribute declaration.

An IEM is a normative specification for XML data components in the format of World Wide Web Consortium (W3C) XML Schema definition language **[W3-XML-SchemaDatatypes] [W3-XML-SchemaStructures]**. These schema files define the semantics and structure for data exchange instance documents in the W3C Extensible Markup Language (XML) **[W3-XML]**.

An IEM becomes an MPD when it has been properly packaged with the documentation and supplemental files required to understand how the IEM is to be used and implemented.

Because an MPD contains one and only one IEM, henceforth this specification will refer to an MPD by the name of the IEM class it contains. The reader should simply remember that an IEM is a set of related schemas, while an MPD is a complete package including IEM, documentation, and other artifacts.

MPD content design, development, and assembly may be difficult and time-consuming, especially if done manually. Software tools have been shown to significantly reduce the complexity of designing, constructing, changing, and managing MPDs. In order to reduce ambiguity and to facilitate interoperable and effective tool support, this baseline specification imposes some degree of consistency on the terminology, syntax, semantics, and composition of MPDs.

1.2 Purpose

This document (and its appendices) is a normative specification for the various kinds of NIEM MPDs. The rules and guidance herein are designed to encourage and facilitate NIEM use and tools by balancing consistency, simplicity, and flexibility. Consistency and simplicity and make MPDs easy to design correctly, build rapidly, and find easily (for reuse or adaptation).

Consistency also facilitates tool support. Flexibility enables more latitude to design and tailor MPDs for complex data exchange requirements. As such, this document does not necessarily prescribe mandates or rules for all possible situations or organizational needs. So, if an organization determines it should impose additional constraints or requirements on its IEPDs beyond those specified in this document (for example, requiring some normative form of business requirements and domain model within IEPD documentation), then it is free to do so, as long as there are no conflicts with this MPD Specification or the **[NIEM-NDR]**.

This document defines terminology, identifies required and optional artifacts and metadata in MPDs, specifies normative rules, schemes, and syntax, provides non-normative guidance, and as needed, refers to other related NIEM specifications for more detail.

1.3 Scope

This specification applies to information exchange definitions and release products that employ the data component definitions and declarations in NIEM Core and Domains. In particular, this version of this document only applies to the following MPDs:

- NIEM releases (including major, minor, and micro releases).
- NIEM core updates.
- NIEM domain updates **[NIEM-DomainUpdate]** (Note that these are NOT the same as the NIEM domain schemas that are part of numbered releases).
- NIEM-conforming Information Exchange Package Documentation (IEPD) defining NIEM data exchanges.
- NIEM Enterprise Information Exchange Models (EIEM) from which one or more NIEM-conforming IEPDs can be built or based.

In the future, as required, other types of MPDs may be added to this list.

At any point in time, an incomplete MPD will be in some state of development. This specification is applicable to such developing products in that it establishes standards on the final, published, production-quality state. In turn, tool vendors can craft, adapt, and/or integrate software tools that will assist in the development of MPDs from raw parts to finished product.

NIEM is a data layer for an information architecture. Files in an MPD generally define XML Schema types and declare XML elements and attributes to use in payloads for information exchanges. While an MPD may also contain files from layers beyond the data layer, this specification is not intended to define details of other architectural layers. Such files are generally present only to provide additional context and understanding for implementing the exchange of payloads.

Authoritative sources are not required to revise MPDs that exist before this specification becomes effective. However, they are always encouraged to consider revising MPDs to meet this specification, especially when making other significant changes.

1.4 Audience

The following groups should review and comply with this specification:

- The NIEM release manager who is responsible to integrate and publish NIEM releases and core updates.
- NIEM domain stewards who will develop and publish domain updates.
- NIEM IEPD developers and implementers.
- NIEM tool developers and vendors.
- Organizations that intend to develop an EIEM.
- Individuals or groups responsible to review and approve MPDs.

2 Concepts and Terminology

The presentation of concepts and terms in this section is sequenced for understanding. Each subsection builds upon the previous ones. This section concludes with an explanation of each of the five IEM classes and a summary of their similarities and differences.

2.1 Key Words for Requirement Levels

Within normative content rules and definitions, the key words MUST, MUST NOT, SHALL, SHALL NOT, SHOULD, SHOULD NOT, MAY, RECOMMENDED, REQUIRED, and OPTIONAL in this document are to be interpreted as described in **[RFC2119-KeyWords]**.

2.2 Character Case Sensitivity

This specification imposes many constraints on the syntax for identifiers, names, labels, strings, etc. In all cases, unless otherwise explicitly noted, syntax is case sensitive. In particular, XML files in appendices that define particular artifacts, transformations, and examples are case sensitive.

Also, note that as a general principle, lower case characters are used whenever such will not conflict with the **[NIEM-NDR]**.

2.3 Artifacts

MPDs are generally composed of files and file sets grouped for a particular purpose. Each file AND each logical set of such files is called an *artifact*. In other words, we refer to a set of files (with a defined purpose) as an artifact, and we refer to each file within that set as an artifact.

> Definition: **artifact** – A single file with a defined purpose or a set of files logically grouped for a defined purpose. An MPD is a collection of related artifacts.

While the kernel of an MPD is its set of XML schema artifacts (i.e., its IEM), there are also other kinds of MPD artifacts. These may include XML, text, or graphic files meant to inform humans, or intended to help accelerate use and implementation of the MPD by software tools. **Appendix F: MPD Artifacts** contains a listing of mandatory and optional artifacts for the five types of MPDs. The various types of artifacts are described in more detail in subsequent sections.

2.4 Schema-Namespace Correspondence in NIEM

To simplify automatic schema processing and reduce the potential for confusion and error, **[NIEM-NDR]** Principle 8 states that "each NIEM-conforming namespace SHOULD be defined by exactly one reference schema." To support this principle, **[NIEM-NDR]** Rule 6-5 disallows the use of `xsd:include`, and **[NIEM-NDR]** Rule 6-35 mandates the use of the `xsd:schema targetNamespace` attribute in NIEM-conforming schemas.

The foregoing NDR rules and principle imply that each NIEM namespace is a single NIEM-conforming schema, and each NIEM-conforming schema declares a target namespace. NIEM does not permit schemas without target namespaces, unless they are from sources outside of NIEM.

2.5 Harmonization

Harmonization is a process that NIEM governance committees and domain stewards iteratively apply to NIEM content (specifically, its semantics, structure, and relationships) during the preparation of a NIEM major or minor release. The result is change and evolution of the model with the intent of removing semantic duplication and overlap while improving representational quality and usability.

> Definition: **harmonization –** Given a data model, *harmonization* is the process of reviewing its existing data definitions and declarations; reviewing how it structures and represents data; integrating new data components; and refactoring data components as necessary to remove (or reduce to the maximum extent) semantic duplication and/or semantic overlap among all data structures and definitions resulting in representational quality improvements.

2.6 Validation

This specification often refers to the process of *validation*, that is, validating XML schemas and XML document instances. Do not interpret these references to mean that validation must *always* be performed. In reality, the only time a schema or instance validates is when some process executes and performs validation. Generally, this should occur periodically during design time to ensure the conformance and quality of an information exchange definition. However, there are many cases the need to validate is a local architectural or policy decision, and some of these cases may require runtime validation.

The schemas that define a NIEM information exchange must be authoritative. Application developers may use other schemas for various purposes, but for the purposes of determining conformance, the authoritative schemas are relevant. This does not mean that XML validation

must be performed on all XML instances as they are served or consumed; only that the XML instances validate if and when XML validation is performed. The XML Schema component definitions specify XML documents and element information items, and the instances should follow the rules specified by the schemas, even when validation is not performed.

2.7 Reference Schema

A NIEM *reference schema* is a schema that is intended to be the authoritative definition of business semantics for components in its namespace. Reference schemas include the NIEM Core schema, NIEM domain schemas, and NIEM domain update schemas. The normative definition for a reference schema and applicable conformance rules are found in the **[NIEM-NDR]**. It is repeated here:

> Definition: **reference schema** – An XML Schema document that meets all of the following criteria:
>
> • It is explicitly designated as a reference schema. This may be declared by an MPD catalog or by a tool-specific mechanism outside the schema.
>
> • It provides the broadest, most fundamental definitions of components in its namespace.
>
> • It provides the authoritative definition of business semantics for components in its namespace.
>
> • It is intended to serve as the basis for components in IEPD and EIEM schemas, including subset schemas, constraint schemas, extension schemas, and exchange schemas.
>
> • It satisfies all rules specified in the **[NIEM-NDR]** for reference schemas.
>
> See also **reference schema set**.

> Definition: **reference schema set** – A set of related reference schemas, such as a NIEM release. See also **reference schema**.

The **[NIEM-NDR]** conformance rules for reference schemas are generally stricter than those for other classes of NIEM-conformant schemas. For example, they are required to employ an `xsd:annotation` with `xsd:documentation`, and `xsd:appinfo` elements that encapsulate semantic information for each XML element and attribute declaration, and type definition.

Reference schemas are very uniform in their structure. As they are the primary definitions for data components, they do not need to restrict other data definitions, and they are not allowed to use XML Schema's complex type restriction mechanisms.

2.8 Coherence of Schema Sets

A NIEM release is always a *coherent* set of schemas in which multiple versions of semantically identical types or properties do not exist; and all types and properties are uniquely defined and declared. Each numbered release has been harmonized, tested, and carefully reviewed by NIEM governance committees in order to eliminate semantic duplication.

The [NIEM-HLVA] defines a coherent schema set as one that has the following properties: (1) the schema set does not refer to a schema outside the set (i.e., the set is closed), and (2) the set does not include two different versions of the same component in an incompatible way.

> Definition: **schema coherence** – A schema set is coherent when it has the following properties: (1) the schema set does not refer to a schema outside the set (i.e., the set is closed), and (2) the set does not include two different versions of the same component in an incompatible way.

Consider the following simple example of incoherence in the figures below. Consider Figure 2-1 in which Justice domain has published a new schema (version 4.1). Note the *descendant* relationships between the old and new data components. A schema set consisting of Screening 1.1 and Justice 4.1 is incoherent because it refers to the old Justice 4.0 schema outside the set, and therefore, violates the first criterion (the set must be closed). To resolve this we could incorporate the older 4.0 version into this set. Figure 2-2 indicates that adding Justice 4.0 violates the second criterion because multiple versions of the same component will exist that are incompatible. To make a coherent schema set, either the Screening domain must be adjusted to use the new Justice 4.1 component or the schema set must be revised to use the Screening domain with Justice 4.0 and not Justice 4.1.

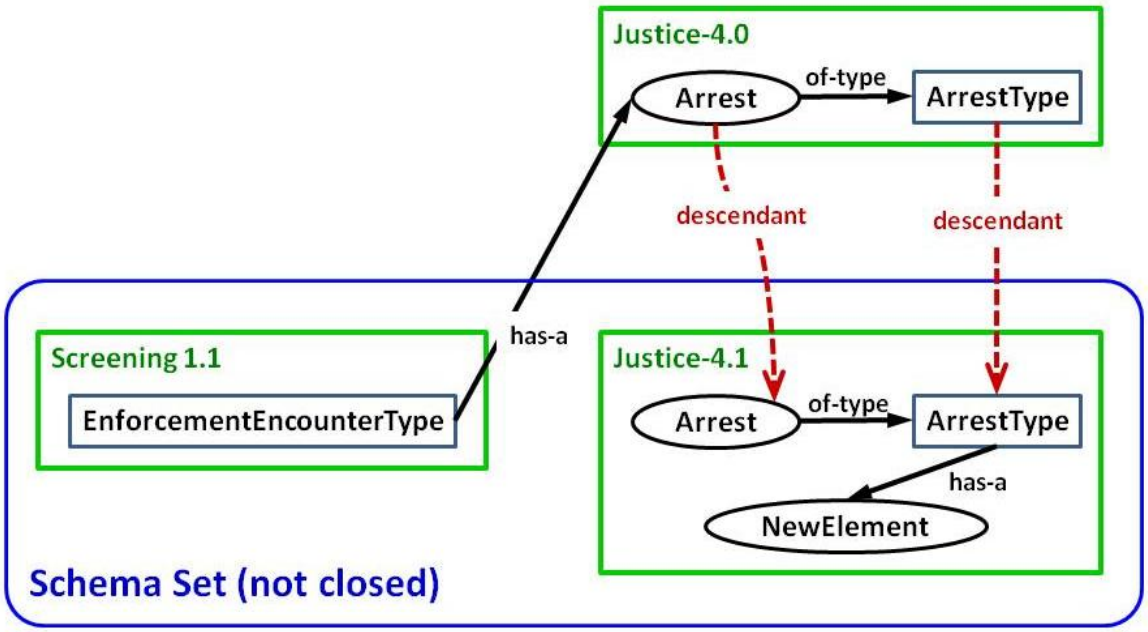


Figure 2-1. Incoherent schema set – not closed.

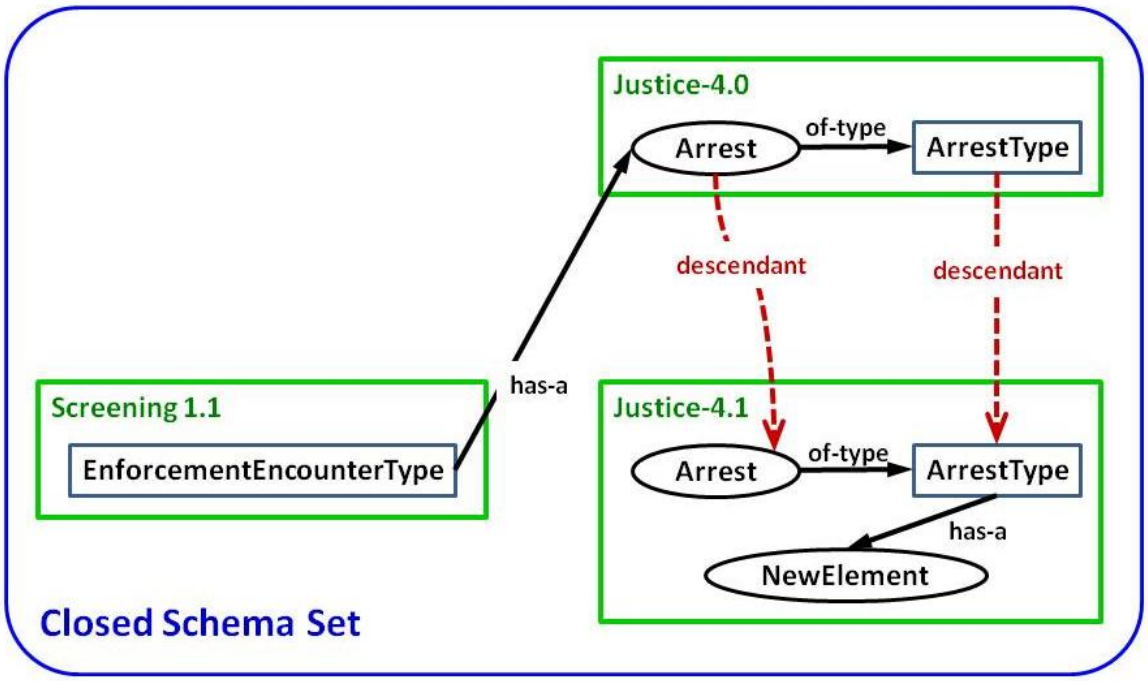


Figure 2-2. Incoherent schema set – incompatible data components.

In general, two or more versions of a data component are incompatible when a type or element in one version of a schema has been copied to or redefined/redeclared in another, and both versions must exist in the same set because of cross referencing (as in the figure above). Note that even if all data components have not changed within two versions of the same schema, a set that contains both schemas will still be incoherent because the mere duplication of a data component in a new namespace is considered redefinition (and, of course, duplication).

However, two versions of a data component can also exist in a compatible way. The compatibility of two different versions of a data component depends on the way the ancestor component was changed to obtain the descendant. In Figure 2-2, Justice 4.1 and 4.0 `Arrest` elements are incompatible because the 4.1 version of `Arrest` was simply given an additional property (`NewElement`) and is essentially a redeclaration of the 4.0 version. This results in two semantically identical elements. In fact, as already mentioned, even if the ArrestType had remained the same across both versions, the 4.1 version is considered a redefinition and duplication of the 4.0 version.

On the other hand, if the 4.1 `ArrestType` had been derived (through type derivation) from the 4.0 version, and the 4.1 `Arrest` element had been made substitutable for the 4.0 version, then these components would be compatible. The difference is that these components have a clear relationship to their ancestors that is defined through XML mechanisms, whereas the former components do not. Furthermore, the substitutability property makes these components easily usable together (i.e., compatible).

The need to be a coherent schema set is ONLY required by official NIEM releases (major, minor, and micro). A core update is not absolutely required to be coherent with the core it

applies to. However, except in rare cases, it will be crafted to be coherent. In order to provide flexibility to domains, a domain update schema set is not required to be coherent. Whether or not a domain update is coherent with a given release, is dependent upon its change log which indicates how it changes the schemas it applies to.

2.9 MPD Types

This section details the five types of MPDs currently in NIEM. As previously indicated, an IEM is a set of related XML schemas that conform to NIEM and define the structure, semantics, and relationships of XML data objects that will be transmitted in XML documents. An IEM by itself is not particularly useful or understandable. However, when properly packaged as an MPD (with documentation, metadata, examples, and other context), an IEM becomes purposeful, meaningful, and useful.

2.9.1 NIEM Release

A NIEM *release* is an MPD containing a full set of harmonized reference schemas that coherently define and declare all content within a single version of NIEM. NIEM releases include major, minor, and micro releases (as defined in the NIEM High Level Version Architecture (HLVA) **[NIEM-HLVA]**).

> Definition: **release –** A set of schemas published by the NIEM Program Management Office (PMO) at http://release.niem.gov/niem/ and assigned a unique version number. Each schema defines data components for use in NIEM information exchanges. Each release is independent of other releases, although a schema may occur in multiple releases. A release is of high quality, and has been vetted by NIEM governance bodies. A numbered release may be a major, minor, or micro release.

Current real examples of NIEM releases include NIEM major releases 1.0 and 2.0, and minor release 2.1. Each numbered release is a set of schemas that includes a NIEM Core (along with the various code list schemas that supplement Core) and NIEM domain schemas.

> Definition: **major release** – A NIEM release in which the NIEM Core schema has changed since previous releases. The first integer of the version number indicates the major release series; for example, versions 1.0, 2.0, and 3.0 are different major releases.

> Definition: **minor release** – A NIEM release in which the NIEM Core has not changed from previous releases in the series, but at least one or more domain schemas have changed. A second digit greater than zero in the version number indicates a minor release (for example, v2.1). Note also that major v2.0 and minor v2.1 are in the same series (i.e., 2) and contain the same NIEM Core schema.

> Definition: **micro release** – A NIEM release in which neither the NIEM Core nor the domain schemas have changed from the previous minor release, but one or more new schemas have been added (without impact to the domain or Core schemas). A third digit greater than zero in the version number indicates a micro release (for example, v2.1.1 – this release does not exist as of this date).

A micro release is NIEM release that strictly adds data components to Core or domain schemas without removal or modification of existing content. Figure 2-3 illustrates both real (v1.0, v2.0, and v2.1) and fictitious examples of major, minor, and micro release composition.

Note that a given NIEM schema (namespace) can exist in multiple numbered releases. For example, as illustrated in Figure 2-3, both NIEM 2.0 and 2.1 contain (reuse) the same NIEM Core 2.0 schema. Reuse of schemas among releases is carefully coordinated to ensure coherence is maintained within each release. The **[NIEM-HLVA]** defines the processes for numbering releases and identifying the schemas that compose them. Later, this specification will outline a similar version numbering scheme for MPDs and their artifacts.

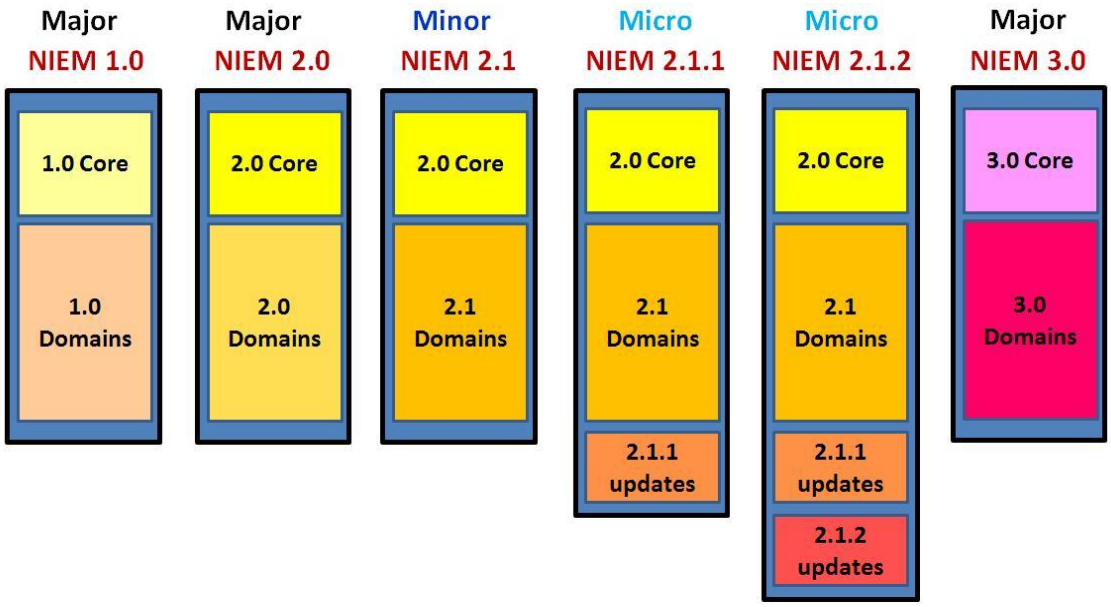


Figure 2-3. Examples of NIEM numbered releases

2.9.2 Domain Update

A *domain update* is an MPD containing reference schemas that represent changes to NIEM domains. The [NIEM-HLVA] defines a *domain update* as both a process and a NIEM product. Through use and analysis of NIEM releases and published content, domain users will identify issues and new data requirements for the domain and sometimes Core. NIEM domains use these issues as the basis for incremental improvements, extensions, and proposed changes to future NIEM releases. Both the process and product of the process are referred to as *domain update*. This MPD Specification is applicable to a *domain update* product.

> Definition: **domain update** – A MPD that contains a schema or set of schemas issued by one or more domains that constitutes new content or an update to content that was previously included in a NIEM release. A domain update may define and declare new versions of content from NIEM releases or other published content. The issuing body vets each update before publishing, but the update is not subject to review by other NIEM bodies. A domain update must be NIEM-conformant, but otherwise it has fewer constraints on quality than does a NIEM release. Domain update schemas contain proposed future changes to NIEM that have not been published in a numbered release and have not been vetted by NIEM governance bodies (except by the domain or domains involved). Domain updates are published to the NIEM Publication Area at http://publication.niem.gov/ and available for immediate use within information exchanges.

A *domain update* represents changes to domain schemas. The primary artifacts that define these changes are one or more reference schemas and a change log. A domain update may apply to one or more domain namespaces within a single NIEM major, minor, or micro release. A domain steward uses a domain update to: (1) make new or changed domain content immediately available to NIEM data exchange developers between NIEM releases, and (2) request that new or changed content be harmonized into a future NIEM release. (See the Domain Update Specification **[NIEM-DomainUpdate]** which provides normative details about domain updates and the associated processes.)

2.9.3 Core Update

When necessary, the NIEM PMO can publish a *core update*. This is essentially identical to a domain update in terms of structure and use, with two important exceptions. First, a core update records changes that apply to a particular NIEM core version or another core update. This also means it is applicable to all NIEM releases using that same core version. Second, a core update is never published to replace a NIEM core. It is intended to add new schemas, new data components, new code values, etc. to a core without waiting for the next major release. In some cases, minor modifications to existing data components are possible.

> Definition: **core update** – An MPD that it applies changes to a given NIEM core. It is never used to replace a NIEM core; instead, it is used to add new schemas, new data components, new code values, etc. to a particular NIEM core. In some cases, a core update can make minor modifications to existing core data components.

As with domain updates, all core updates are published to the NIEM Publications Area, their changes are immediately available for use in IEPDs, and they will be harmonized and integrated into the next major NIEM release.

2.9.4 Information Exchange Package Documentation (IEPD)

NIEM *Information Exchange Package Documentation* (IEPD) is an MPD that defines a recurring XML data exchange.

> Definition: **Information Exchange Package Documentation (IEPD)** – An MPD that contains NIEM-conforming schemas that define one or more recurring XML data exchanges.

A NIEM IEPD is a set of valid XML schemas that may include portions of NIEM Core schemas (and updates), portions of NIEM Domain schemas (and updates), enterprise-specific or IEPD-specific extension schemas, and at least one exchange schema that defines a *document element* (as defined in **[W3-XML-InfoSet]**). The schemas contained in an IEPD work together to define a class of XML instances that consistently encapsulate data for information exchanges. Each XML instance in this class validates against the set of XML schemas contained within the IEPD. XML schemas in a NIEM IEPD conform to the **[NIEM-NDR]** and may use or extend data component definitions drawn from NIEM.

An IEPD consists of a minimal but complete set of artifacts (XML schemas, documentation, sample XML instances, etc.) that together define and describe an implementable NIEM information exchange. A complete and normative IEPD should contain all the schema definitions and instructional material necessary to:

- Understand information exchange content, semantics, and structure.
- Create and validate information exchanges defined by the IEPD.
- Identify the lineage of the IEPD and optionally its artifacts.

An NIEM IEPD defines an *Information Exchange Package* (*IEP*), which is an XML instance document that validates to the schemas in the IEPD. An IEP is an information message payload as it will appear when transmitted electronically and serialized as XML. (**[FEA-DRM]** and **[GJXDM-IEPD]** and are the original sources of the terms *information exchange package* and *information exchange package documentation*, respectively).

> Definition: **Information Exchange Package (IEP)** – An XML document that is a correct instantiation of a NIEM IEPD, and therefore, validates with the XML schemas of that IEPD.

2.9.5 Enterprise Information Exchange Model (EIEM)

As an organization develops IEPDs, the organization may realize that many of its IEPDs have similar business content. A collection of closely related business data could be organized at an object level and defined as extension data components. In NIEM, these extension components are referred to as *Business Information Exchange Components (BIEC)*, because they are either specific to an organization's business or they represent a more general line of business that crosses organizational lines. Often they are business data components developed and used by multiple organizations within the same community of interest. So, instead of an "organization," it is more appropriate and provides better context if we use the term *information sharing enterprise*.

> Definition: **Information Sharing Enterprise** – A group of organizations with business interactions that agree to exchange information, often using multiple types of information exchanges. The member organizations have similar business definitions for objects used in an information exchange and can usually agree on their common BIEC names and definitions.

Information sharing enterprises may cross various levels of government and involve multiple business domains. They are self-defining and can be formal (with specific governance) or informal and *ad hoc*. An information sharing enterprise is the primary entity that supports the development and management of BIECs and an associated Enterprise Information Exchange Model (EIEM) (to be discussed next). Henceforth, unless otherwise stated, all references to an *enterprise* will implicitly mean *information sharing enterprise*.

A *Business Information Exchange Component* (*BIEC*) **[NIEM-BIEC]** is a NIEM-conforming content model in XML Schema for a data component that meets the specific business needs of an information sharing enterprise for exchanging data about something that is a part of one or more information exchanges. This data component is tailored and intended to be used consistently across multiple IEPDs built by an enterprise. A BIEC is a NIEM-conforming data component that is:

- Reused from a NIEM release (for example, as a subset; with possibly modified cardinality), or
- Extended per the **[NIEM-NDR]** from an existing NIEM data component, or
- Created per the **[NIEM-NDR]** as a new data component that does not duplicate existing NIEM components within a release in use.

> Definition: **Business Information Exchange Component (BIEC)** – A NIEM-conforming XML schema data component definition or declaration (for a type, element, attribute, or other XML construct) reused, subsetted, extended, and/or created from NIEM that meets a particular recurring business requirement for an information sharing enterprise.

The use of BIECs has the potential for simplifying IEPD development and increasing consistency of the business object definitions at all steps in the process, including exchange content modeling, mapping to NIEM, creating NIEM extension components, and generating XML schemas.

An *Enterprise Information Exchange Model* (*EIEM*) is an MPD that incorporates BIECs that meet enterprise business needs for exchanging data using NIEM **[NIEM-BIEC]**. An EIEM is an adaptation of NIEM schemas, tailored and constrained for and by an enterprise. An EIEM contains the following schemas that are commonly used or expected to be used by the authoring enterprise:

- One standard NIEM schema subset (or reference schema set).
- One or more NIEM extension schemas that extend existing NIEM data components or establish new NIEM-conforming data components.

- Optionally, as needed, one or more NIEM constraint schema sets (usually based on a subset).
- Optionally, as needed, one or more XML schemas for non-NIEM (non-conforming) standards with associated extension schema(s) that contains adapter types for the data components that will be used from those non-NIEM schemas (per **[NIEM-NDR]**).

> Definition: **Enterprise Information Exchange Model (EIEM)** – An MPD that contains NIEM-conforming schemas that define and declare data components to be consistently reused in the IEPDs of an enterprise. An EIEM is a collection of BIECs organized into a subset and one or more extension schemas. Constraint schemas and non-NIEM-conforming external standards schemas with type adapters are optional in an EIEM.

An information sharing enterprise that creates and maintains an EIEM authors IEPDs by reusing its EIEM content instead of (re)subsetting NIEM components and (re)creating extensions. An EIEM may also contain business rules or constraint schemas tailored to the enterprise and designed to restrict variability in use of NIEM data components. This not only saves time, but it also ensures that enterprise IEPDs reuse NIEM and associated extensions consistently. (Note that schema subsets, extension schemas, and constraint schemas will be defined and discussed in more detail later in this document). Figure 2-4 below illustrates relationships among BIECs, an EIEM, and a family of IEPDs (Constraint schemas are optional and not depicted in Figure 2-4).

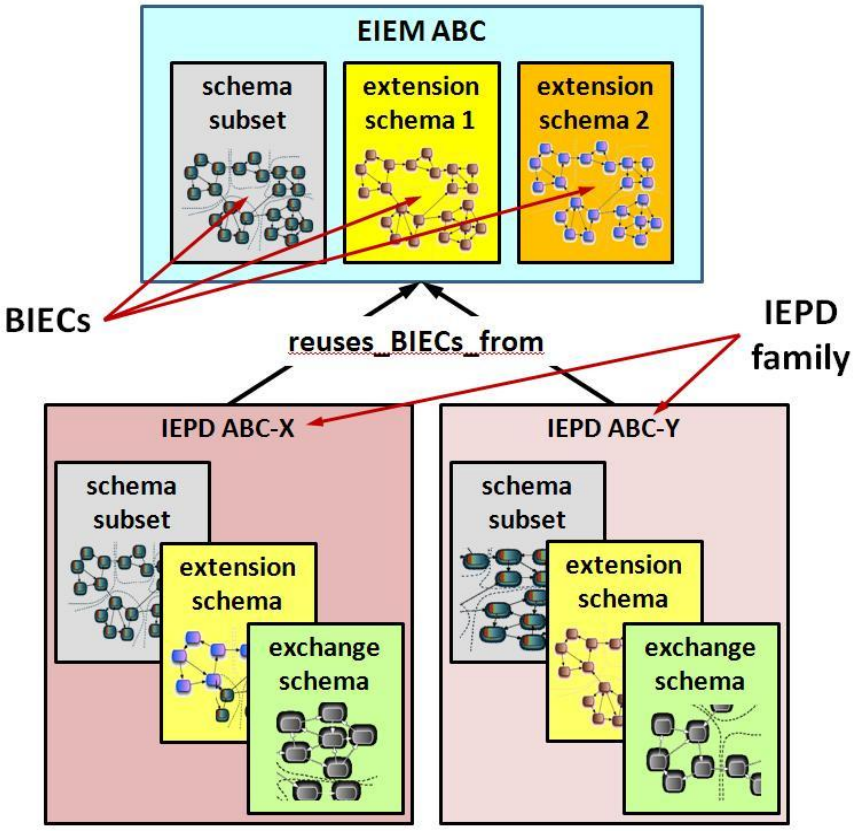


Figure 2-4. BIECs, EIEM, and a small family of IEPDs

2.10 Similarities and Differences of MPD Classes

It will be helpful to summarize the foregoing discussions by listing the primary similarities and differences among the various types of MPDs. This will help highlight the nature of this specification as a baseline and point of leverage for all five classes of IEMs and their associated MPDs: NIEM release, core update, domain update, IEPD, and EIEM. Note that these lists are not all inclusive.

MPD class similarities:

- Principal artifacts are XML schemas (each MPD contains an IEM), the purpose for which is to define and declare reusable data components for information exchanges or to define the exchanges themselves.
- Each MPD requires a self-documenting catalog artifact (containing metadata and a listing of all its artifacts), which establishes its purpose, lineage, organization, etc.
- Each MPD requires a change log.
- Each MPD requires a Uniform Resource Identifier (URI) and a version number.
- Each MPD must be packaged as a self-contained ZIP archive (in one form). Self-contained simply means that an MPD has copies of (not just URLs or references to) all the schemas needed to validate instances it defines.
- Each MPD may contain optional alternate representations for its IEM besides the required XML Schema (for example, a generic graph, a UML graph, XMI, database format, spreadsheet, etc.).

MPD class differences:

- IEPDs and EIEMs contain subset, extension, and constraint schema sets. NIEM releases, core updates, and domain updates contain reference schema sets.
- IEPDs must contain one or more exchange schemas; other IEM classes do not.
- An IEPD must contain at least one sample instance corresponding to each root element declared in its exchange schema(s). An XSLT artifact to display instances is optional.
- EIEMs and domain updates may optionally contain sample XML instances and XSLT files to display them. NIEM releases and core updates do not.
- Domain updates supersede other published schemas/namespaces and do not include other unchanged content. IEPDs, EIEMs, and NIEM releases are independently complete. Core updates can only supplement and never replace a NIEM core. However, a core update can be issued as a new complete standalone schema to be used with a NIEM core.

The following table summarizes the similarities and differences of MPD classes by indicating the characteristics for each:

Characteristics	Model Package Description (MPD) Classes				
	Release	Core Update	Domain Update	IEPD	EIEM
Requires a URI	X	X	X	X	X
Requires a version number	X	X	X	X	X
Must be packaged as a ZIP archive	X	X	X	X	X
May contain alternate model representations (in addition to XSD)	X	X	X	X	X
Requires self-documenting XML catalog (specified by an XSD)	X	X	X	X	X
Requires a formal XML change log (specified by an XSD)	X	X	X		
Requires a change log, but may be informal; any format				X	X
Requires a master document				X	X
Its XML schemas define reusable metadata components	X	X	X		X
Its XML schemas define data exchanges				X	
May contain subset, extension, constraint, and/or exchange schemas				X	
Contains subset and extension schemas; optionally constraint schemas					X
Contains reference schemas only	X	X	X		
Must contain at least one or more exchange schemas				X	
May contain sample XML instances that validate to schemas			X	X	X
Required to be independently complete standalone schema sets	X			X	X
May be independently standalone schema sets		X	X		
May supersede other published schemas/namespaces			X		

Table 2-1. Comparison of MPD classes

3 MPD XML Schema Artifacts

XML schema artifacts are the essential content of MPDs because they normatively define and declare data components. The purpose of an MPD is determined by the IEM (schemas) it contains, and each schema within that IEM may have a different purpose. The **[NIEM-NDR]** addresses the various types of schemas as conformance targets: reference, subset, extension, exchange, and constraint schemas. Each conformance target may adhere to a different (though possibly overlapping) set of conformance rules. Consult the **[NIEM-NDR]** for these rules.

The following subsections will define each type of NIEM schema and identify the types of MPDs that may or must contain them. The last subsection discusses sample XML instances (IEPs) that validate with IEPD schemas, and when such instances are mandatory.

3.1 Reference Schemas

This section generally applies to NIEM releases, core updates, and domain updates. Though not common, it is also valid to use reference schemas within IEPDs and EIEMs.

Reference schema and *reference schema set* were defined earlier in Section 2.6, but for convenience we replicate their definitions and a few of the salient points here.

Definition: **reference schema** – An XML Schema document that meets all of the following criteria:

• It is explicitly designated as a reference schema. This may be declared by an MPD catalog or by a tool-specific mechanism outside the schema.

• It provides the broadest, most fundamental definitions of components in its namespace.

• It provides the authoritative definition of business semantics for components in its namespace.

• It is intended to serve as the basis for components in IEPD and EIEM schemas, including subset schemas, constraint schemas, extension schemas, and exchange schemas.

• It satisfies all rules specified in the [NIEM-NDR] for reference schemas.

See also **reference schema set**.

Definition: **reference schema set** – A set of related reference schemas, such as a NIEM release. See also **reference schema**.

A NIEM reference schema is intended to be the authoritative definition schema for a NIEM namespace, therefore, all NIEM releases, core updates, and domain updates are composed of one or more reference schemas. As a standalone artifact, a reference schema set is always coherent and harmonized such that all types and properties are semantically unique (i.e., multiple versions of semantically identical types or properties do not exist within the set).

As authoritative definitions, NIEM reference schemas satisfy more rigorous documentation requirements. The **[NIEM-NDR]** requires that each type definition, and element and attribute declaration in a reference schema contain an `xsd:annotation` element that defines its semantic meaning. As will be explained later, *extension schemas* are also authoritative definitions, but in a local sense. They are authoritative within a given IEPD or EIEM, and therefore, must also satisfy the same rigorous documentation rules as reference schemas.

Typically reference schemas contain data components with the most relaxed cardinality (zero to unbounded repetition). However, this is not an absolute requirement. Cardinality in reference schemas may be constrained if necessary to model reality. For example, we might say that NIEM releases should restrict a `Person` object to a single occurrence of `BirthDate`. Unfortunately, also in reality, criminal persons often present multiple identities and multiple birth dates; and so the capability to represent such is an important data requirement of NIEM.

3.2 Subset Schemas

This section only applies to IEPDs and EIEMs. NIEM releases, core updates, and domain updates do not contain schema subsets (only reference schema sets).

3.2.1 Basic Subset Concepts

A NIEM *schema subset* (also referred to as simply *subset*) is a set of schemas that constitutes a reduced set of components extracted from a NIEM *reference schema set* associated with a given numbered release or domain update. Any given schema within a subset is referred to as a *subset schema* (terms reversed). The primary purpose for a subset is to tailor (usually reducing the size of) a full NIEM reference schema set for use within an IEPD or EIEM.

> Definition: **subset schema** – An XML Schema document that meets all of the following criteria:
>
> • It is explicitly designated as a subset schema. This may be declared by an MPD catalog or by a tool-specific mechanism outside the schema.
>
> • It has a target namespace previously defined by a reference schema. That is, it does not provide original definitions and declarations for schema components, but instead provides an alternate schema representation of components that are defined by a reference schema.
>
> • It does not alter the business semantics of components in its namespace. The reference schema defines these business semantics.
>
> • It is intended to express the limited vocabulary necessary for an IEPD or EIEM and to support XML Schema validation for an IEPD.
>
> • It satisfies all rules specified in the Naming and Design Rules [**NIEM-NDR**] for subset schemas.
>
> See also **schema subset**.

Subset schemas generally occur in sets that have been derived from a set of related reference schemas (such as a NIEM release).

> Definition: **schema subset** (**or subset schema set**) – A NIEM-conformant set of subset schemas, derived from a set of reference schemas, but which specifies instances that may consist of a portion of the reference schema set. See also **subset schema**.

Because NIEM adopts an optional and over-inclusive data representation strategy, most elements in a NIEM reference schema have zero to unbounded cardinality. So, elements with cardinality `minOccurs="0"` are optional and may be omitted from a subset if not needed for business reasons. It is also valid to constrain element cardinality within a subset, as long as it is not constrained such that the subset relationship is broken. For example, a reference schema element with cardinality (`minOccurs="0", maxOccurs="unbounded"`) may be constrained to `(0,1)` or `(1,1)` in a subset. However, if a reference schema element's cardinality is `(1,unbounded)`, it may not be constrained to `(0,1)` since this breaks the subset relationship. The interval `(0,1)` is not contained within, and instead, overlaps the interval `(1,unbounded)`.

The fundamental rule for schema subsets is as follows:

> **[Rule 3-1]** Any XML instance that validates against a correct NIEM schema subset will always validate against the entire NIEM reference schema set from which that subset was created.

Per previous discussion:

> **[Rule 3-2]** NIEM subsets may omit elements with zero cardinality and adjust the cardinality of elements in reference schemas from which they are derived, as long as the subset property is maintained.

In addition to adjusting cardinality and omitting data components with zero cardinality, a valid subset may also omit `xsd:annotation` elements that are mandatory in reference schemas. These elements do not impact XML validation which is vital to subsets in IEPDs and EIEMs.

> **[Rule 3-3]** NIEM subset schemas may omit all `xsd:annotation` elements that are in the NIEM reference schema from which it is derived.

Note that the process of deriving a subset from a NIEM reference schema set is optional; it is valid to reuse NIEM reference schemas as is within IEPDs or EIEMs. The primary reasons for subsetting are to reduce IEPD size and complexity and to focus constraints.

3.2.2 Subset Namespaces

A subset is essentially a reference schema set (numbered release) that has been modified per the above rules to support business requirements represented in an IEPD or EIEM. A subset derived from a reference schema set may differ from that reference set only in that its content has been reduced and/or constrained. For this reason, schemas in a subset adopt target namespaces that are identical to the corresponding schemas in the reference schema set.

> **[Rule 3-4]** Each schema in a schema subset derived from a schema reference set bears the same (target) namespace as the schema in the reference set on which it is based.

3.2.3 Omitting Schemas in Subsets

In some cases, an entire schema appearing in a reference set may be omitted from a corresponding subset. The following rule specifies when this is valid.

[Rule 3-5] A schema contained in a reference schema set may be omitted from a derived subset, if and only if ALL of the following conditions are true within that schema:

- No elements/attributes declared or types defined in the schema are required for business exchange purposes. AND
- No elements/attributes declared or types defined in the schema are required to support other elements or types within the subset for exchange purposes; in other words, no references to elements or types in the schema exist in any other schema of the subset.

When a schema has been omitted from a valid subset, there is no longer a reason to import it from anywhere within the subset. Therefore, because most XML validators expect the file identified in the `xsd:import schemaLocation` attribute to be present, imports of that schema must be removed:

[Rule 3-5.1] If a schema in a reference schema set has been omitted from a derived subset, then every `xsd:import` occurrence of that schema MUST be removed from all schemas within the subset.

3.2.4 Multiple Subsets in a Single IEPD or EIEM

Previous sections defined a single schema subset derived from a reference schema set. In general, an IEPD or EIEM contains a single cohesive schema subset (which may be a rather large set of files) based on one numbered NIEM release or domain update.

However, this specification does not restrict the number of different schema subsets that may be employed within a single IEPD or EIEM. Furthermore, it does not restrict the employment of schema subsets from different numbered releases within a single IEPD or EIEM. However, exercising this degree of flexibility makes it critically important that developers understand the potential consequences.

NIEM subsets represent a delicate compromise between flexibility and interoperability. On the one hand, a set of IEPDs based on the same subset and numbered release use identical data components, thereby enhancing interoperability. On the other hand, mixing dissimilar subsets from the same numbered release or mixing subsets derived from various numbered releases has the potential to negatively impact interoperability.

The NIEM mandate that every schema have a namespace prevents name conflicts between reference schema sets and between two subsets derived from different reference sets. In spite of namespace distinction, mixing subsets of multiple reference schema sets can still introduce multiple versions of semantically equivalent data components, a potentially ambiguous situation. Even employing multiple subsets together that have been derived from the same reference set has the potential to create a similar result.

Above all, it is the developer's responsibility to ensure that, if mixing subsets from one or more numbered releases within a single IEPD or EIEM, these artifacts are carefully coordinated and clearly documented to ensure the various versions of semantically equivalent data components

and different schemas with the same namespaces will not cause conflicts, confusion, and/or failure during validation or exchange implementation.

3.3 Extension Schemas

This section only applies to NIEM IEPDs and EIEMs. NIEM releases, core updates, and domain updates do not contain extension schemas.

The normative definition for a NIEM IEPD extension schema is in the **[NIEM-NDR]**. In general, an extension schema contains components that use or are derived from the components in reference schemas. It is intended to express the additional vocabulary required for an IEPD, above and beyond the vocabulary available from reference schemas.

A IEPD or EIEM developer who determines that NIEM does not contain all elements required for a given information exchange has two options to account for such requirement shortfalls. Using rules and techniques outlined in the **[NIEM-NDR]**:

- Extend an existing NIEM data component.
- Build a new NIEM conforming data component.

A NIEM extension schema may contain data components built from both options above. Employment of extension schemas in an IEPD is entirely optional.

Multiple extension schemas are allowed in a single IEPD. Developers will likely want to reuse many of their extensions in other IEPDs. Therefore, it is most efficient to put all extensions designed for reuse into one extension schema (or into a well-organized set of extension schemas), while keeping IEPD-specific extensions in a different schema. The reusable extension schema or schema set can then be easily redeployed in other IEPDs as needed.

Extension schemas generally contain new data component declarations that may (though not necessarily) be derived from or reference existing NIEM components. This being the case, reference schemas do not exist for new data components found within extension schemas. Therefore, extension schemas must satisfy the more rigorous documentation requirements of reference schemas in accordance with the **[NIEM-NDR]**. The definition or declaration of each new data component (type, element, or attribute) in an extension schema includes an `xsd:annotation` element that provides its semantics and NIEM-specific relationships.

3.4 Exchange Schemas

This section only applies to IEPDs. NIEM releases, core updates, domain updates, and EIEMs do not contain exchange schemas.

An IEPD defines one or more NIEM XML data exchanges, and therefore, a class of XML instance documents, each of which validates against the XML schemas in that IEPD. An XML instance document contains exactly one *document element*, which is the root element of the instance, and which cannot appear in the content of any other element within that XML instance document (by definition in **[W3-XML]** and **[W3-XML-InfoSet]**).

An IEPD exchange schema declares one or more document elements. Only one document element can be used in any given information exchange instance (IEP).

> Definition: **exchange schema** – A NIEM-conformant schema which specifies a document in a particular exchange.

However, a NIEM IEP (instance) could later become the payload of an XML envelope (such as a SOAP message). That XML envelope (itself an XML document instance) will have its own document element. In this case, the IEP no longer contains the document element for the instance. Therefore, in the context of an IEPD, it is more appropriate to refer to a document element as a *root element* of an IEP.

> Definition: **IEP root element** – The single top-level element in an IEP (instance). In the absence of any other XML wrapping of an IEP, a root element declared in an exchange schema is an IEP document element.

This results in the following rule:

> **[Rule 3-6]** An IEPD MUST contain at least one exchange schema artifact that declares at least one top-level root element for IEP instances specified by the IEPD.

Note that neither **[W3-XML]** nor this rule restricts the number of root elements that can be declared by an IEPD exchange schema. Any XML schema, including a NIEM exchange schema, may declare multiple root elements. Furthermore, a single IEPD may contain multiple exchange schemas if such are necessary to meet business requirements.

Nonetheless, regardless of the number of root elements declared by a given exchange schema, and in accordance with **[W3-XML]**, any XML instance (i.e., IEP) that validates against a NIEM IEPD must contain one and only one root element and that element must be declared within at least one exchange schema in the IEPD.

Similar to the case of multiple subsets, the flexibility provided by allowing multiple root elements and multiple exchange schemas in a single IEPD has the potential to be both powerful and problematic. Again, developers are responsible to carefully coordinate and clearly document multiple roots and/or multiple exchange schemas in a single IEPD to prevent ambiguity and misinterpretation related to validation, implementation, and use. (See **Section 5 Optional MPD Artifacts** for documentation artifacts that an IEPD may include when needed.)

The **[NIEM-NDR]** does not allow locally declared elements; all NIEM elements are declared with global scope at the top level. This means that any new element declaration in a NIEM-conforming exchange schema has the potential to be the root element in a corresponding IEP. Therefore, if an IEPD author does not intend for a new element to be an IEP root element, then *do not* declare it in a NIEM exchange schema. Declare it in an extension schema instead.

> **[Rule 3-7]** An IEPD exchange schema MUST NOT declare any XML element that is not intended for use as an IEP root element.

Note that this rule does not preclude the use (through "`ref=`") of other elements within the exchange schema that are declared globally elsewhere within an IEPD. In general, elements that

must be used within an exchange schema, but are not intended to be IEP root elements should be declared in extension schemas.

Now that both extension and exchange schemas have been described, it is useful to mention the following special case. Although generally rare, it is possible to develop an IEPD without an extension schema. If an author creates an IEPD based entirely on existing NIEM elements, then no extension schema is necessary (and this is the reason it is deemed optional). In this case, the exchange schema defines a root element type to contain only existing NIEM elements (i.e., drawn from a subset or reference schemas) and declares the root element of that type. The rule above merely ensures that the exchange schema declares root element(s) that are intended and will be used as the roots of XML instances, and no others. However, because all NIEM elements must be declared with global scope **[NIEM-NDR]**, in order to declare non-root elements in an IEPD, they will have to be declared in an extension schema.

It is usually good practice to maintain namespace cohesion. If root elements declared in an IEPD tend to change together, then group them together in the same namespace (i.e., single exchange schema). If elements tend to change independently, then group them in separate namespaces (i.e., multiple exchange schemas). Furthermore, while there are no restrictions on the number of root elements and exchange schemas, it may be best to first consider declaring one root per exchange schema if this arrangement can support the IEPD business requirements. If not, only then consider scaling upward. In all cases, clearly document the intent and rationale for how an IEPD is organized and to be implemented.

3.5 Constraint Schemas

This section only applies to NIEM IEPDs and EIEMs which may use constraint schemas. NIEM releases, core updates, and domain updates do not contain constraint schemas.

A constraint schema is an optional IEPD or EIEM artifact that is used to express business rules for a class of XML documents, and is not assumed to be a definition for the semantics of the components it contains and describes. Instead, a constraint schema uses the XML Schema definition language to add constraints and restrictions to components defined or declared by other schemas, usually subset schemas.

> Definition: **constraint schema** – A schema which adds additional constraints to NIEM-conformant instances. A constraint schema or a constraint schema set validates additional constraints imposed on an XML instance only after it is known to be NIEM-conforming (i.e., has been validated by reference schemas, subset schemas, extension schemas, and/or exchange schemas). Constraint schema validation is a second-pass validation that occurs independently of and after conformance validation. A constraint schema is not required to be NIEM-conforming. A constraint schema need not validate constraints that are applied by other schemas. See also **constraint schema set**.

> Definition: **constraint schema set** – A set of related constraint schemas that work together, such as a constraint schema set built by adding constraints to a schema subset. See also **constraint schema**.

Note that constraint schemas are generally useful when it is necessary to impose restrictions that are more complex than cardinality. If only cardinality restrictions are needed, then it is easier and more efficient to set these directly in the schema subset and avoid the use of constraint schemas. Otherwise, a constraint schema may be necessary. Note however, that any cardinality restrictions placed on NIEM release components within subsets must not violate the rules established in **3.2.1 Basic Subset Concepts** which define the relationship of a subset schema to the reference schema on which it is based.

The **[NIEM-NDR]** provides a normative definition and description of constraint schemas. However, a few points are worth mentioning here.

Use of constraint schemas is one option for applying additional business rules to or tightening constraints on NIEM IEPs beyond what NIEM itself provides. This particular technique uses XML Schema. NIEM also allows other methods that do not use XML Schema, such as **[ISO-Schematron]** or other language methods. However, at this time there are no normative rules for how these techniques should be employed in NIEM IEPDs or EIEMs. Therefore, if other techniques are used, it is a developer responsibility to incorporate appropriate artifacts and clear documentation.

Constraint schemas are generally designed and employed in sets, similar to schema subsets and reference sets. A common practice for creating an IEPD or EIEM constraint schema set is to start with a valid NIEM subset and modify this set to further restrict the class of XML documents (IEPs) that will validate with this constraint set. However, an extension or exchange schema can also be used to derive a constraint schema. The namespace of a constraint schema is established the same way the namespace of a subset schema is established – it reuses the target namespace of the schema from which it is derived.

> **[Rule 3-8]** A constraint schema bears a target namespace that has been previously assigned by a reference schema, extension schema, or exchange schema, or is a schema that is intended to support a constraint schema that has such a target namespace.

To use a constraint schema to tighten constraints on IEPs, a two-pass validation technique is employed. In the first pass, an IEP is validated against the schema subset, extensions, and one exchange schema. This pass ensures that IEP semantics and structure conform to the NIEM model and NDR. In the second pass, an IEP is checked against the constraint schema set, extensions, and one exchange schema. This pass ensures that the IEP also satisfies the additional constraints (i.e., business rules that the first pass was unable to validate).

In general, constraint schemas have far fewer requirements than other classes of NIEM schemas. Since they work in tandem with NIEM normative schemas, constraint schemas are allowed to use the XML Schema language in any way necessary to express business rules. This means that to constrain instances, constraint schemas can employ XML Schema constructs that are not allowed in other classes of NIEM schemas.

BIECs in particular may have additional business rules in constraint schemas. A normative NIEM BIEC Specification (in progress at the time of the publication of this MPD Specification), will supplement or obviate constraint schemas with consistent and formal techniques for representing business rules within NIEM components. However, as already mentioned, the MPD Specification does not prohibit or restrict the application of formal business rule techniques to MPDs now.

3.6 Classes of IEMs vs. Classes of Schemas

The chart in Table 3-1 summarizes which types of schemas are contained in which classes of IEMs and where they are not applicable (NA = Not Applicable; U = unbounded).

Notice that only NIEM releases, core updates, and domain updates contain reference schema sets, while only IEPDs and EIEMs contain the user developed schema sets. Since an IEPD defines at least one data exchange, it must contain at least one exchange schema. Furthermore, the diamonds (♦) indicate that a NIEM-conforming IEPD or EIEM must have at least one schema that is either a NIEM reference schema or a subset derived from a NIEM reference schema (See **[Rule 3-9]** below this table).

Schema Classes	IEM Classes				
	Release	Core Update	Domain Update	IEPD	EIEM
Reference	1,U	1,U	1,U	0♦,U	0♦,U
Subset	NA	NA	NA	0♦,U	0♦,U
Constraint	NA	NA	NA	0,U	0,U
Extension	NA	NA	NA	0,U	0,U
Exchange	NA	NA	NA	1,U	NA

Table 3-1. IEM classes vs. schema classes

> **[Rule 3-9]** A NIEM-conforming IEPD or EIEM MUST contain at least one schema that is either a NIEM reference schema or a subset derived from a NIEM reference schema.

3.7 Sample XML Instances

XML schemas define XML data exchange instances. It is certainly possible to construct an example XML instance for an entire NIEM release, but such an instance is of questionable value. Rarely, if ever, will an entire NIEM release be used to define a data exchange. However, sample XML instances are very valuable artifacts to include with IEPDs. As samples of the actual exchange data, instances can help an IEPD implementer to understand the original intent of the IEPD developer. They can be used by an implementer as data points for validation with the IEPD schemas. And finally, the inclusion of valid, meaningful sample XML instances is an indication of IEPD quality. For these reasons, IEPDs require valid sample XML instances.

> **[Rule 3-10]** A NIEM IEPD MUST contain at least one valid sample XML instance (i.e., IEP) artifact for each exchange schema element that can be the root of a corresponding IEP.

The intent of this rule is not to provide a test for all permutations of XML instances that might be possible given the schema definitions and declarations. As the value propositions explain, its purpose is to ensure IEPD developers have tested their own designs, and to provide implementers with examples they can use for additional understanding and guidance. IEPD developers should strive to include sample XML instances that (1) capture real world business cases of data exchanges, and (2) exercise as many data component definitions and declarations in the schemas as possible. While both of these goals may not be achievable in a single sample XML instance, developers have the option to include multiple sample XML instances; however, only one per intended root element is mandatory.

Note that each sample XML instance illustrates one view of the data based on a chosen set of conditions that apply to an IEP. Other views based on different conditions likely exist. A developer must review the business rules and other documentation in an IEPD to ensure understanding of all possible conditions. Do not rely exclusively on sample XML instances, since they are not required to account for all IEP permutations.

This specification encourages but does not mandate the inclusion of sample XML instances for EIEMs and domain updates. Again, such instances may be valuable to user understanding of the intent and usage of data components (especially, those that are new, extended, or changed).

4 MPD Documentation Artifacts

A variety of documentation files may be incorporated into a NIEM MPD. However, in addition to XML schemas, there are only two mandatory documentation artifacts required by every MPD: the *catalog* and the *change log*. An MPD catalog contains identifiers, key metadata, information, and relationships about the MPD. The change log provides a history of modifications.

A *master document* is also mandatory for IEPDs and EIEMs. These MPD classes are built by different developers, and may be registered into a repository for reuse by many other users and developers; therefore, a minimal form of documentation is absolutely necessary. An IEPD or EIEM master document is the primary source and starting point for human readable documentation (similar to a `"readme"` file), and should reference (and describe) any other separate documentation artifacts. This requirement ensures that baseline documentation is consistently rooted in a clearly visible artifact within each IEPD and EIEM.

The following subsections will address these artifacts as well as the concepts, metadata, and content each supports.

4.1 Catalog

Every NIEM MPD describes itself through a mandatory catalog artifact. A catalog is a multi-purpose XML file containing metadata that describe an MPD's

- Unique identification
- Basic descriptive characteristics
- Directory structure and artifacts
- Lineage and relationships to other MPDs

This metadata is designed to be the minimal required that will facilitate human understanding, tool support, and machine processing. The MPD uses and functions that the catalog is designed to support include (but are not limited to):

- Automatic identification and processing of artifacts
- Browsing and understanding of MPD artifacts and their content
- Conformance and instance validation (as needed)
- Automatic registration in a registry/repository
- Search, discovery, retrieval of MPDs (for example, through metadata values)
- Reuse of MPDs and their artifacts
- Reuse of Business Information Exchange Components (BIEC) and associated EIEMs
- Tracing and analysis of MPD pedigree

> **[Rule 4-1]** An MPD MUST contain an XML catalog artifact that validates with the NIEM MPD catalog schema (XSD) and that resides in the root directory of the MPD and bears the file name "`catalog.xml`".

The catalog identifies every artifact, its relative path name, file type, and purpose. This enables a machine to find every artifact regardless of its location within an MPD, and know exactly what it is used for, and therefore, how to process it. **Appendix A: MPD Catalog Schema** defines the structure and semantics of a NIEM `catalog.xml` file.

4.2 Metadata Concepts

The MPD catalog contains both required and optional metadata for the MPD and its artifacts. The following subsections specify the syntax, formats, and semantics for that metadata.

4.2.1 Version Numbering Scheme

Many published MPDs will be periodically revised and updated; therefore, versioning is required to clearly indicate that changes have occurred. A version number is actually part of the unique identification for an MPD (to be discussed in a subsequent section). For this reason:

> **[Rule 4-2]** Every MPD MUST be assigned a version number.

In order to maintain some consistency while allowing reasonable flexibility to authors, this specification establishes a simple version numbering scheme that is consistent with most common practices. This is the same version numbering scheme that is used for all NIEM releases.

[Rule 4-3] All NIEM version numbers adhere to the regular expression:

```
version ::= digit+ ('.' digit+)* (status digit+)?
```

Where:

```
digit ::= [0-9]
```

```
status ::= 'alpha' | 'beta' | 'rc' | 'rev'
```

`'alpha'` indicates early development

`'beta'` indicates late development; but changing or incomplete

`'rc'` indicates release candidate; complete but not approved as operational

`'rev'` indicates very minor revision that does not impact schema validation

(The regular expression notation used above is from XML 1.0 (Fifth Edition): http://www.w3.org/TR/2008/REC-xml-20081126/#sec-notation)

The regular expression in **[Rule 4-3]** allows the following example version numbers:

- 1
- 1.2
- 1.3.1.0
- 1.2alpha13
- 199.88.15rev6

There are two implications in **[Rule 4-3]**. The first is that in some cases this version scheme implies and confirms a chronology of releases. For example, a given product labeled version 2.3 must have been released before the same product labeled 2.3.1. Therefore, version 2.3.1 is more current than version 2.3.

However, this is a multi-series version scheme, and chronological relationships exist only within a given series. So, for example, nothing can be said about a chronological relationship between versions 2.2.4 and 2.3. This is because version 2.2.4 is in a different series (i.e., 2.2) and could actually have been released after 2.3. Figure 4-1 illustrates a system of versions that uses the numbering scheme of **[Rule 4-3]**.

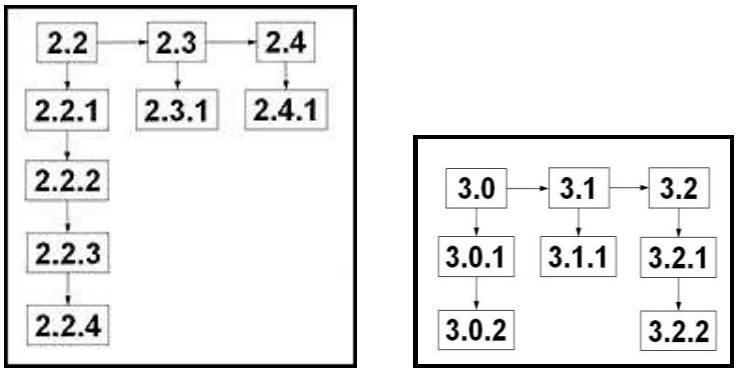


Figure 4-1. Example versioning system.

Figure 4-1 illustrates eight different version series. Within this illustration these are the only sequences that have chronological relationships that can be identified through version numbers.

- Series 2: 2.2, 2.3, 2.4
- Series 3: 3.0, 3.1, 3.2
- Series 2.2: 2.2(.0), 2.2.1, 2.2.2, 2.2.3, 2.2.4
- Series 2.3: 2.3(.0), 2.3.1
- Series 2.4: 2.4(.0), 2.4.1
- Series 3.0: 3.0(.0), 3.0.1, 3.0.2
- Series 3.1: 3.1(.0), 3.1.1
- Series 3.2: 3.2(.0), 3.2.1, 3.2.2

The second implication of **[Rule 4-3]** is that pre-releases are easily identified by the strings "`alpha`", "`beta`", and "`rc`". These strings are simple visible indicators of the status or stage of development of an MPD.

This specification places no further restrictions or meaning (implied or otherwise) on a version or release number. Authors have the option to use numbers between dots to indicate degree of compatibility or other relationships between versions as needed. For example, for a given MPD, the author may declare that if an instance validates to version 4.2.3, then it will also validate to version 4.2. Such a claim is acceptable. However, this specification does not imply any such relationships. Any meaning assigned to version sequence by an authoritative source should be unambiguously documented within the MPD.

> **[Rule 4-3.1]** A higher MPD version number within a version series does NOT imply compatibility between versions. Compatibility between or among MPD versions MUST be explicitly stated in documentation.

Note that an author who updates an existing MPD to a new version for no other reason than to conform to this specification may choose the version number based on its previous version number or not, as long as it follows the version number syntax.

4.2.2 URI Scheme for MPDs

To facilitate MPD sharing and reuse the assignment of a URI (Uniform Resource Identifier) to each MPD is essential.

> **[Rule 4-4]** Every MPD MUST be assigned a valid `http` URI.

This specification follows **[RFC3986-URI]**, which defines the syntax and format for a URI. However, this specification also restricts an MPD URI to a URL and does not allow a URN (Uniform Resource Name) to be assigned to an MPD.

Here is a typical example of an `http` URI:

```
http://www.abc.org/niem-iepd/order/2.1.2rev3/
```

Note that **[Rule 4-4]** explicitly states that a URI assigned to an MPD must be valid. This means that the person or organization assigning the URI either is the registrant of the domain name, or has authority from the registrant to assign this URL as an MPD URI. In the example above,

`www.abc.org` is the domain name (between the second and third `"/"`). There is no requirement for an URL assigned to an MPD to resolve to any particular Internet resource or to resolve at all. However, it is always good practice for such a URL to resolve to the resource it represents, the directory it resides in, or to documentation for that resource. See http://www.w3.org/Provider/Style/URI.html

The MPD version number is essential to its unique identification. Incorporation of the version number within the MPD URI provides a simple visual (as well as machine readable) means of identifying one of the most fundamental relationships between MPDs, i.e., that one is a different version of another.

> **[Rule 4-5]** The URI for an MPD MUST end in its version number.

Another advantage to this technique is that different versions of an MPD will generally group together in a standard sorted ordering. (Of course, this assumes that a related family of MPDs follows the same URI scheme.)

4.2.3 URI Scheme for MPD Artifacts

Given the URI for an MPD, a URI exists for each artifact in that MPD. Again, this specification follows **[RFC3986-URI]** and employs a fragment identifier to designate an artifact URI.

The `mpd-catalog.xsd` schema in **Appendix A: MPD Catalog Schema** declares an `id` attribute of type `xsd:ID` that is required for use in `FileType` and `FileSetType`; optional in FolderType. Within `Catalog.xml` an MPD author must assign a locally unique `id` value each artifact and artifact set of the MPD. A globally unique URI for an artifact is the concatenation of the MPD URI with `"#"` followed by the `id` value of the artifact or artifact set.

Since every MPD must have a URI, and an MPD catalog must list all artifacts contained in the MPD, and each artifact must be assigned a locally unique `id` value, then each MPD artifact has a globally unique URI that can be referenced from other external resources as needed. The following rules concerning an artifact `id` apply:

> **[Rule 4-6]** Each file artifact in an MPD MUST have a corresponding File element in the catalog for that MPD.

> **[Rule 4-7]** Each file set artifact in an MPD MUST have a corresponding FileSet element in the catalog for that MPD. This FileSet element must identify each file artifact that is a member of that file set artifact.

These rules and the catalog schema specified in **Appendix A: MPD Catalog Schema** require that each individual file artifact and each set of file artifacts grouped for a particular purpose must be identified in the catalog. This is to facilitate automatic processing of the MPD by software tools. Note that file subdirectories (or folders) are independent of file set grouping. Each file artifact can be identified in one and only one subdirectory (or folder) by its relative path name and file name. However, each file artifact can be a member of multiple file set

artifacts. Therefore, while it is possible to associate a file set artifact with a single subdirectory within an MPD, it is not required.

Also note that the `Folder` element in **Appendix A: MPD Catalog Schema** is available to represent subdirectories in MPDs. This allows an XSLT to generate a catalog index that can display subdirectories as needed. However, operating system subdirectories are not authoritative for file artifact and file set organization. For this reason the `Folder` element contains an optional `id` attribute, a required `relativePathName` attribute, but no URI attributes that could enable file grouping.

> **[Rule 4-8]** Each artifact identified in the catalog MUST be assigned an id in the format of an NCName (Non-Colonized Name) as defined by [W3-XML-Namespaces]. This is required for both File and FileSet artifacts.

By the rules of **[W3-XML]**, the value of each `id` attribute (which is of type `xsd:ID`) must be locally unique within the MPD `catalog.xml` file. However, globally unique URIs are required to identify and reference artifacts between MPDs. To facilitate references from one MPD catalog to another, the following rule applies:

> **[Rule 4-9]** A URI reference to an individual MPD artifact from another resource is the concatenation of
>
> • The URI of the MPD that contains the artifact.
>
> • The crosshatch or pound character ("#").
>
> • A fragment identifier that is the locally unique `id` of the artifact within the catalog of the MPD itself.

Example of an artifact URI:

```
http://www.abc.org/niem-iepd/order/2.1.2rev3/#a552
```

To illustrate a typical scenario for using this URI, a developer can build an IEPD that contains a schema artifact to which (within the catalog) he assigns:

```
id="x25"
```
(a locally unique `id` within the IEPD catalog)

If this schema artifact is an exact duplicate of the `a552` artifact from the published MPD whose URI is

```
http://www.abc.org/niem-iepd/order/2.1.2rev3/
```

then the developer can optionally assign the following attribute to this artifact's catalog entry:

```
externalURI=
"http://www.abc.org/niem-iepd/order/2.1.2rev3/#a552"
```

Additional `externalURI` attributes can be assigned to this artifact if the author knows of other reuses of this same artifact in other MPDs.

4.2.3.1 Compact URIs (CURIEs)

In order, to simplify creation and human review of an MPD catalog that will likely contain many long URI strings, this specification allows the optional use of a *safe CURIE* **[W3-CURIE-Syntax]**. A CURIE is a compact URI that employs a prefix for a URI similar to the way namespace prefixes are used in XML Schema. A safe CURIE encapsulates a CURIE within square brackets ("`[`" and "`]`") to ensure machine readability where it might otherwise be impossible to disambiguate between a CURIE and a URI. Within an MPD catalog.xml artifact, a CURIE may optionally be used anywhere a URI is required.

Following the lead of XML Schema, to employ a CURIE, ensure that a unique prefix has been declared in the `Catalog` element (the root element of the catalog artifact) using `xmlns`. This prefix can now be used to reduce long URI strings that contain the leading string represented by the prefix. A simple example follows.

Given the following sample catalog requirement for an XML attribute that contains a long URI:

```
ca:externalURI="http://www.organizational-iepds.any-company-
                inc.com/niem-iepd/codes/1.1.5rev6/#code-list33"
```

Within `Catalog` element (among other namespaces) use `xmlns` to declare prefix `ab` (for example):

```
<ca:Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ca="http://reference.niem.gov/niem/resource/mpd/
              catalog/1.0/"
    xmlns:ab="http://www.organizational-iepds.any-company-
              inc.com/niem-iepd/codes/1.1.5rev6/" >
```

With this declaration, and using a safe CURIE, the previous `ca:externalURI` attribute becomes:

```
ca:externalURI="[ab:#code-list33]"
```

Note that when using this technique it is very easy to inadvertently insert errors into URI references by dropping or adding characters. When applying safe CURIEs, be careful to ensure that string breaks in URIs are correct (at all locations!) so that the original URI results when the prefix is replaced to recompose the URI.

4.2.3.2 MPD Artifact URIs Are Not NIEM Namespaces

URI does not have the same meaning as *namespace*. Do not rely on namespaces for artifact URIs. Recall that the namespaces used in a schema subset derived from a NIEM release are identical to the namespaces of the release itself. Furthermore, an IEPD or an EIEM may contain multiple subsets. The non-uniqueness of NIEM namespaces implies that they cannot be used as URIs for MPD artifacts.

> **[Rule 4-10]** NIEM namespaces MUST NOT be used as URIs for MPD artifacts.

4.2.4 Artifact Vocabularies: Nature and Purpose

In order for software tools to unambiguously understand and process MPD artifacts, NIEM specifies a formal vocabulary. Each term in the vocabulary is identified by URI. The current vocabularies declare terms for both the *nature* and *purpose* of an artifact. The nature of an artifact identifies its file type; the purpose of an artifact identifies what it is used for. These vocabularies will likely expand when new NIEM requirements and specifications demand additional terms. Furthermore, other vocabularies beside nature and purpose may become necessary in the future.

> Definition: **nature** – An indication of the type of an MPD artifact. This further indicates how it should be processed by software tools.

A *nature* **[W3-AssocResourcesNS]** of an artifact specifies fundamental classification of that resource. Often the nature of a document is expressed in an informal manner. For example, "that's an XML Schema", or "that's an HTML document", or "that's an XML DTD". These phrases are identifying the *nature* of those documents. The *nature* of an artifact is an indication of its file type. However, it is important to note that the nature of an artifact is usually but not required to be consistent with its file extension.

> Definition: **purpose** – A property of an MPD artifact that indicates its usage or function. This determines what to do with this artifact and/or how it should be processed by software tools.

A *purpose* **[W3-AssocResourcesNS]** is a property of an artifact that conveys its intended usage. *Purpose* is used in conjunction with *nature* to ensure that a processor or tool will know exactly what to do with an MPD artifact. For example, two related MPD artifacts might have natures indicating they are both schemas, but the *purpose* property of each can indicate that one is part of a schema subset, while the other is an extension schema.

Artifact natures and purposes may have many relationships. For example, the Nature class diagram at the end of **Appendix G: MPD Lexicon (Nature and Purpose)** indicates that natures are organized into a hierarchical class diagram of *is-a* (sub-class) relationships; similarly, purpose properties can be sub-properties of one another. W3C Web Ontology Language (OWL) **[W3-OWL]** and the language from which it is derived, the W3C Resource Description Framework (RDF) **[W3-RDF]**, are well-suited for precisely defining vocabularies and their associated relationships. For example, the Dublin Core Metadata Initiative uses RDF to specify its library science metadata vocabularies (See http://dublincore.org/schemas/rdfs/).

Note that the use of OWL to specify the semantics of these vocabularies does not require that NIEM users have in depth knowledge of OWL. It is only important to understand the URI strings (i.e., values in `rdf:about` attributes) for the terms and their corresponding definitions in the OWL schemas that define the vocabularies. These URI strings are simply used to identify the nature and purpose values for artifacts listed in an MPD catalog. However, maintaining the vocabulary in OWL ensures the vocabulary can be easily extended with minimal impact to previous versions. It can also be published in a standard form that can be consistently interpreted by tools that must process MPDs and their associated artifacts.

Appendix G: MPD Lexicon (Nature and Purpose) contains the normative specification for both vocabularies. This appendix also illustrates nature and purpose in hierarchical formats.

Through the use of URIs, URI fragments, nature (classes), and purpose (relationship properties), the structure and content of a complete MPD can be represented with all its artifacts in a form suitable for software tools. This representation accounts for (with `relativePathName`), but is independent of, the directory organization of the MPD. Figure 4-2 below, depicts the respresentation of a few artifacts in an example IEPD. The complete IEPD is represented by the green ball in the upper left corner of the figure. Its artifacts are the remaining green balls.

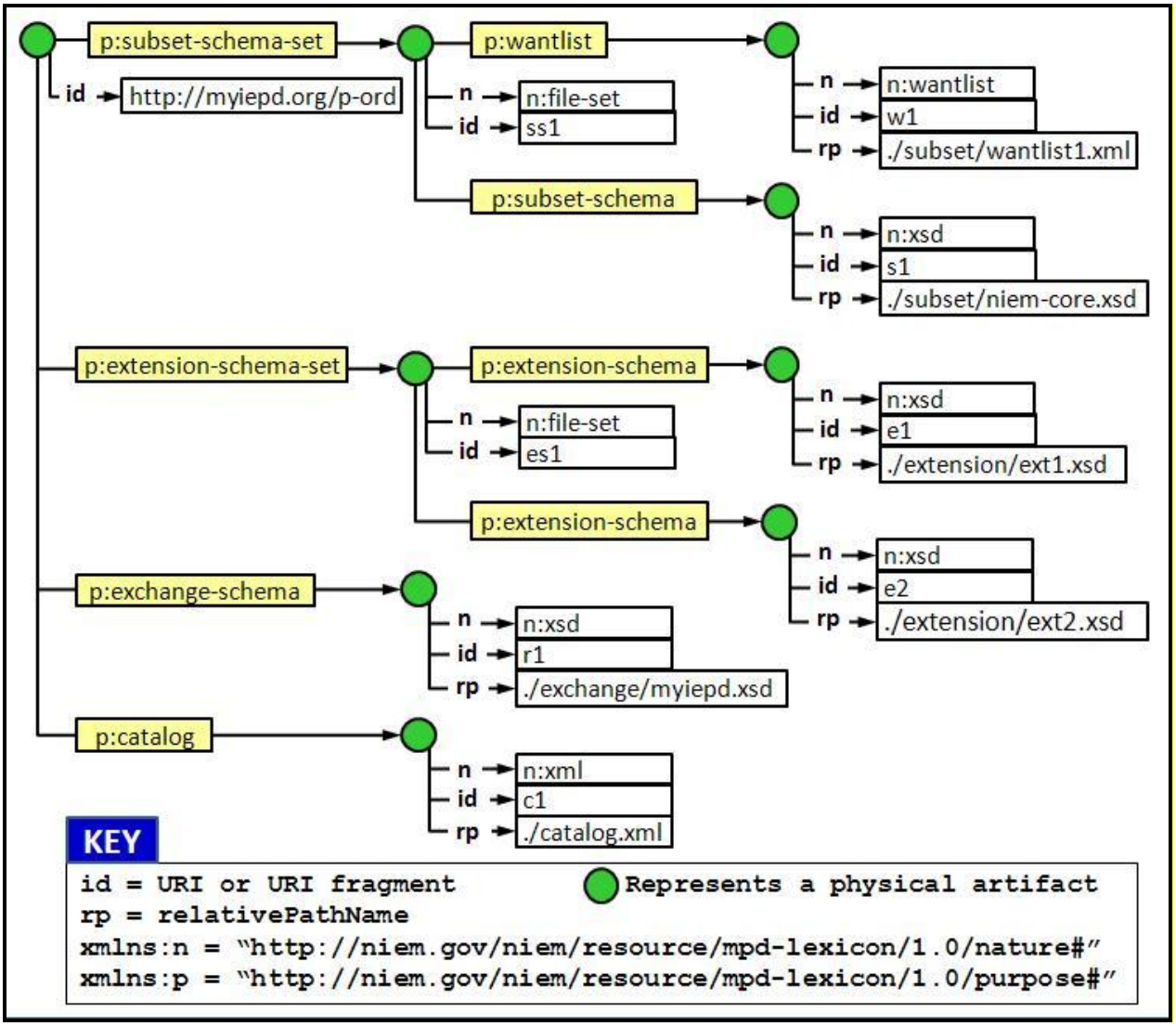


Figure 4-2. An example of IEPD representation.

Note that this figure represents an abbreviated version (only a few of the artifacts are included) of the sample IEPD catalog illustrated in **Appendix C: Sample MPD Catalog Instance**.

4.2.5 MPD Artifact Lineage

An important MPD business requirement is transparency of lineage. It must be possible to easily identify the relationships that may exist among families, versions, adaptations, specializations, generalizations, etc. of MPDs. The established URI scheme for MPDs and MPD artifacts as well as the catalog help make this possible.

The catalog provides a `Relationship` element with three attributes (`resourceURI`, `relationshipCode`, and `descriptionText`) to identify the pedigree of an MPD. There are many ways that one MPD may relate to another. This makes it extremely difficult to specify a fixed set of values that could objectively define an exact relationship between a pair of MPDs. Therefore, the optional `descriptionText` attribute is provided to further explain the nature of any of the eight `relationshipCode` values available (`version_of`, `specializes`, `generalizes`, `deprecates`, `supersedes`, `adapts`, `conforms_to`, `updates`). In some cases, the value of `relationshipCode` may be generic enough to require a more detailed explanation in `descriptionText` (for example, if the value is "`adapts`").

The catalog also enables an MPD author to record a fine-grained pedigree between MPDs when reusing artifacts from other MPDs. By default each artifact identified in a catalog has a globally unique URI (using a fragment reference) that can refer to it. An MPD author signifies reuse of a given artifact by entering the URI for that artifact in the optional `externalURI` attribute within the `File` element. Use of the `externalURI` for a given artifact does not preclude the mandatory requirement to assign a locally unique id to that artifact (per **[Rule 4-7]** and **[Rule 4-8]**).

Note that some MPDs are designed for more extensive reuse than others. For example, many IEPDs are expected to reuse an EIEM. In such cases, the catalogs for these IEPDs and the corresponding EIEM may overlap in or duplicate a large number of metadata and references. This is expected. The catalog contains many references to and semantics for artifacts and MPDs. Correct and consistent use of these references and semantics will create networks of related MPDs so that tools can automatically locate, parse, and process MPDs and their corresponding artifacts as needed and when available in shared repositories.

4.3 Change Log

4.3.1 NIEM Releases, Core Updates, and Domain Updates

Although the version identifier is useful for a fast and visual indication of the state of an MPD relative to others, it only provides a fairly generalized indication of the volume, complexity, and impact of changes that have been applied since a previous version. Of course, a change log is required to ensure a more specific accounting of changes from one version to another.

Once published, NIEM releases always exist. This ensures that IEPDs and EIEMs built from them will always be stable, and may be updated to a new NIEM release only when convenient or absolutely necessary to take advantage of new or modified data components. Though not recommended, the NIEM program does not prohibit a developer from building an IEPD based on a NIEM release that is older than the most current version. There may be potential disadvantages related to interoperability levels achievable with others developing to the latest

release. Nonetheless, an older version might meet the business needs of a particular organization quite well.

In spite of this built-in stability, the NIEM architecture is designed to evolve as requirements change. New versions of reference schema sets such as NIEM releases, core updates, and domain updates can have significant impacts on future IEPDs and EIEMs. Developers must understand in detail how changes will affect their IEPD and EIEM products and the tools used to build them. To work effectively, tools for domain content development, impact analysis, migration between releases, etc. must be able to digest formal change logs. A formal change log is also essential to efficiently process and integrate new and changed content into NIEM for new releases, and to simultaneously maintain multiple versions of NIEM for users. All of the foregoing reasons dictate that NIEM require a normative change log for reference schema sets.

[Rule 4-11] Every MPD that is a reference schema set (i.e., NIEM releases, core updates, and domain updates) MUST contain an XML change log artifact that:

- Validates with the NIEM change log schemas (`mpd-changelog.xsd` and `niem-model.xsd`).
  Note: These are the base filenames; the actual filenames also contain a version number. For example: `mpd-changelog-1.0.xsd` is the current version.
- Records changes to previous reference schemas that this MPD represents.
- Bears the file name "`changelog.xml`".
- Resides in the root directory of the MPD.

The current version of `mpd-changelog.xsd` can be found at:

http://reference.niem.gov/niem/resource/mpd/changelog/

The current version of `niem-model.xsd` which describes the NIEM conceptual model can be found at:

http://reference.niem.gov/niem/resource/model/

Since the schemas are the authority for a release or update and because almost all tool support depends on the schemas, the change log is only designed to audit transactional change to the schemas in the reference set. There is no provision for logging changes to support documentation or other non-schema artifacts. Non-schema changes are handled non-normatively in the form of release notes.

4.3.2 IEPDs and EIEMs

IEPD and EIEM change log requirements are less strict and are not required to conform to the naming and schema specifications in **[Rule 4-11]**. However, a change log is still required.

[Rule 4-12] Every MPD that is an IEPD or EIEM MUST contain a change log artifact that:

- Records changes to previous IEPD or EIEM schemas that this MPD represents.
- Begins with the substring "`changelog`".
- Resides in the root directory of the MPD.

This rule does not specify the format for an IEPD or EIEM change log. This is left to the discretion of the author. While use of `mpd-changelog.xsd` is encouraged for IEPD and EIEM schemas, it is not required. Relaxing the change log format encourages and facilitates easier and more rapid development. IEPDs and EIEMs are developed by a variety of NIEM domains, organizations, and users; and they are intended to specify implementable exchanges. As a result, IEPDs and EIEMs usually contain many documentation artifacts as well as various machine readable artifacts in various formats. A consistent standard change log for these artifacts is very difficult to specify strictly.

The initial version of an IEPD or EIEM would not likely require a change log. However, for consistency of validation and to help facilitate automatic processing of IEPDs and EIEMs by tools:

[Rule 4-13] The initial version of an IEPD or EIEM MUST contain a change log artifact with at least one entry for its creation date.

Finally, if the `mpd-changelog.xsd` specification is used for IEPD/EIEM schema changes, then it is potentially possible that such an MPD will need a second change log if the author wants to accommodate documentation or other changes not related to schemas (since `mpd-changelog.xsd` cannot be extended to accommodate such changes). If this is the case, then the following rule applies:

[Rule 4-13.1] If an IEPD or EIEM contains more than one change log artifact, then each change log artifact MUST:

- Have a file name that begins with the substring "`changelog`".
- Reside in the MPD root directory .

4.4 Master Document

A master document is only required for IEPDs and EIEMs since these MPDs are allowed the greatest design flexibility, can be developed and implemented different ways, and are not centrally managed. On the other hand, releases and domain updates have fairly restrictive rules to obey, standard documentation for how to use them, and are centrally managed.

[Rule 4-14] An IEPD or an EIEM MUST contain a master document located in the MPD root directory whose filename begins with the substring "`master-document`".

The master document may replicate some of the metadata in the catalog. However, the catalog is intentionally designed to be efficient, easily to parse, and minimal. It is intended for search, discovery, registration, and Web page generation, and not to support various types of detailed technical prose often required for human understanding.

The primary purposes of the master document include:

- To help facilitate understanding and reuse of IEPDs and EIEMs.
- To ensure that fundamental and detailed business-level information about an IEPD or EIEM are documented for human understanding.
- To ensure an author has considered and conveys such fundamental information.
- To provide an initial source within an IEPD or EIEM for human consumable documentation (similar to a `"readme"` file) and/or references to other business or technical documentation needed for understanding.

The master document is not intended to be the only source of written documentation for an MPD (though it can be). It is expected to be the initial resource that references and coordinates all others whether physically present in the MPD or linked by reference. Consider the master document to be similar to a `"readme"` file.

Many organizations have their own customized formats and operating procedures for documenting their work and products. This specification does not attempt to standardize master document format or layout. Only the file name and relative path within the MPD are strictly specified. The following section will also describe in general terms minimal content that should be in the master document. Adherence to such a requirement is certainly a subjective judgment.

4.4.1 Master Document Content

This section is neither a cookbook nor a normative specification for a master document. It simply suggests typical topics that a master document should or might address, and provides some non-normative guidance.

The master document should help another user or developer to understand the content and use of an IEPD or EIEM, as well as determine potential for reuse or adaptation. It should describe what implementers need to understand and what the author considers is important to understanding an IEPD or EIEM. There is no limit or constraint on its content.

At a minimum, the master document should contain several fundamental elements of information about the MPD:

- Purpose of this MPD and the IEM contained therein.
- Scope of its deployment, usage, and information content.
- Business value and rationale for developing it.
- Type of information it is intended to exchange (in business terms).
- Identification of senders and receivers (or the types of senders and receivers).
- Typical interactions between senders, receivers, and systems.

- References to other documentation within the MPD, and links to external documents that may be needed to understand and implement it.

> **[Rule 4-15]** A NIEM IEPD or EIEM master document SHOULD (at a minimum) describe the MPD purpose, scope, business value, exchange information, senders/receivers, interactions, and references to other documentation.

MPD documentation types and formats will vary with the methodologies and tools used to develop them. Most of this documentation will likely be typical of that generated for data-oriented software projects. Some documentation may only require sections in the master document. Other documentation may be more suitable as separate artifacts that are referenced and explained by a section in the master document (such as diagrams, large tables, data dictionaries, test results/reports, etc.). The following are some common examples of sections in or separate artifacts associated with the master document:

- Use cases
- Business processes
- Business requirements
- Business rules
- Metadata security considerations
- Domain model design specifications and documentation and/or diagrams
- Data dictionary
- Testing and conformance
- Development tools and methodologies used
- Implementation guidance (An IEPD is meant to be implemented, so this section is very important, particularly in the case of a complex IEPD that uses multiple subsets, exchange schemas, and/or IEP root elements.)
    - Security considerations (for protecting sensitive or classified information)
    - Privacy considerations (for example, Personal Identifiable Information or PPI)
    - Types of implementation
    - If an IEPD employs multiple subsets (either from the same release or from different releases):
        - Where and how are these used?
        - What are the caveats regarding duplicative data components?
        - How are these coordinated in the implementation?
    - If an IEPD employs multiple exchange schemas and/or exchange schemas with multiple root elements:
        - What is the purpose of each (exchange and root) and when is it used?

- How are these coordinated during the runtime preparation and transmission of IEPs?

Authors are also encouraged to include an executive summary, especially for particularly lengthy master documents.

5 Optional MPD Artifacts

Aside from the required artifacts, MPD content is relatively flexible. A variety of other optional documentation files may be incorporated into an MPD. When applicable, these may include (but are not limited to) files that describe or explain:

- Implementation details (hardware, software, configuration, etc.)
- Use of multiple root elements
- Use of multiple subsets or mixed releases
- How to use/reuse an MPD for various purposes (such as Web Services)
- Rationales and/or business purposes

In addition to documentation artifacts, a variety of other optional files can be added to an MPD to facilitate tool support and make reuse, adaptation, and/or implementation easier. These are often files that are inputs to or outputs from software tools. Examples include content diagrams, content models in tool-specific formats, and business rules (either formal or informal representations).

Another optional artifact that is encouraged, especially for IEPDs, is a conformance report or other evidence of quality. In the future, as NIEM processes and tools mature, a conformance and quality reports and a corresponding certificate may become required artifacts. For now, inclusion of a conformance report is at the discretion of the author or sponsor. Though clearly, such reports can only increase confidence in MPDs that contain them.

An MPD author may include any files believed to be useful to understand, implement, reuse, and/or adapt an MPD. However, **[Rule 4-6]** always applies – any file included as an artifact in an MPD must also be identified with an entry in the MPD catalog.

An MPD of relatively simple content and scope may only need to contain the minimum mandatory artifacts required by this specification in order to understand and implement it. (See **Appendix F: MPD Artifacts** for a listing of the mandatory and optional artifacts for each type of MPD.)

Files vary widely in format and are often specific to the tools an author uses to parse, consume, or output them. Therefore, if tool-specific files are included in an MPD, it is also a good practice to include copies of those files in formats that display with standard Web browsers or other cost-free, publicly available viewing tools. Common formats include, but are not limited to ASCII text, CSV, HTML, XHTML, JPG, GIF, PNG, and PDF. This guidance is intended to encourage and facilitate maximal sharing and distribution of MPDs; it does not prohibit and is not intended to discourage the inclusion of other file formats.

In particular, this specification does not discourage use of Microsoft (MS) file formats for documentation and other optional artifacts. A number of MS Office products are commonly used in most large organizations. Furthermore, viewers are publicly available for many MS products, including Word (DOC), Excel (XLS), PowerPoint (PPT), Access (MDB), and Visio (VSD). (See http://office.microsoft.com/en-us/downloads/office-online-file-converters-and-viewers-HA001044981.aspx)

6 Directory Organization, Packaging, Other Criteria

An MPD is a logical set of electronic files aggregated and organized to fulfill a specific purpose in NIEM. Directory organization and packaging of an MPD should be designed around major themes in NIEM: reuse, sharing, interoperability, and efficiency.

This rule is also applicable to all MPDs:

> **[Rule 6-1]** An MPD is packaged as a single compressed archive of files that represents a sub-tree of a file system in standard **[PK-ZIP]** format. This archive MUST preserve and store the logical directory structure intended by its author.

IEM schema artifacts must be valid for both XML Schema and NIEM:

> **[Rule 6-2]** Within an MPD archive, all XSD and XML artifacts MUST be valid against and follow all rules for their respective **[NIEM-NDR**] conformance targets (i.e., subset, constraint, extension, exchange, reference schemas, and XML instances); this includes being well-formed and valid XML Schema documents.

NIEM releases, core updates, and domain updates follow a relatively consistent approach to directory organization **[NIEM-DomainUpdate]**. But there are many ways to organize IEPD and EIEM directories that may depend on a number of factors including (not limited to) business purpose and complexity. For this reason, strict rules for IEPD and EIEM directory structure are difficult to establish. Therefore, IEPD and EIEM authors may create their own logical directory structures. However, for consistency and efficiency:

> Definition: **MPD root directory** – The top level file directory relative to all MPD artifacts and subdirectories.

> **[Rule 6-3]** An MPD archive MUST uncompress (unzip) to a one and only one MPD root directory.

[Rule 6-3] ensures that:

- Unpacking an MPD archive will not scatter its contents on a storage device.
- A common starting point always exists to explore or use any MPD.

- Catalog and change log artifacts will always be found in the MPD root directory (as a result of **[Rule 4-1]** and **[Rule 4-12]**).

6.1 MPD File Name Syntax

As previously stated, the MPD Specification is intended to facilitate tool support for processing MPDs. Given a tool must process an MPD, providing it basic information about the MPD as early as possible will help to reduce processing time and complexity. So, if the class and version of an MPD archive could be easily identified by its file name, then a tool would not have to immediately open the archive and process the catalog just to determine this information. Of course, ultimately, to do anything useful, a tool will have to open the MPD archive and process its catalog. However, a standard file name syntax would allow a tool to search through a set of MPD archives to find a particular MPD name, version, or class without having to open (unzip) each. The following rules apply:

[Rule 6-3a] An MPD archive file MUST use file name syntax defined by the regular expression:

```
mpd-filename ::= name '-' version '.'class'.zip'
```

Where:

```
name    ::= alphanum ((alphanum | special)* alphanum)?
alphanum ::= [a-zA-Z0-9]
special  ::='.'|'-'|'_'
version  ::= digit+ ('.' digit+)* (status digit+)?
digit    ::= [0-9]
status   ::= 'alpha' | 'beta' | 'rc' | 'rev'
class    ::= 'rel' | 'cu' | 'du' | 'iepd' | 'eiem'
```

All alpha characters SHOULD be lower case to reduce the risk of complications across various file systems. See **[Rule 4-3]** for an explanation of the `status` options.

(The regular expression notation used above is from XML 1.0 (Fifth Edition): http://www.w3.org/TR/2008/REC-xml-20081126/#sec-notation)

Obviously, the set of class values corresponds to release, core update, domain update, IEPD, and EIEM respectively. A valid IEPD file name corresponding to the example in **Appendix C: Sample MPD Catalog Instance** would be: `Planning_Order-1.0.3rev2.iepd.zip`

Checking this Appendix you will find that this example obeys the following two rules:

[Rule 6-3b] Within an MPD, the `<name>` and `<version>` substrings in the file name MUST match exactly the values for attributes `mpdName` and `mpdVersionID` within its `catalog.xml` artifact.

[Rule 6-3c] Within an MPD, the `<class>` substring in the file name MUST correctly correspond to the value for the attribute `mpdClassCode` within `catalog.xml`. Correct correspondence is:

IF file name `<class>` =	THEN `catalog.xml mpdClassCode` =
`rel`	`release`
`cu`	`core-update`
`du`	`domain-update`
`iepd`	`iepd`
`eiem`	`eiem`

In HTTP-based Web Services environments, the MIME type designation of a MPD archive is important to facilitate processing by service consumers.

[Rule 6-3d] When represented on the Internet, an MPD archive SHOULD use the following MIME Type:

`application/zip+<class>` where

`<class>` is one member from the set `{rel, cu, du, iepd, eiem}`

Use of the generic zip MIME type `application/zip` is allowed, but discouraged. No other MIME types are allowed when representing MPD archives.

6.2 Artifact Links to Other Resources

The **[NIEM-NDR]** requires that each `xsd:import` in a NIEM schema contain a `schemaLocation` attribute with either an absolute or relative path reference that resolves to the correct imported schema. However, this specification restricts an MPD import to a relative path reference that resolves to the correct schema within the MPD itself. It is important to understand that the URI scheme previously discussed in Section **4.2.3 URI Scheme for MPD Artifacts** should be used only to identify relationships among and provide source links to external schemas being reused. It is not sufficient to allow references or links to such schemas stand in for a physical copy.

Regardless of references to external sources or internal MPD directory organization, each schema `xsd:import` must adhere to the following rule:

> **[Rule 6-4]** Within an MPD archive, the value of each `xsd:import schemaLocation` attribute MUST be a relative path reference that resolves to the correct schema within the sub-tree.

The implication of this rule is a guarantee that all schema artifacts necessary to define, validate, and use an MPD are physically present within that MPD without requiring modifications to `xsd:import schemaLocation` attributes within schemas. In accordance with the **[NIEM-NDR],** if MPD schemas are moved to an operational environment for implementation, validation, or other purposes, then absolute references may replace relative path references when needed. When absolute references to Internet resources are required:

> **[Rule 6-5]** Absolute references to Internet resources MUST use a well-known transfer protocol (`http, https, ftp, ftps`) and MUST resolve (If applicable, documentation that describes how to resolve with security, account, and/or password issues MUST be included).

Releases, core updates, and domain updates must adhere to packaging rules primarily to enable development tools to process them consistently and efficiently. The NIEM PMO controls the format and documentation for these MPDs and publishes them at http://release.niem.gov/niem./. However, many different organizations author IEPDs and EIEMs. As such, they may be distributed, published in repositories (possibly to a limited community), and reused by others. Furthermore, EIEMs are the basis for families of IEPDs. Therefore, it is important that both of these MPD classes are well documented for understanding and use.

> **[Rule 6-6]** A published IEPD MUST contain all documents necessary to understand it and allow it to be implemented correctly.

> **[Rule 6-7]** A published IEPD MUST link (through its catalog) to any EIEM it is based on.

The **[NIEM-NDR]** explains how NIEM employs a special type adaption mechanism to encapsulate and use other standards (e.g., geospatial and emergency management standards) in their native forms that are not NIEM-conforming. Other standards may use `xsd:import` without requiring `schemaLocation` attributes (instead, relying only on the namespace). These standards may also use `xsd:include` which is disallowed by NIEM. When standards external to NIEM are required within NIEM MPDs, the following rule applies:

> **[Rule 6-8]** Within an MPD archive, if non-NIEM-conforming schemas from other standards are used and referenced within an MPD, then all `xsd:import, xsd:include,` and `xsd:redefine` constructs used within those schemas MUST be modified as needed to have a value for the `schemaLocation` attribute that is a relative path reference that resolves to the correct schema within the sub-tree.

For the case of non-NIEM-conforming schemas, this rule ensures that all schemas (or corresponding artifacts and namespaces) from other standards required for definition, validation, and use of the MPD are present within the archive.

XML schemas are the heart of MPDs since they formally specify normative structure and semantics for data components. However, in general, an MPD is a closed set of artifacts. This means that all hyperlink references within artifacts should resolve to the appropriate artifact.

> **[Rule 6-9]** Within any artifact of an MPD archive, any direct reference to another resource (i.e., another artifact such as an image, schema, stylesheet, etc.) that is required to process or display an artifact SHOULD exist within the archive at the location specified by that reference.

This means that MPD artifacts, including documentation artifacts, should be complete. For example, if an HTML document contains a hyperlink reference (`href`) to a schema (`xsd`) or stylesheet (`xsl`) that is part of the MPD, then the schema file associated with that hyperlink should be present within the MPD; likewise for a sourced (`src`) image. Authors should exercise good judgment with this rule. For example, it does not require an MPD to contain copies of all cited documents from a table of references if it contains hyperlinks to those documents. The key operating words in this rule are: "another resource … required to process or display an artifact SHOULD exist within the archive."

6.3 Duplication of Artifacts

Within an MPD, the replication of files or entire file sets should be avoided. However, replication is allowed when a reasonable rationale exists. In some cases, file replication may make it easier to use, validate, implement, or automatically process an MPD. For example, multiple subsets may overlap with many identical schemas. Yet, it may be easier or even necessary to allow this form of duplication to accommodate a validation tool, rather than removing duplicate schemas, and forcing the tool to use the catalog to identify required artifacts. `mpd-catalog.xsd` is designed to track duplicate artifacts (`File` or `FileSet`), as well as, reference a single `File` artifact from multiple `FileSet` artifacts.

6.4 Non-normative Guidance for Directories

Aside from the rules above, this specification does not impose additional constraints on an IEPD or EIEM author's freedom to organize directory structure. This is why the catalog is required to list every artifact, along with its locally unique identifier, relative path name, nature, and purpose. This enables a machine to find every artifact regardless of its location in an MPD archive and know exactly what it is. The catalog artifact always takes precedence over the directory structure of an IEPD or EIEM.

Non-normative guidance for directory structuring may be useful to authors for a relatively simple IEPD or EIEM with a single subset, extension set, constraint set, and exchange schema set. The following general guidance has been common practice for IEPD directories:

- Create a root directory for the IEPD from the name and version identifier of the IEPD. For example `"my-iepd-3.2.4rev2"`.

- Per **[Rule 4-1]** and **[Rule 4-12]**, `catalog.xml` and the change log (XML file format is not required for IEPD or EIEM change logs) must reside in the root directory.
- Maintain XML stylesheets used with the catalog and change log in the MPD root directory.
- Maintain each subset organized as generated by the Schema Subset Generation Tool (SSGT). The reason for maintaining the SSGT grouping is that the SSGT ensures that all `xsd:import schemaLocation` attributes contain relative path names that are correctly coordinated with the directory structure.
- If derived from a schema subset, maintain the constraint schema set grouped as the subset from which it was derived (for the same reason as above).
- Establish a subdirectory of the MPD root directory with the name `"XMLschemas"`. Within this subdirectory:
    - Maintain each subset in a subdirectory with a name that contains the substring `"subset"`. Maintain and correlate a wantlist with the subset it represents. To do so, change a wantlist filename if appropriate.
    - Maintain each constraint schema set (or all constraint schemas if appropriate) in a subdirectory with a name that contains the substring `"constraint"`.
    - Maintain each extension schema set (or all extension schemas if appropriate) in a subdirectory with a name that contains the substring `"extension"`.
    - Maintain each exchange schema in a subdirectory with a name that contains the substring `"exchange"`.
- Maintain all documentation in a subdirectory of the MPD root directory with a name that contains the substring `"documentation"`. Create additional documentation subdirectories inside this one as needed.
- Maintain all sample XML instances that validate to the schemas in a subdirectory of the MPD root directory with a name that contains the substring `"XMLsamples"`. This subdirectory can also contain any XML stylesheets (XSL) used with the sample instances.
- Maintain tool specific files (inputs and outputs) in a subdirectory of the MPD root directory with a name that contains the substring `"library"`.

The guidance above results in an IEPD directory structure illustrated below in Figure 6-1. Obviously, there are many other ways to organize for more complex business requirements in which multiple releases, subsets, constraint sets, core updates, and domain updates are employed in a single IEPD.

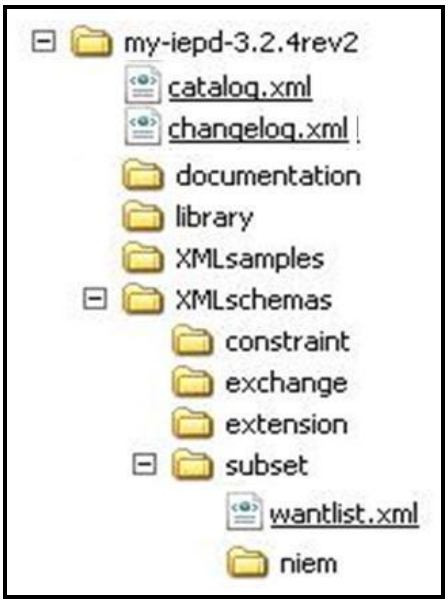


Figure 6-1. IEPD sample directory structure

Appendix A: MPD Catalog Schema

`mpd-catalog-1.0.xsd` (version 1.0)

URI: http://reference.niem.gov/niem/resource/mpd/catalog/1.0/

Note: Download the latest text version of this file using its `http` URI above (also its target namespace). Files in appendices are intended for browsing only; do not copy/paste for testing or processing.

```
<?xml version="1.0" encoding="UTF-8"?>

<schema
    attributeFormDefault="qualified"
    elementFormDefault="qualified"

targetNamespace="http://reference.niem.gov/niem/resource/mpd/catalog/1.0/"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:ca="http://reference.niem.gov/niem/resource/mpd/catalog/1.0/"
    version="1.0">

    <annotation>
      <documentation>
          Model Package Description (MPD) Catalog;
          XML document element = Catalog;
          This schema defines the catalog.xml artifact
          for Model Package Descriptions (MPD) in NIEM.
          MPDs include:
            NIEM releases, core updates, domain updates,
            NIEM Information Exchange Package Documentation (IEPD),
            and NIEM Enterprise Information Exchange Models (EIEM).
          The purpose of this schema is to define/declare metadata
          for NIEM MPDs to search, discover, and process MPDs efficiently.
          Instances of this schema are not NIEM data exchanges.  For
          this reason this schema is not NIEM conforming, even though
          type definitions and element/attribute declarations generally
          conform to NIEM NDR naming rules.
      </documentation>
    </annotation>

    <element name="Catalog" type="ca:CatalogType"/> <!-- Root element -->

<!-- Artifacts and structures ========================================== -->

    <complexType name="CatalogType">
        <sequence>
          <element ref="ca:Artifact" minOccurs="1" maxOccurs="unbounded"/>
          <element ref="ca:Metadata" minOccurs="1" maxOccurs="1"/>
        </sequence>
        <attribute      ref="ca:mpdURI"         use="required"/>
        <attribute      ref="ca:mpdClassCode"   use="required"/>
        <attribute      ref="ca:mpdName"        use="required"/>
        <attribute      ref="ca:mpdVersionID"   use="required"/>
        <attribute      ref="ca:descriptionText" use="optional"/>
    </complexType>
```

```
    <element name="Artifact" abstract="true"/>

    <element name="FileSet" type="ca:FileSetType"
substitutionGroup="ca:Artifact"/>

    <complexType name="FileSetType">
      <sequence>
          <element ref="ca:File" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute  ref="ca:id"                   use="required"/>
      <attribute  ref="ca:externalURI"          use="optional"/>
      <attribute  ref="ca:natureURI"            use="required"/>
      <attribute  ref="ca:purposeURI"           use="required"/>
      <attribute  ref="ca:descriptionText"      use="optional"/>
      <attribute  ref="ca:files"                use="optional"/>
    </complexType>

    <element name="Folder" type="ca:FolderType"
substitutionGroup="ca:Artifact"/>

    <complexType name="FolderType">
      <attribute  ref="ca:id"                   use="optional"/>
      <attribute  ref="ca:relativePathName"     use="required"/>
      <attribute  ref="ca:descriptionText"      use="optional"/>
    </complexType>

    <element name="File" type="ca:FileType" substitutionGroup="ca:Artifact"/>

    <complexType name="FileType">
      <attribute  ref="ca:id"                   use="required"/>
      <attribute  ref="ca:externalURI"          use="optional"/>
      <attribute  ref="ca:relativePathName"     use="required"/>
      <attribute  ref="ca:natureURI"            use="required"/>
      <attribute  ref="ca:purposeURI"           use="required"/>
      <attribute  ref="ca:descriptionText"      use="optional"/>
    </complexType>

<!-- Metadata ============================================================ -->

    <element name="Metadata" type="ca:MetadataType"/>

    <complexType name="MetadataType">
      <sequence>
          <element ref="ca:SecurityMarkingText" minOccurs="1" maxOccurs="1"/>
          <element ref="ca:CreationDate"  minOccurs="1" maxOccurs="1"/>
          <element ref="ca:LastRevisionDate"    minOccurs="0" maxOccurs="1"/>
          <element ref="ca:NextRevisionDate"    minOccurs="0" maxOccurs="1"/>
          <element ref="ca:StatusText"          minOccurs="0" maxOccurs="1"/>
          <element ref="ca:Relationship" minOccurs="0"
maxOccurs="unbounded"/>
          <element ref="ca:KeywordText" minOccurs="1" maxOccurs="unbounded"/>
          <element ref="ca:DomainText"  minOccurs="1" maxOccurs="unbounded"/>
          <element ref="ca:PurposeText" minOccurs="0" maxOccurs="unbounded"/>
          <element ref="ca:ExchangePatternText" minOccurs="0"
maxOccurs="unbounded"/>
```

```
          <element ref="ca:ExchangePartnerName" minOccurs="0"
maxOccurs="unbounded"/>
          <element ref="ca:AuthoritativeSource" minOccurs="1" maxOccurs="1"/>
      </sequence>
    </complexType>

    <element name="AuthoritativeSource" type="ca:AuthoritativeSourceType"/>

    <complexType name="AuthoritativeSourceType">
      <sequence>
          <element ref="ca:ASName"        minOccurs="1"   maxOccurs="1"/>
          <element ref="ca:ASAddressText" minOccurs="0"   maxOccurs="1"/>
          <element ref="ca:ASWebSiteURL"  minOccurs="0"   maxOccurs="1"/>
          <element ref="ca:POC"     minOccurs="1"   maxOccurs="unbounded"/>
      </sequence>
    </complexType>

    <element name="POC" type="ca:POCType"/>

    <complexType name="POCType">
      <sequence>
         <element ref="ca:POCName"  minOccurs="1"  maxOccurs="1"/>
         <element ref="ca:POCEmail" minOccurs="1"  maxOccurs="unbounded"/>
         <element ref="ca:POCTelephone" minOccurs="1"
maxOccurs="unbounded"/>
      </sequence>
    </complexType>

    <element name="SecurityMarkingText"   type="string"
      default="unclassified"/>
    <element name="CreationDate"          type="date"/>
    <element name="LastRevisionDate"      type="date"/>
    <element name="NextRevisionDate"      type="date"/>
    <element name="StatusText"            type="string"/>
    <element name="Relationship"          type="ca:RelationshipType"/>
    <element name="KeywordText"           type="string"/>
    <element name="DomainText"            type="string"/>
    <element name="PurposeText"           type="string"/>
    <element name="ExchangePatternText"   type="string"/>
    <element name="ExchangePartnerName"   type="string"/>

    <element name="ASName"                type="string"/>
    <element name="ASAddressText"         type="string"/>
    <element name="ASWebSiteURL"          type="anyURI"/>
    <element name="POCName"               type="string"/>
    <element name="POCEmail"              type="string"/>
    <element name="POCTelephone"          type="string"/>

    <complexType name="RelationshipType">
      <attribute ref="ca:relationshipCode"    use="required"/>
      <attribute ref="ca:resourceURI"         use="required"/>
      <attribute ref="ca:descriptionText"     use="optional"/>
    </complexType>
```

```
<!-- Primitives ============================================================ -->
    <attribute name="id"                  type="ID"/>
    <attribute name="mpdURI"              type="ca:URISimpleType"/>
    <attribute name="mpdClassCode"        type="ca:MPDClassCodeSimpleType"/>
    <attribute name="mpdName"             type="string"/>
    <attribute name="mpdVersionID"        type="ca:MPDVersionIDSimpleType"/>
    <attribute name="externalURI"         type="ca:URISimpleType"/>
    <attribute name="relativePathName"    type="string"/>
    <attribute name="natureURI"           type="ca:URISimpleType"/>
    <attribute name="purposeURI"          type="ca:URISimpleType"/>
    <attribute name="descriptionText"     type="string"/>
    <attribute name="files"               type="IDREFS"/>
    <attribute name="resourceURI"         type="ca:URISimpleType"/>
    <attribute name="relationshipCode" type="ca:RelationshipCodeSimpleType"/>

    <simpleType name="MPDClassCodeSimpleType">
      <restriction base="token">
          <enumeration value="release"/>
          <enumeration value="core-update"/>
          <enumeration value="domain-update"/>
          <enumeration value="iepd"/>
          <enumeration value="eiem"/>
      </restriction>
    </simpleType>

    <simpleType name="RelationshipCodeSimpleType">
      <restriction base="token">
          <enumeration value="version_of"/>
          <enumeration value="specializes"/>
          <enumeration value="generalizes"/>
          <enumeration value="supersedes"/>
          <enumeration value="deprecates"/>
          <enumeration value="adapts"/>
          <enumeration value="updates"/>
          <enumeration value="conforms_to"/>
      </restriction>
    </simpleType>

    <simpleType name="MPDVersionIDSimpleType">
        <restriction base="token">
          <pattern value="[0-9]+(\.[0-9]+)*((alpha|beta|rc|rev)[0-9]+)?"/>
          <minLength value="1"/>
        </restriction>
    </simpleType>

    <simpleType name="URISimpleType">
        <union memberTypes="anyURI ca:SafeCurieSimpleType"/>
    </simpleType>

    <simpleType name="SafeCurieSimpleType">
        <restriction base="string">
            <pattern value="\[(([\i-[:]][\c-[:]]*)?:)?.+\]"/>
            <minLength value="3"/>
        </restriction>
    </simpleType>
</schema>
```

Appendix B: Catalog Data Dictionary

The following table is a data dictionary for the **Appendix A: MPD Catalog Schema**.

The Cardinality column lists the minimum and maximum cardinality for each element or attribute in an MPD Catalog, where U = Unbounded. Grayed rows are the code values for the attribute `relationshipCode`. Cardinality is not applicable to code values. Typical cardinalities are interpreted as follows:

Min, Max	The associated element or attribute is …
1, 1	required; not optional; one and only one is allowed
1, U	required; not optional; one or any number is allowed; repeatable
0, 1	optional; zero or exactly one is allowed (0 or 1)
0, U	optional; zero or any number is allowed; repeatable

Name	Definition	Cardinality Min, Max
`Catalog`	A dataset that describes MPD artifacts and metadata. (the root element)	1,1
`id`	A locally unique (within a catalog file) identifier for an artifact or set of artifacts in an MPD.	1,1
`mpdURI`	A globally unique identifier (a URI) for an MPD.	1,1
`mpdClassCode`	A type of MPD (i.e., type of the IEM it contains); its values are drawn from the following set: `{release, core-update, domain-update, iepd, eiem}`	1,1
`mpdName`	A label or title for an MPD.	1,1
`descriptionText`	A statement that provides an explanation or additional detail.	0,1
`Artifact`	A file or file set in an MPD.	1,U
`FileSet`	A set of artifacts that are grouped for a purpose.	0,U
`File`	An electronic file stored on a computer system.	0,U
`Folder`	A file directory in an MPD.	0,U
`relativePathName`	A locally unique string that represents the location of the file (including the file name) within an MPD relative to the MPD root directory.	1,1
`mpdVersionID`	An identifier that distinguishes releases of a given MPD.	1,1
`externalURI`	A globally unique identifier for an artifact in another MPD that is reused by this MPD.	0,1
`purposeURI`	A formal NIEM vocabulary term that defines the usage of an MPD artifact.	1,1
`natureURI`	A formal NIEM vocabulary term that defines the file type of an MPD artifact.	1,1

Name	Definition	Cardinality Min, Max
files	A list of local MPD file identifiers (id attributes for File entries).	0,1
SecurityMarkingText	A label that defines how this MPD must be handled or can be distributed to protect the information it contains.	1,1
CreationDate	A date this MPD was published.	1,1
LastRevisionDate	A date this MPD was revised most recently (i.e., CreationDate of previous version).	0,1
NextRevisionDate	A projected date on which this MPD is expected to be revised (i.e., an estimate).	0,1
StatusText	A description of the current state of this MPD in development; may also project future plans for the MPD.	0,1
Relationship	A reference to another MPD related to this MPD.	0,U
resourceURI	A globally unique identifier for the MPD that this MPD relates to.	1,U
relationshipCode	A classification or reason for the connectedness between this MPD and the resource referenced in resourceURI. (See **relationshipCode** enumerated values and applicability table immediately following this table.)	1,U
descriptionText	A more detailed or specific explanation of the relationship between this MPD and the resource referenced in resourceURI.	0,U
KeywordText	A common alias, term, or phrase that would help to facilitate search and discovery of this MPD.	1,U
DomainText	A description of the environment or NIEM Domain in which this MPD is applicable or used.	1,U
PurposeText	A description of the intended usage and reason for which an MPD exists.	0,U
ExchangePatternText	A description of a transactional or design pattern used for this IEPD (generally, applicable to IEPDs only).	0,U
ExchangePartnerName	A name of an entity or organization that uses this MPD.	0,U
AuthoritativeSource	A set of metadata that describes the entity responsible for configuration and change management for this MPD.	1,1
ASName	A name for the current MPD authoritative source; could be the author, creator, sponsor, etc. (organization or person name).	1,1
ASAddressText	A description of the location of the authoritative source for the MPD.	0,1

Name	Definition	Cardinality Min, Max
ASWebSiteURL	A URL for the Web site of the authoritative source for the MPD.	0,1
POC	A set of metadata used to contact the authoritative source for an MPD.	1,U
POCName	A name for a person, position, or title.	1,1
POCEmail	An email address.	1,U
POCTelephone	A telephone number.	1,U

relationshipCode		Applicable to				
Value	Definition	IEPD	EIEM	CU	DU	Release
version_of	A relationshipCode value for indicating that this MPD is a different version of the MPD referenced in resourceURI. This code value is only needed in cases where significant name changes might obscure the relationship to the previous version. For example, NIEM Justice 4.1 is a version of GJXDM 3.0.3.	X	X			
generalizes	A relationshipCode value for indicating that this MPD is a generalization of the MPD referenced in resourceURI. This value is the inverse of specializes.	X				
specializes	A relationshipCode value for indicating that this MPD is a specialization of the MPD referenced in resourceURI. This value is the inverse of generalizes.	X				
supersedes	A relationshipCode value for indicating that this MPD replaces the MPD referenced in resourceURI.	X	X		X	X
deprecates	A relationshipCode value for indicating that content in this MPD is preferred over content in the MPD referenced in resourceURI; and at some time in the future will supersede the MPD referenced in resourceURI	X	X	X	X	
adapts	A relationshipCode value for indicating that this MPD is an adaptation of the MPD referenced in resourceURI.	X	X			
conforms_to	A relationshipCode value for indicating that this MPD conforms to the specification or standard referenced in resourceURI.	X	X	X	X	X
updates	A relationshipCode value for indicating that this MPD is an incremental update to the resource referenced in resourceURI. Used by a core or domain update to identify the domain schema in a NIEM release being incrementally updated (not replaced).			X	X	

Appendix C: Sample MPD Catalog Instance

`catalog.xml` (part of the example that corresponds to version 1.0)

URI:
http://reference.niem.gov/niem/resource/mpd/catalog/1.0/example/

Note: Download the latest text version of this file using its `http` URI above. Files in appendices are intended for browsing only; do not copy/paste for testing or processing.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="./catalog.xsl"?>

<!-- This IEPD sample catalog is fictitious and abbreviated.  However,
it illustrates use of various options from mpd-catalog-1.0.xsd; including
reuse of the File element with IDREFS in the "files" attribute, use of
safe CURIEs, full URIs, Folders, and various metadata. -->

<ca:Catalog

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ca="http://reference.niem.gov/niem/resource/mpd/catalog/1.0/"

xmlns:p="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#"
    xmlns:n="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#"
    xmlns:yy="http://my.iepd.program.org/new/"
    xmlns:zz="http://zoom.com/services/planning-order/"


xsi:schemaLocation="http://reference.niem.gov/niem/resource/mpd/catalog/1.0/
mpd-catalog.xsd"

    ca:mpdURI="http://myiepd.org/p-ord/"
    ca:mpdClassCode="iepd"
    ca:mpdName="Planning_Order"
    ca:mpdVersionID="1.0.3rev2"
    ca:descriptionText="A Planning Order notifies subordinate agencies
                  to initiate planning for contingency operations." >

    <ca:File ca:id="cl0"
      ca:relativePathName="./changelog.txt"
      ca:natureURI="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/n
ature#text"
      ca:purposeURI="[p:changelog]"
      ca:descriptionText="History of changes to Planning Order"/>

    <ca:File ca:id="d2"
      ca:relativePathName="./documentation/master-document.docx"
      ca:natureURI="[n:doc]"
      ca:purposeURI="[p:master-document]"
      ca:descriptionText="Instructions for implementing Planning Order"/>

    <ca:FileSet ca:id="es1"
      ca:externalURI="[yy:order-ext0]"
      ca:natureURI="[n:file-set]"
```

```
      ca:purposeURI="[p:extension-schema-set]"
      ca:descriptionText="Extension schema set for IEPD Planning Order">

      <ca:File ca:id="e1"
          ca:relativePathName="./XMLschemas/extension/ext1.xsd"
          ca:natureURI="[n:xsd]"
          ca:purposeURI="[p:extension-schema]"
          ca:descriptionText="my extension schema 1"/>

      <ca:File ca:id="e2"
          ca:relativePathName="./XMLschemas/extension/ext2.xsd"
          ca:natureURI="[n:xsd]"
          ca:purposeURI="[p:extension-schema]"
          ca:descriptionText="my extension schema 2"/>

    </ca:FileSet>

    <ca:FileSet ca:id="ss1"
      ca:natureURI="[n:file-set]"
      ca:purposeURI="[p:subset-schema-set]"
      ca:descriptionText="Subset for IEPD Planning Order">

      <ca:File ca:id="w1"
          ca:externalURI="[yy:order-w1]"
          ca:relativePathName="./XMLschemas/subset/wantlist1.xml"

ca:natureURI="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#
wantlist"

ca:purposeURI="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpos
e#wantlist"
          ca:descriptionText="Wantlist 1 for schema subset ss1"/>

      <ca:File ca:id="s1"
          ca:relativePathName="./XMLschemas/subset/niem/niem-core/2.0/niem-
core.xsd"
          ca:natureURI="[n:xsd]"
          ca:purposeURI="[p:subset-schema]"
          ca:descriptionText="NIEM Core 2.0 (niem-core.xsd)"/>

      <ca:File ca:id="s2"
          ca:relativePathName="./XMLschemas/subset/niem/fbi/2.0/fbi.xsd"
          ca:natureURI="[n:xsd]"
          ca:purposeURI="[p:subset-schema]"
          ca:descriptionText="FBI code lists"/>

    </ca:FileSet>

    <ca:FileSet ca:id="ss2"
      ca:natureURI="[n:file-set]"
      ca:purposeURI="[p:subset-schema-set]"
      ca:descriptionText="A second subset for Planning Order; includes ss1"
      ca:files="s1  s2 ">
```

```
      <ca:File ca:id="w2"
          ca:externalURI="[yy:order-w2]"
          ca:relativePathName="./XMLschemas/subset/wantlist2.xml"

ca:natureURI="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#
wantlist"

ca:purposeURI="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpos
e#wantlist"
          ca:descriptionText="Wantlist 2 for schema subset ss2; includes
ss1"/>

      <ca:File ca:id="s3"
          ca:relativePathName="./XMLschemas/subset/niem/fips_6-4/2.0/fips_6-
4.xsd"

ca:natureURI="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#
xsd"

ca:purposeURI="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpos
e#subset-schema"
          ca:descriptionText="FIPS County codes"/>

    </ca:FileSet>

    <ca:File ca:id="c0"
      ca:relativePathName="./catalog.xml"
      ca:natureURI="[n:catalog]"
      ca:purposeURI="[p:catalog]"
      ca:descriptionText="Catalog: identifies all artifacts in Planning
Order"/>

    <ca:File ca:id="cs0"
      ca:relativePathName="./catalog.xsl"
      ca:natureURI="[n:xslt]"
      ca:purposeURI="[p:display]"
      ca:descriptionText="stylesheet to display a catalog index with
metadata"/>

    <ca:File ca:id="w13"
      ca:externalURI="[zz:order-service13]"
      ca:relativePathName="./library/wsdl/p-order1.wsdl"
      ca:natureURI="[n:wsdl]"
      ca:purposeURI="[p:web-service]"
      ca:descriptionText="Web Service description for IEPD Planning Order"/>

    <ca:File ca:id="r1"
      ca:externalURI="[zz:order-22]"
      ca:relativePathName="./XMLschemas/exchange/myiepd.xsd"
      ca:natureURI="[n:xsd]"
      ca:purposeURI="[p:exchange-schema]"
      ca:descriptionText="Exchange schema for IEPD Planning Order"/>
```

```
<!-- Folders (directories) -->

    <ca:Folder ca:id="f0"
      ca:relativePathName="./"
      ca:descriptionText="ROOT"/>

    <ca:Folder ca:id="f1"
      ca:relativePathName="./XMLschemas/"
      ca:descriptionText="XML SCHEMAS"/>

    <ca:Folder ca:id="f11"
      ca:relativePathName="./XMLschemas/subset/"
      ca:descriptionText="SUBSET"/>

    <ca:Folder ca:id="nc2"
      ca:relativePathName="./XMLschemas/subset/niem/niem-core/2.0/"
      ca:descriptionText="NIEM CORE 2.0"/>

    <ca:Folder ca:id="bi2"
      ca:relativePathName="./XMLschemas/subset/niem/fbi/2.0/"
      ca:descriptionText="FBI"/>

    <ca:Folder ca:id="ip2"
      ca:relativePathName="./XMLschemas/subset/niem/fips_6-4/2.0/"
      ca:descriptionText="FIPS 6-4"/>

    <ca:Folder ca:id="f12"
      ca:relativePathName="./XMLschemas/extension/"
      ca:descriptionText="EXTENSION"/>

    <ca:Folder ca:id="f13"
      ca:relativePathName="./XMLschemas/exchange/"
      ca:descriptionText="EXCHANGE"/>

    <ca:Folder ca:id="f2"
      ca:relativePathName="./library/"
      ca:descriptionText="LIBRARY"/>

    <ca:Folder ca:id="f21"
      ca:relativePathName="./library/wsdl/"
      ca:descriptionText="WEB SERVICES"/>

    <ca:Folder ca:id="f3"
      ca:relativePathName="./documentation/"
      ca:descriptionText="DOCUMENTATION"/>

    <ca:Folder ca:id="f4"
      ca:relativePathName="./XMLsamples/"
      ca:descriptionText="SAMPLES"/>

    <ca:Metadata>

      <ca:SecurityMarkingText>Unclassified/public</ca:SecurityMarkingText>
      <ca:CreationDate>2009-02-16</ca:CreationDate>
```

```
      <ca:StatusText>
          implemented and operational in 25 of 50 states since 2009
      </ca:StatusText>

      <ca:Relationship    ca:relationshipCode="adapts"
                     ca:resourceURI="[zz:pln-ord6]"
                     ca:descriptionText="reuses 4 artifacts" />
      <ca:Relationship
                     ca:relationshipCode="adapts"
                     ca:resourceURI="[zz:pln-ord8]"
                     ca:descriptionText="uses modified extension set" />

      <ca:KeywordText>program plan</ca:KeywordText>
      <ca:KeywordText>contingency</ca:KeywordText>
      <ca:KeywordText>planning</ca:KeywordText>

      <ca:DomainText>Logistics</ca:DomainText>
      <ca:DomainText>Emergency Management</ca:DomainText>
      <ca:PurposeText>intended to initiate execution of planning
cycles</ca:PurposeText>
      <ca:ExchangePatternText>query/response</ca:ExchangePatternText>
      <ca:ExchangePartnerName>FEMA</ca:ExchangePartnerName>
      <ca:ExchangePartnerName>NASCIO</ca:ExchangePartnerName>
      <ca:ExchangePartnerName>82d Abn Div</ca:ExchangePartnerName>

      <ca:AuthoritativeSource>
          <ca:ASName>FEMA</ca:ASName>
          <ca:ASAddressText>30 Dupont Cir, Wash DC, 12345,
USA</ca:ASAddressText>
          <ca:ASWebSiteURL>http://www.fema.gov</ca:ASWebSiteURL>
          <ca:POC>
            <ca:POCName>John Smith</ca:POCName>
            <ca:POCEmail>john.smith@fema.gov</ca:POCEmail>
            <ca:POCTelephone>202-333-1234</ca:POCTelephone>
          </ca:POC>
          <ca:POC>
            <ca:POCName>FEMA, Office of the Chief Administrator</ca:POCName>
            <ca:POCEmail>OCA@fema.gov</ca:POCEmail>
            <ca:POCTelephone>888-333-5678</ca:POCTelephone>
          </ca:POC>
      </ca:AuthoritativeSource>

    </ca:Metadata>

</ca:Catalog>
```

Appendix D: Sample XSLT for a Catalog Index

`catalog.xsl` (part of the example that corresponds to version 1.0)

URI:
http://reference.niem.gov/niem/resource/mpd/catalog/1.0/example/

Note: Download the latest text version of this file using its `http` URI above. Files in appendices are intended for browsing only; do not copy/paste for testing or processing.

This simple example XSLT generates the hypertext links to the catalog entries shown in **Appendix C: Sample MPD Catalog Instance** based on values of the `relativePathName` attribute for each artifact listed in the catalog. An MPD author is free to use or adapt this XSLT, develop a more sophisticated XSLT as necessary, or not include an XSLT at all.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:ca="http://reference.niem.gov/niem/resource/mpd/catalog/1.0/">

<xsl:output method="html" indent="yes"/>

<xsl:template name="put-row">
  <xsl:param name="item"/>
      <tr>
      <td><xsl:value-of select="$item/@ca:id"/></td>
      <td>
          <a>
            <xsl:attribute name="href">
                  <xsl:value-of select="$item/@ca:relativePathName"/>
            </xsl:attribute>
            <xsl:value-of select="$item/@ca:relativePathName"/>
          </a>
      </td>
      <td><xsl:value-of select="$item/@ca:descriptionText"/></td>
      <td><xsl:value-of select="$item/@ca:externalURI"/></td>
    </tr>
</xsl:template>

<xsl:template name="process-ids">
    <xsl:param name="in-string"/>
    <xsl:variable name="normalized" select="normalize-space($in-string)"/>
    <xsl:variable name="head">
      <xsl:choose>
        <xsl:when test="contains($normalized, ' ')">
          <xsl:value-of select="substring-before($normalized, ' ')"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="$normalized"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
```

```
    <xsl:choose>
      <xsl:when test="string-length($head) = 0"/>
      <xsl:otherwise>
        <xsl:variable name="match" select="//*[normalize-space(@ca:id) =
$head]"/>
        <xsl:call-template name="put-row">
          <xsl:with-param name="item" select="$match"/>
        </xsl:call-template>
      <xsl:variable name="tail" select="substring-after(normalize-space($in-
string), ' ')"/>
      <xsl:if test="string-length($tail) &gt; 0">
        <xsl:call-template name="process-ids">
        <xsl:with-param name="in-string" select="$tail" />
        </xsl:call-template>
      </xsl:if>
     </xsl:otherwise>

   </xsl:choose>
</xsl:template>

<xsl:template match="/">
    <html>
      <body><xsl:apply-templates/></body>
    </html>
</xsl:template>

<xsl:template match="/ca:Catalog">
    <h1>IEPD: <xsl:value-of select="@ca:mpdName" /></h1>
    <table border="3">
      <thead style="background-color:maroon; color:white; ">
          <tr>
            <th>Attribute</th>
            <th>Value</th>
          </tr>
      </thead>
      <tr>
          <td>URI</td>
          <td><xsl:value-of select="@ca:mpdURI" /></td>
      </tr>
      <tr>
          <td>Class</td>
          <td><xsl:value-of select="@ca:mpdClassCode" /></td>
      </tr>
      <tr>
          <td>Name</td>
          <td><xsl:value-of select="@ca:mpdName" /></td>
      </tr>
      <tr>
          <td>Version</td>
          <td><xsl:value-of select="@ca:mpdVersionID" /></td>
      </tr>
      <tr>
          <td>Description</td>
          <td><xsl:value-of select="@ca:descriptionText" /></td>
      </tr>
    </table>
```

```
    <h2>All File Artifacts (and Folders; organized by directory)</h2>

    <table border="3">
      <thead style="background-color:maroon; color:white; ">
          <tr>
            <th>Local ID</th>
            <th>Path and File Name</th>
            <th>Description</th>
            <th>External URI</th>
          </tr>
      </thead>

      <xsl:for-each select="//ca:File | //ca:Folder">
      <xsl:sort select="@ca:relativePathName" />
        <xsl:call-template name="put-row">
          <xsl:with-param name="item" select="."/>
        </xsl:call-template>
    </xsl:for-each>

</table>

<h2>All FileSet Artifacts (organized by set)</h2>

    <table border="3">

      <thead style="background-color:maroon; color:white; ">
          <tr>
            <th>Local ID</th>
            <th>Path and File Name</th>
            <th>Description</th>
            <th>External URI</th>
          </tr>
      </thead>

    <xsl:for-each select="ca:FileSet">

<!-- process FileSet element -->

        <xsl:call-template name="put-row">
          <xsl:with-param name="item" select="."/>
        </xsl:call-template>

<!-- process FileSet/@files list -->

      <xsl:call-template name="process-ids">
          <xsl:with-param name="in-string" select="@ca:files" />
      </xsl:call-template>

<!-- process each FileSet/File element -->

    <xsl:for-each select="ca:File">
        <xsl:call-template name="put-row">
          <xsl:with-param name="item" select="."/>
        </xsl:call-template>
      </xsl:for-each>
```

```
    </xsl:for-each>
    </table>

    <h2>Notes about artifacts:</h2>
    <p>Artifacts with External URIs are reused from another MPD.</p>

    <h2>Metadata</h2>

    <table border="3">
      <thead style="background-color:#24ffc0; ">
      <tr>
        <th>Attribute</th>
      <th>Value</th>
      </tr>
      </thead>
      <tr>
      <td>Security Marking</td>
      <td><xsl:value-of select="ca:Metadata/ca:SecurityMarkingText"/></td>
      </tr>
      <tr>
      <td>Creation Date</td>
      <td><xsl:value-of select="ca:Metadata/ca:CreationDate"/></td>
      </tr>
      <tr>
      <td>Last Revision Date</td>
      <td><xsl:value-of select="ca:Metadata/ca:LastRevisionDate"/></td>
      </tr>
      <tr>
      <td>Next Revision Date</td>
      <td><xsl:value-of select="ca:Metadata/ca:NextRevisionDate"/></td>
      </tr>
      <tr>
      <td>Status</td>
      <td><xsl:value-of select="ca:Metadata/ca:StatusText"/></td>
      </tr>

      <xsl:for-each select="ca:Metadata/ca:KeywordText">
      <tr>
          <td>Keyword</td>
          <td><xsl:value-of select="."/></td>
      </tr>
      </xsl:for-each>

      <xsl:for-each select="ca:Metadata/ca:DomainText">
      <tr>
          <td>Domain</td>
          <td><xsl:value-of select="."/></td>
      </tr>
      </xsl:for-each>

      <tr>
        <td>Purpose</td>
        <td>
            <xsl:value-of select="ca:Metadata/ca:PurposeText"/>
        </td>
      </tr>
```

```
      <xsl:for-each select="ca:Metadata/ca:ExchangePatternText">
      <tr>
          <td>Exchange Pattern</td>
          <td><xsl:value-of select="."/></td>
      </tr>
      </xsl:for-each>

      <xsl:for-each select="ca:Metadata/ca:ExchangePartnerName">
      <tr>
          <td>Exchange Partner</td>
          <td><xsl:value-of select="."/></td>
      </tr>
      </xsl:for-each>

    </table>

    <h2>Lineage</h2>

    <table border="3">
      <thead style="background-color:green; color:white; ">
          <tr>
            <th>Relationship</th>
            <th>Resource</th>
            <th>Description</th>
          </tr>
      </thead>
      <xsl:for-each select="ca:Metadata/ca:Relationship">
          <tr>
            <td><xsl:value-of select="@ca:relationshipCode"/></td>
            <td><xsl:value-of select="@ca:resourceURI"/></td>
            <td><xsl:value-of select="@ca:descriptionText"/></td>
          </tr>
      </xsl:for-each>

    </table>

    <h2>Authoritative Source</h2>

    <table border="3">
      <thead style="background-color:blue; color:white; ">
          <tr>
            <th>Attribute</th>
            <th>Value</th>
          </tr>
      </thead>
      <tr>
          <td>Name</td>
          <td>
            <xsl:value-of
select="ca:Metadata/ca:AuthoritativeSource/ca:ASName"/>
          </td>
      </tr>
```

```
      <tr>
          <td>Address</td>
            <td>
              <xsl:value-of
select="ca:Metadata/ca:AuthoritativeSource/ca:ASAddressText"/>
            </td>
      </tr>
      <tr>
          <td>Web Site</td>
          <td>
            <xsl:value-of
select="ca:Metadata/ca:AuthoritativeSource/ca:ASWebSiteURL"/>
          </td>
      </tr>
      <xsl:for-each select="ca:Metadata/ca:AuthoritativeSource/ca:POC">
          <tr>
            <td>POC</td>
            <td>
                <xsl:value-of select="ca:POCName"/>,
                <xsl:value-of select="ca:POCEmail"/>,
                <xsl:value-of select="ca:POCTelephone"/>
            </td>
          </tr>
      </xsl:for-each>
    </table>

</xsl:template>
</xsl:stylesheet>
```

NOTE:

The XSLT example above works with XSLT 1.0 and will execute in most browsers. It does not recompose original URIs from their safe CURIE form. Additional XSLT 1.0 code would be required to convert safe CURIEs to URIs. That code would likely be complex.

XSLT 2.0 makes this conversion easier through the application of customized functions such as the one below (in `curie-functions.xsl.xslext`). However, browsers do not support XSLT 2.0. To use these functions requires an XSLT 2.0 processor such as:

http://saxon.sourceforge.net/

http://www.altova.com/altovaxml.html

If you need to use a browser to see URIs (instead of safe CURIEs), then you have two options:

1. Use XSLT 2.0 to generate an XHTML index artifact and then include it with the catalog.xml artifact within the MPD.
2. Do not use safe CURIEs at all. Instead, use only URIs in the catalog.xml artifact.

`curie-functions.xsl.xslext` (part of the example that corresponds to version 1.0)

URI:
http://reference.niem.gov/niem/resource/mpd/catalog/1.0/example/

Note: Download the latest text version of this file using `http` URI above. Files in appendices are intended for browsing only; do not copy/paste for testing or processing.

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet xmlns:cf="http://ittl.gtri.org/wr24/2010-03-03-1533/curie-
functions/1"
            xmlns:private="http://ittl.gtri.org/wr24/2010-03-03-1533/curie-
functions/1/private"
            xmlns:saxon="http://saxon.sf.net/"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:xslext="http://ittl.gtri.gatech.edu/wr24/2009-03-23-1736/xsl-
extension"
            xmlns="http://www.w3.org/1999/XSL/Transform"
            version="2.0">

  <function name="cf:resolve-curie" as="xs:anyURI?">
      <param name="curie" as="xs:string"/>
      <param name="element" as="element()"/>
      <choose>
        <when test="not(matches($curie, '[^:]*:[^:]*'))">
          <message>Curie has bad format. Value was &quot;<value-of
select="$curie"/>&quot;.</message>
        </when>
        <otherwise>
          <variable name="prefix" select="substring-before($curie, ':')"/>
          <variable name="rest" select="substring-after($curie, ':')"/>
          <variable name="base-URI" select="namespace-uri-for-prefix($prefix,
$element)"/>
          <choose>
            <when test="empty($base-URI)">
              <message>Curie has bad prefix. Can't resolve prefix &quot;<value-
of select="$prefix"/>&quot;.</message>
            </when>
            <otherwise>
              <sequence select="xs:anyURI(concat($base-URI, $rest))"/>
            </otherwise>
          </choose>
        </otherwise>
      </choose>
  </function>

  <function name="cf:resolve-safe-curie" as="xs:anyURI?">
    <param name="safe-curie" as="xs:string"/>
    <param name="element" as="element()"/>
    <choose>
      <when test="matches($safe-curie, '^\[[^\[\]]+\]$')">
        <!-- it's a [curie] -->
        <sequence select="cf:resolve-curie(
         substring($safe-curie, 2, string-length($safe-curie) - 2), $element)"/>
      </when>
```

```
<!-- better to put improved URI tests here. xs:anyURI will take anything. -->
      <when test="string-length(normalize-space($safe-curie)) &gt; 0
                  and matches($safe-curie, '^[^\[\]]+$')">
        <sequence select="xs:anyURI($safe-curie)"/>
      </when>
      <otherwise>
        <message>Safe CURIE probably is not a URI. Value was &quot;<value-of
select="$safe-curie"/>&quot;</message>
      </otherwise>
    </choose>
  </function>

</stylesheet>
<!--
  Local Variables:
  mode: sgml
  indent-tabs-mode: nil
  fill-column: 9999
  End:
-->
```

Appendix E: Browser Display of Catalog (XSLT 1.0)

IEPD: Planning_Order

Attribute	Value
URI	http://myiepd.org/p-ord/
Class	iepd
Name	Planning_Order
Version	1.0.3rev2
Description	A Planning Order notifies subordinate agencies to initiate planning for contingency operations.

All File Artifacts (and Folders; organized by directory)

Local ID	Path and File Name	Description	External URI
f0	./	ROOT	
c0	./catalog.xml	Catalog: identifies all artifacts in Planning Order	
cs0	./catalog.xsl	stylesheet to display a catalog index with metadata	
cl0	./changelog.txt	History of changes to Planning Order	
f3	./documentation/	DOCUMENTATION	
d2	./documentation/master-document.docx	Instructions for implementing Planning Order	
f2	./library/	LIBRARY	
f21	./library/wsdl/	WEB SERVICES	
w13	./library/wsdl/p-order1.wsdl	Web Service description for IEPD Planning Order	[zz:order-service13]
f4	./XMLsamples/	SAMPLES	
f1	./XMLschemas/	XML SCHEMAS	
f13	./XMLschemas/exchange/	EXCHANGE	
r1	./XMLschemas/exchange/myiepd.xsd	Exchange schema for IEPD Planning Order	[zz:order-22]
f12	./XMLschemas/extension/	EXTENSION	
e1	./XMLschemas/extension/ext1.xsd	my extension schema 1	
e2	./XMLschemas/extension/ext2.xsd	my extension schema 2	
f11	./XMLschemas/subset/	SUBSET	
bi2	./XMLschemas/subset/niem/fbi/2.0/	FBI	
s2	./XMLschemas/subset/niem/fbi/2.0/fbi.xsd	FBI code lists	
ip2	./XMLschemas/subset/niem/fips_6-4/2.0/	FIPS 6-4	
s3	./XMLschemas/subset/niem/fips_6-4/2.0/fips_6-4.xsd	FIPS County codes	
nc2	./XMLschemas/subset/niem/niem-core/2.0/	NIEM CORE 2.0	
s1	./XMLschemas/subset/niem/niem-core/2.0/niem-core.xsd	NIEM Core 2.0 (niem-core.xsd)	
w1	./XMLschemas/subset/wantlist1.xml	Wantlist 1 for schema subset ss1	[yy:order-w1]
w2	./XMLschemas/subset/wantlist2.xml	Wantlist 2 for schema subset ss2; includes ss1	[yy:order-w2]

All FileSet Artifacts (organized by set)

Local ID	Path and File Name	Description	External URI
es1		Extension schema set for IEPD Planning Order	[yy:order-ext0]
e1	./XMLschemas/extension/ext1.xsd	my extension schema 1	
e2	./XMLschemas/extension/ext2.xsd	my extension schema 2	
ss1		Subset for IEPD Planning Order	
w1	./XMLschemas/subset/wantlist1.xml	Wantlist 1 for schema subset ss1	[yy:order-w1]
s1	./XMLschemas/subset/niem/niem-core/2.0/niem-core.xsd	NIEM Core 2.0 (niem-core.xsd)	
s2	./XMLschemas/subset/niem/fbi/2.0/fbi.xsd	FBI code lists	
ss2		A second subset for Planning Order; includes ss1	
s1	./XMLschemas/subset/niem/niem-core/2.0/niem-core.xsd	NIEM Core 2.0 (niem-core.xsd)	
s2	./XMLschemas/subset/niem/fbi/2.0/fbi.xsd	FBI code lists	
w2	./XMLschemas/subset/wantlist2.xml	Wantlist 2 for schema subset ss2; includes ss1	[yy:order-w2]
s3	./XMLschemas/subset/niem/fips_6-4/2.0/fips_6-4.xsd	FIPS County codes	

Notes about artifacts:

Artifacts with External URIs are reused from another MPD.

Metadata

Attribute	Value
Security Marking	Unclassified/public
Creation Date	2009-02-16
Last Revision Date	
Next Revision Date	
Status	implemented and operational in 25 of 50 states since 2009
Keyword	program plan
Keyword	contingency
Keyword	planning
Domain	Logistics
Domain	Emergency Management
Purpose	intended to initiate execution of planning cycles
Exchange Pattern	query/response
Exchange Partner	FEMA
Exchange Partner	NASCIO
Exchange Partner	82d Abn Div

Lineage

Relationship	Resource	Description
adapts	[zz:pln-ord6]	reuses 4 artifacts
adapts	[zz:pln-ord8]	uses modified extension set

Authoritative Source

Attribute	Value
Name	FEMA
Address	30 Dupont Cir, Wash DC, 12345, USA
Web Site	http://www.fema.gov
POC	John Smith, john.smith@fema.gov, 202-333-1234
POC	FEMA, Office of the Chief Administrator, OCA@fema.gov, 888-333-5678

Appendix F: MPD Artifacts

This appendix (the table for which begins on the next page) identifies mandatory and optional artifacts for various types of MPDs. The Cardinality columns list the minimum and maximum cardinality for each artifact in an MPD. Typical cardinalities are interpreted as follows:

Min, Max	The associated artifact is …
1, 1	required; not optional; one and only one is allowed
1, U	required; not optional; one or any number (unbounded) is allowed; repeatable
0, 1	optional; zero or exactly one is allowed (0 or 1)
0, U	optional; zero or any number (unbounded) is allowed; repeatable
NA	not applicable to this MPD (effectively 0, 0)

Key to File Extensions (types)	
csv	Comma Separated Values
doc	Microsoft Word (text document); includes docx
gif	Graphic Interchange Format
htm	HyperText Markup Language; includes html
jpg	Joint Photographic Experts Group; includes jpeg
mdb	Microsoft Access (database); includes mdbx
owl	Web Ontology Language
pdf	Portable Document Format
png	Portable Network Graphics
ppt	Microsoft PowerPoint (graphic, presentation); includes pptx
sch	Schematron
svg	Scalable Vector Graphics
txt	ASCII text
vsd	Microsoft Visio (graphic); includes vdx and other derivatives
wsdl	Web Services Description Language
xmi	XML Metadata Interchange
xml	Extensible Markup Language
xsd	XML Schema Definition
xls	Microsoft Excel (spreadsheet); includes xlsx
xsl	XML Stylesheet Language, includes xslt transformation
Sets of File Types	
TEXT	{txt, htm, pdf, doc}
GRAPH	{jpg, gif, png, pdf, svg, ppt, vsd}
SPREAD	{csv, xsl}

				Cardinality (Min, Max)				
Artifact	**Definition / Description**	**Typical File Extensions**	**Recommended Purpose**	**IEPD**	**EIEM**	**NIEM Release**	**Core Update**	**Domain Update**
The following MPD artifacts are either mandatory, not applicable, or if optional, they are common in some MPDs.								
Subset Schema (or Subset Schema Set)	A NIEM schema (or set) derived from a NIEM reference schema (or set) that is a subset of the reference schema in terms of its data components and cardinality of those components.	xsd	subset-schema (subset-schema-set)	0♦,U	0♦,U	NA	NA	NA
Reference Schema (or reference schema set)	A schema (or set) that is the authoritative definition of business semantics for components in its namespace; intended to serve as basis for components in IEPD schemas; constitute NIEM releases and domain updates	xsd	reference-schema (reference-schema-set)	0♦,U	0♦,U	1,U	1,U	1,U
Constraint Schema (or constraint schema set)	A schema (or set) used to represent and validate (using XML Schema) tighter constraints than are possible with NIEM alone.	xsd	constraint-schema (constraint-schema-set)	0,U	0,U	NA	NA	NA
Extension Schema (or extension schema set)	A schema (or set) that contain new data components or that extend existing NIEM components per NIEM NDR.	xsd	extension-schema (extension-schema-set)	0,U	0,U	NA	NA	NA
Exchange Schema	A schema declaring one or more root elements for use in IEPs (instances of an IEPD).	xsd	exchange-schema (exchange-schema-set)	1,U	NA	NA	NA	NA
Wantlist	An XML artifact that is input or output to the NIEM Schema Subset Generation Tool (SSGT) to build, save, or generate a NIEM schema subset.	xml	wantlist	1,U	1,U	NA	NA	NA
Sample XML instance	An XML instance document that validates with corresponding defining schemas; a sample IEP.	xml	sample-instance	1,U	0,U	NA	NA	0,U

♦ A NIEM-conforming IEPD or EIEM is required to have at least one schema that is either a NIEM reference schema or a subset derived from a NIEM reference schema.

				Cardinality (Min, Max)				
Artifact	**Definition / Description**	**Typical File Extensions**	**Recommended Purpose**	**IEPD**	**EIEM**	**NIEM Release**	**Core Update**	**Domain Update**
Catalog	A manifest that contains identifiers, directory structure, characteristic properties, and other metadata for the MPD and its artifacts.	xml	catalog	1,1	1,1	1,1	1,1	1,1
Change log	A history of modifications (initial change log identifies creation date only).	xml, SPREAD, TEXT	changelog	1●,1	1●,1	1■,1	1■,1	1■,1
Master document	An artifact containing explanatory prose text; primary and initial source of human readable information (similar to a `readme` file); references other key documents	Various	master-document	1,1	1,1	NA	NA	NA
Conformance verification	An artifact output from a conformance tool or other competent authority validating conformance to NIEM.	Various	conformance-report	0,U	0,U	0,U	0,U	0,U
The following MPD artifacts are either optional or not applicable								
Stylesheet or transformation	An XSLT or XSL file that converts an XML file to another form. Often used to convert an XML file to HTML for display in a browser.	xsl	transformation OR display	0,U	0,U	0,U	0,U	0,U
Ontology	A conceptual model that represents facts, relationships, and rules among entities.	owl	ontology	0,U	0,U	0,U	0,U	0,U
Mapping (to NIEM)	An artifact that represents how a data content model relates to equivalent NIEM data components.	xmi, SPREAD	mapping	0,U	0,U	NA	NA	NA
Data content model (UML)	A representation of a business data requirements, semantics, structure, and relationships.	xmi	data-model	0,U	0,U	NA	NA	NA

● An IEPD or EIEM change log may be informal in nature and is not required to conform to the strict XML schema definition that is mandatory for releases, core updates, and domain updates. Therefore, csv or TEXT formats are also authorized.

■ Each NIEM release, core update, and domain update is required to conform to the change log XML schema in http://reference.niem.gov/niem/resource/mpd/changelog/.

				Cardinality (Min, Max)				
Artifact	**Definition / Description**	**Typical File Extensions**	**Recommended Purpose**	**IEPD**	**EIEM**	**NIEM Release**	**Core Update**	**Domain Update**
Web Service	A definition for network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.	wsdl	web-service	0,U	0,U	NA	NA	NA
General documentation	An explanatory artifact.	TEXT	documentation	0,U	0,U	0,U	0,U	0,U
Graphic	A visual artifact such as a diagram, chart, picture, etc.	GRAPH	documentation	0,U	0,U	NA	NA	NA
Data dictionary	A lexicon that provides metadata names and associated semantics.	TEXT	data-dictionary	0,U	0,U	NA	NA	NA
Business Rules	An artifact used to document constraints beyond the capability of NIEM and XML Schema; may be used to validate or verify that such constraints are satisfied.	sch, TEXT	business-rules	0,U	0,U	NA	NA	NA
MOA/MOU	A Memorandum of Agreement or Understanding among exchange partners or enterprises.	TEXT	memorandum	0,U	0,U	NA	NA	NA
Endorsement	An acknowledgement from a competent authority (Federal, State, tribal, local government, or commercial) that an MPD has been adopted to use for a given purpose, locale, domain, or environment.	TEXT	endorsement	0,U	0,U	NA	NA	NA
Test data; quality assurance report	Testing results, data, or reports that indicate product quality or extent of testing performed.	Various	test-report	0,U	0,U	0,U	0,U	0,U
Tool-specific artifact	An artifact in a non-standard format (not listed in this Appendix) that is processed as input or generated as output to/from a software tool.	Various	tool-specific-file	0,U	0,U	NA	NA	NA

Appendix G: MPD Lexicon (Nature and Purpose)

`mpd-lexicon-1.0.owl` (version 1.0)

URI: http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/

Note: Download the latest text version of this file using its `http` URI above. Files in appendices are intended for browsing only; do not copy/paste for testing or processing.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="./mpd-lexicon.xsl"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY lexicon
"http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/" >
    <!ENTITY nature
"http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#" >
    <!ENTITY purpose
"http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#" >
]>

<rdf:RDF xmlns="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/"
     xml:base="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:lexicon="http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:nature="&lexicon;nature#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:purpose="&lexicon;purpose#">
    <owl:Ontology rdf:about=""/>

<!--
///////////////////////////////////////////////////////////////////////////
// Object Properties
///////////////////////////////////////////////////////////////////////////
-->

<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#administrativ
e -->

    <owl:ObjectProperty rdf:about="purpose#administrative">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of documentation that relates to business
operations, decisions, organizing people and resources, agreements,
testimonials, etc. about a model package description.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#documentation"/>
    </owl:ObjectProperty>
```

```
<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#business-
rules -->

    <owl:ObjectProperty rdf:about="purpose#business-rules">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a statement or set of statements (formal
or informal) that define and/or constrain some aspect of business data or
information.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#catalog
-->

    <owl:ObjectProperty rdf:about="purpose#catalog">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of identifying, locating, and describing all
artifacts in a model package description.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#changelog -->

    <owl:ObjectProperty rdf:about="purpose#changelog">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of recording all changes (additions,
deletions, modifications) to a model package description relative to a
previous version.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#conformance-
report -->

    <owl:ObjectProperty rdf:about="purpose#conformance-report">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a report that evaluates NIEM
conformance.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#report"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#constraint-
schema -->

    <owl:ObjectProperty rdf:about="purpose#constraint-schema">
        <rdfs:comment rdf:datatype="&xsd;string"
```

```
            >Serves the purpose of an XML schema that defines additional
constraints on XML instances beyond those that NIEM Naming and Design Rules
are capable of enforcing.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#constraint-
schema-set -->

    <owl:ObjectProperty rdf:about="purpose#constraint-schema-set">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a collection of XML schemas which impose
additional constraints on the XML instance documents (IEPs) of an
IEPD.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema-set"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#data-
dictionary -->

    <owl:ObjectProperty rdf:about="purpose#data-dictionary">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a record of information about data (i.e.
metadata) such as names, meaning, relationships to other data, origin, usage,
and format.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#data-
model -->

    <owl:ObjectProperty rdf:about="purpose#data-model">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a formal representation (e.g., UML) of
business data requirements that indicates data semantics, structure, and
relationships.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#display
-->

    <owl:ObjectProperty rdf:about="purpose#display">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of formatting information for viewing on a
computer monitor.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#transformation"/>
    </owl:ObjectProperty>
```

```
<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#documentation
-->

    <owl:ObjectProperty rdf:about="purpose#documentation">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of describing or explaining any aspect of
design, usage, testing, processing, etc. of a model package description
and/or its contents.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#endorsement -
->

    <owl:ObjectProperty rdf:about="purpose#endorsement">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of documentation that supports, approves,
sanctions, or recommends a model package description.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#administrative"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#exchange-
schema -->

    <owl:ObjectProperty rdf:about="purpose#exchange-schema">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of an XML schema that is a root schema (and
declares a root element) for an IEPD.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#exchange-
schema-set -->

    <owl:ObjectProperty rdf:about="purpose#exchange-schema-set">
        <rdfs:comment rdf:datatype="&xsd;string"
          >Serves the purpose of a collection of XML schemas which declare
root elements for XML instance documents (IEPs) of an IEPD.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema-set"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#extension-
schema -->

    <owl:ObjectProperty rdf:about="purpose#extension-schema">
        <rdfs:comment rdf:datatype="&xsd;string"
```

```
            >Serves the purpose of an XML schema that extends a NIEM release
in accordance with the NIEM Naming and Design Rules.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#extension-
schema-set -->

    <owl:ObjectProperty rdf:about="purpose#extension-schema-set">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a collection of XML schemas which contain
new or extend existing NIEM data components and conform to the NIEM Naming
and Design Rules.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema-set"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#file -->

    <owl:ObjectProperty rdf:about="purpose#file">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a block of arbitrary information, or
resource for storing information, which is available to a computer program
and is usually based on some kind of durable storage.</rdfs:comment>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#file-set
-->

    <owl:ObjectProperty rdf:about="purpose#file-set">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a collection of files managed together for
a specific purpose or reason.</rdfs:comment>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#incremental-
schema -->

    <owl:ObjectProperty rdf:about="purpose#incremental-schema">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of an XML schema within a Domain Update that
represents incremental changes to an existing NIEM domain.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#mapping
-->

    <owl:ObjectProperty rdf:about="purpose#mapping">
        <rdfs:comment rdf:datatype="&xsd;string"
```

```
          >Serves the purpose of describing how one data model or set of data
components corresponds to another by identifying semantic equivalence or
similarity.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#master-
document -->

    <owl:ObjectProperty rdf:about="purpose#master-document">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a primary source of documentation; a
readme file; a first documentation source to consult and one that may
reference other supplemental documentation.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#memorandum --
>

    <owl:ObjectProperty rdf:about="purpose#memorandum">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of documentation that records events,
observations, agreements, or other business related aspects pertaining to a
model package description.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#administrative"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#metadata-
extended -->

    <owl:ObjectProperty rdf:about="purpose#metadata-extended">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of an XML artifact that provides additional
metadata beyond that contained in a model package description
catalog.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#non-
normative-reference -->

    <owl:ObjectProperty rdf:about="purpose#non-normative-reference">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of technical documentation that provides
informal guidance or recommends methods that relate to  technical aspects of
a model package description.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#technical-reference"/>
    </owl:ObjectProperty>
```

```
<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#normative-
reference -->

    <owl:ObjectProperty rdf:about="purpose#normative-reference">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of technical documentation that provides
formal specifications, instructions, or rules for techniques and methods that
relate to a model package description.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#technical-reference"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#ontology
-->

    <owl:ObjectProperty rdf:about="purpose#ontology">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a standardized representation of knowledge
as a set of concepts within a domain, and the relationships between those
concepts.  It can be used to reason about the entities within that domain,
and may be used to describe the domain.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#quality-
assurance-report -->

    <owl:ObjectProperty rdf:about="purpose#quality-assurance-report">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a report that evaluates or measures degree
of excellence against some prescribed criteria.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#report"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#reference-
schema -->

    <owl:ObjectProperty rdf:about="purpose#reference-schema">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of an XML schema that defines and declares
NIEM data components that are authoritative.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#reference-
schema-set -->

    <owl:ObjectProperty rdf:about="purpose#reference-schema-set">
        <rdfs:comment rdf:datatype="&xsd;string"
```

```
            >Serves the purpose of a collection of XML schemas that contain
data components that are authoritive for NIEM.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema-set"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#replacement-
schema -->

    <owl:ObjectProperty rdf:about="purpose#replacement-schema">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of an XML schema within a domain update that
represents a complete replacement of an existing NIEM domain
schema.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#report -
->

    <owl:ObjectProperty rdf:about="purpose#report">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves  the purpose of a document that records evaluation
results produced through review, testing, or analysis that was executed by
automatic processing, manual (human) means, or a combination of
both.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#documentation"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#sample-
instance -->

    <owl:ObjectProperty rdf:about="purpose#sample-instance">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of an XML instance document that exemplifies
a typical instance that validates against an XML schema or schema
set.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#schema -
->

    <owl:ObjectProperty rdf:about="purpose#schema">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of an XML schema in general.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>
```

```
<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#schema-
set -->

    <owl:ObjectProperty rdf:about="purpose#schema-set">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a collection of XML schemas that are
maintained together for a logical reason or purpose (usually because of inter
dependencies).</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file-set"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#subset-
schema -->

    <owl:ObjectProperty rdf:about="purpose#subset-schema">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of an XML schema that has been derived from a
NIEM reference schema and represents a subset of the reference
schema.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#subset-
schema-set -->

    <owl:ObjectProperty rdf:about="purpose#subset-schema-set">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a collection of XML schemas which have
been derived from and represent a subset of a particular NIEM reference
schema set (release).</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#schema-set"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#technical-
reference -->

    <owl:ObjectProperty rdf:about="purpose#technical-reference">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of technical documentation that defines,
describes, or explains technical aspects of a model package
description.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#documentation"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#test-
report -->

    <owl:ObjectProperty rdf:about="purpose#test-report">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of a document that records the results of
testing.</rdfs:comment>
```

```
        <rdfs:subPropertyOf rdf:resource="purpose#report"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#tool-
specific-file -->

    <owl:ObjectProperty rdf:about="purpose#tool-specific-file">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of an artifact that is an input to or output
from some software tool used to develop contents of or implement a model
package description.  Such artifacts may be in either a standard open format
or proprietary format.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#transformatio
n -->

    <owl:ObjectProperty rdf:about="purpose#transformation">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of translating an artifact into another
format or representation.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#wantlist
-->

    <owl:ObjectProperty rdf:about="purpose#wantlist">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of an XML instance document that represents a
NIEM schema subset and is an input to or output from the NIEM Schema Subset
Generation Tool.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/purpose#web-
service -->

    <owl:ObjectProperty rdf:about="purpose#web-service">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Serves the purpose of establishing a method for communication
between two electronic devices over a network.</rdfs:comment>
        <rdfs:subPropertyOf rdf:resource="purpose#file"/>
    </owl:ObjectProperty>
```

```
<!--
////////////////////////////////////////////////////////////////////////
// Classes
////////////////////////////////////////////////////////////////////////
-->

<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#binary --
>

    <owl:Class rdf:about="nature#binary">
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a binary file with unknown or unspecified encoding
or format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#catalog -
->

    <owl:Class rdf:about="nature#catalog">
        <rdfs:subClassOf rdf:resource="nature#xml"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a NIEM model package description (MPD) catalog file
format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#changelog
-->

    <owl:Class rdf:about="nature#changelog">
        <rdfs:subClassOf rdf:resource="nature#xml"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a NIEM model package description (MPD) changelog
format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#character
-->

    <owl:Class rdf:about="nature#character">
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a character encoded file.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#core-
update -->

    <owl:Class rdf:about="nature#core-update">
        <rdfs:subClassOf rdf:resource="nature#mpd"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a NIEM core update model package description
(MPD).</rdfs:comment>
    </owl:Class>
```

```
<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#csv -->

    <owl:Class rdf:about="nature#csv">
        <rdfs:subClassOf rdf:resource="nature#character"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a Comma Separated Value (CSV) file.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#doc -->

    <owl:Class rdf:about="nature#doc">
        <rdfs:subClassOf rdf:resource="nature#binary"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a Microsoft Office Word file (includes doc and
docx).</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#domain-
update -->

    <owl:Class rdf:about="nature#domain-update">
        <rdfs:subClassOf rdf:resource="nature#mpd"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a NIEM domain update model package description
(MPD).</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#eiem -->

    <owl:Class rdf:about="nature#eiem">
        <rdfs:subClassOf rdf:resource="nature#mpd"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a NIEM Enterprise Information Exchange Model (EIEM)
model package description (MPD).</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a NIEM release model package description
(MPD).</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#file-set
-->

    <owl:Class rdf:about="nature#file-set">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a set of files that are grouped for a specified
purpose.</rdfs:comment>
    </owl:Class>
```

```
<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#gif -->

    <owl:Class rdf:about="nature#gif">
        <rdfs:subClassOf rdf:resource="nature#image"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of an image file in Graphic Interchange Format
(GIF).</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#html -->

    <owl:Class rdf:about="nature#html">
        <rdfs:subClassOf rdf:resource="nature#character"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in the W3C HyperText Markup Language (HTML)
format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#iepd -->

    <owl:Class rdf:about="nature#iepd">
        <rdfs:subClassOf rdf:resource="nature#mpd"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a NIEM Information Exchange Package Documentation
(IEPD) model package description (MPD).</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#image -->

    <owl:Class rdf:about="nature#image">
        <rdfs:subClassOf rdf:resource="nature#binary"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of an image file with unknown or unspecified
format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#jpg -->

    <owl:Class rdf:about="nature#jpg">
        <rdfs:subClassOf rdf:resource="nature#image"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of an image file in Joint Photographic Experts Group
(JPEG or JPG) format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#mdb -->

    <owl:Class rdf:about="nature#mdb">
        <rdfs:subClassOf rdf:resource="nature#binary"/>
        <rdfs:comment rdf:datatype="&xsd;string"
```

```
            >A nature of a file in Microsoft Office Access database format
(includes mdb and mdbx).</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#mpd -->

    <owl:Class rdf:about="nature#mpd">
        <rdfs:subClassOf rdf:resource="nature#file-set"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a NIEM model package description
(MPD).</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#owl -->

    <owl:Class rdf:about="nature#owl">
        <rdfs:subClassOf rdf:resource="nature#xml"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in the W3C Web Ontology Language (OWL)
format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#pdf -->

    <owl:Class rdf:about="nature#pdf">
        <rdfs:subClassOf rdf:resource="nature#binary"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in Adobe Portable Document Format
(PDF).</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#png -->

    <owl:Class rdf:about="nature#png">
        <rdfs:subClassOf rdf:resource="nature#image"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of an image file in Portable Network Graphics (PNG)
format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#ppt -->

    <owl:Class rdf:about="nature#ppt">
        <rdfs:subClassOf rdf:resource="nature#binary"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a Microsoft PowerPoint presentation file (includes
ppt and pptx).</rdfs:comment>
    </owl:Class>
```

```
<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#rdf -->

    <owl:Class rdf:about="nature#rdf">
        <rdfs:subClassOf rdf:resource="nature#xml"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in the W3C Resource Description Framework
(RDF) format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#release -
->

    <owl:Class rdf:about="nature#release">
        <rdfs:subClassOf rdf:resource="nature#mpd"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a NIEM release model package description
(MPD).</rdfs:comment>
    </owl:Class>


<!--
http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#schematron -->

    <owl:Class rdf:about="nature#schematron">
        <rdfs:subClassOf rdf:resource="nature#xml"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in Schematron format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#svg -->

    <owl:Class rdf:about="nature#svg">
        <rdfs:subClassOf rdf:resource="nature#binary"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in Scalable Vector Graphic (SVG)
format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#text -->

    <owl:Class rdf:about="nature#text">
        <rdfs:subClassOf rdf:resource="nature#character"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a plain text file.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#vsd -->

    <owl:Class rdf:about="nature#vsd">
        <rdfs:subClassOf rdf:resource="nature#binary"/>
        <rdfs:comment rdf:datatype="&xsd;string"
```

```
            >A nature of a file in Microsoft Visio diagram format (includes
vsd and vdx).</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#wantlist
-->

    <owl:Class rdf:about="nature#wantlist">
        <rdfs:subClassOf rdf:resource="nature#xml"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a NIEM wantlist file.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#wsdl -->

    <owl:Class rdf:about="nature#wsdl">
        <rdfs:subClassOf rdf:resource="nature#xml"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in the W3C Web Services Description Language
(WSDL) format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#xhtml -->

    <owl:Class rdf:about="nature#xhtml">
        <rdfs:subClassOf rdf:resource="nature#xml"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in the W3C XML HyperText Markup Language
format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#xls -->

    <owl:Class rdf:about="nature#xls">
        <rdfs:subClassOf rdf:resource="nature#binary"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a Microsoft Excel spreadsheet file (includes xls and
xlsx).</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#xmi -->

    <owl:Class rdf:about="nature#xmi">
        <rdfs:subClassOf rdf:resource="nature#xml"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in the Object Management Group (OMG) XML
Metadata Interchange (XMI) format.</rdfs:comment>
    </owl:Class>
```

```
<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#xml -->

    <owl:Class rdf:about="nature#xml">
        <rdfs:subClassOf rdf:resource="nature#character"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in the W3C eXtensible Markup Language (XML)
format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#xsd -->

    <owl:Class rdf:about="nature#xsd">
        <rdfs:subClassOf rdf:resource="nature#xml"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in the W3C XML Schema format.</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#xslt -->

    <owl:Class rdf:about="nature#xslt">
        <rdfs:subClassOf rdf:resource="nature#xml"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a file in the W3C XML Stylesheet Language (XSL)
Tranformations (XSLT) format (includes both xsl and xslt).</rdfs:comment>
    </owl:Class>


<!-- http://reference.niem.gov/niem/resource/mpd/lexicon/1.0/nature#zip -->

    <owl:Class rdf:about="nature#zip">
        <rdfs:subClassOf rdf:resource="nature#binary"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >A nature of a compressed archive in PKZIP format.</rdfs:comment>
    </owl:Class>


<!-- http://www.w3.org/2002/07/owl#Thing -->

    <owl:Class rdf:about="&owl;Thing">
        <rdfs:comment rdf:datatype="&xsd;string"
            >The most general OWL class of all individuals; a superclass of
all OWL classes.</rdfs:comment>
    </owl:Class>
</rdf:RDF>


<!-- Generated by the OWL API (version 2.2.1.1138)
http://owlapi.sourceforge.net -->
```

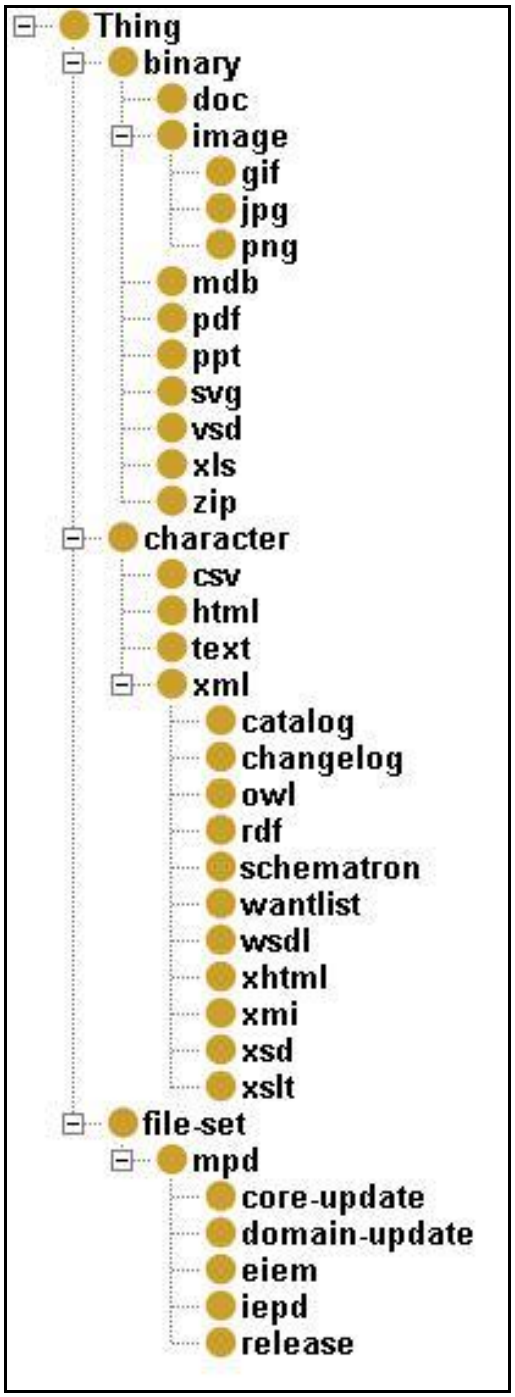
NIEM Natures

Nature	Definition
nature#binary	A nature of a binary file with unknown or unspecified encoding or format.
nature#catalog	A nature of a NIEM model package description (MPD) catalog file format.
nature#changelog	A nature of a NIEM model package description (MPD) changelog format.
nature#character	A nature of a character encoded file.
nature#core-update	A nature of a NIEM core update model package description (MPD).
nature#csv	A nature of a Comma Separated Value (CSV) file.
nature#doc	A nature of a Microsoft Office Word file (includes doc and docx).
nature#domain-update	A nature of a NIEM domain update model package description (MPD).
nature#eiem	A nature of a NIEM Enterprise Information Exchange Model (EIEM) model package description (MPD).
nature#file-set	A nature of a set of files that are grouped for a specified purpose.
nature#gif	A nature of an image file in Graphic Interchange Format (GIF).
nature#html	A nature of a file in the W3C HyperText Markup Language (HTML) format.
nature#iepd	A nature of a NIEM Information Exchange Package Documentation (IEPD) model package description (MPD).
nature#image	A nature of an image file with unknown or unspecified format.
nature#jpg	A nature of an image file in Joint Photographic Experts Group (JPEG or JPG) format.
nature#mdb	A nature of a file in Microsoft Office Access database format (includes mdb and mdbx).
nature#mpd	A nature of a NIEM model package description (MPD).
nature#owl	A nature of a file in the W3C Web Ontology Language (OWL) format.
nature#pdf	A nature of a file in Adobe Portable Document Format (PDF).
nature#png	A nature of an image file in Portable Network Graphics (PNG) format.
nature#ppt	A nature of a Microsoft PowerPoint presentation file (includes ppt and pptx).
nature#rdf	A nature of a file in the W3C Resource Description Framework (RDF) format.
nature#release	A nature of a NIEM release model package description (MPD).
nature#schematron	A nature of a file in Schematron format.
nature#svg	A nature of a file in Scalable Vector Graphic (SVG) format.
nature#text	A nature of a plain text file.
nature#vsd	A nature of a file in Microsoft Visio diagram format (includes vsd and vdx).
nature#wantlist	A nature of a NIEM wantlist file.
nature#wsdl	A nature of a file in the W3C Web Services Description Language (WSDL) format.
nature#xhtml	A nature of a file in the W3C XML HyperText Markup Language format.
nature#xls	A nature of a Microsoft Excel spreadsheet file (includes xls and xlsx).
nature#xmi	A nature of a file in the Object Management Group (OMG) XML Metadata Interchange (XMI) format.
nature#xml	A nature of a file in the W3C eXtensible Markup Language (XML) format.
nature#xsd	A nature of a file in the W3C XML Schema format.
nature#xslt	A nature of a file in the W3C XML Stylesheet Language (XSL) Tranformations (XSLT) format (includes both xsl and xslt).
nature#zip	A nature of a compressed archive in PKZIP format.
http://www.w3.org/2002/07/owl#Thing	The most general OWL class of all individuals; a superclass of all OWL classes.

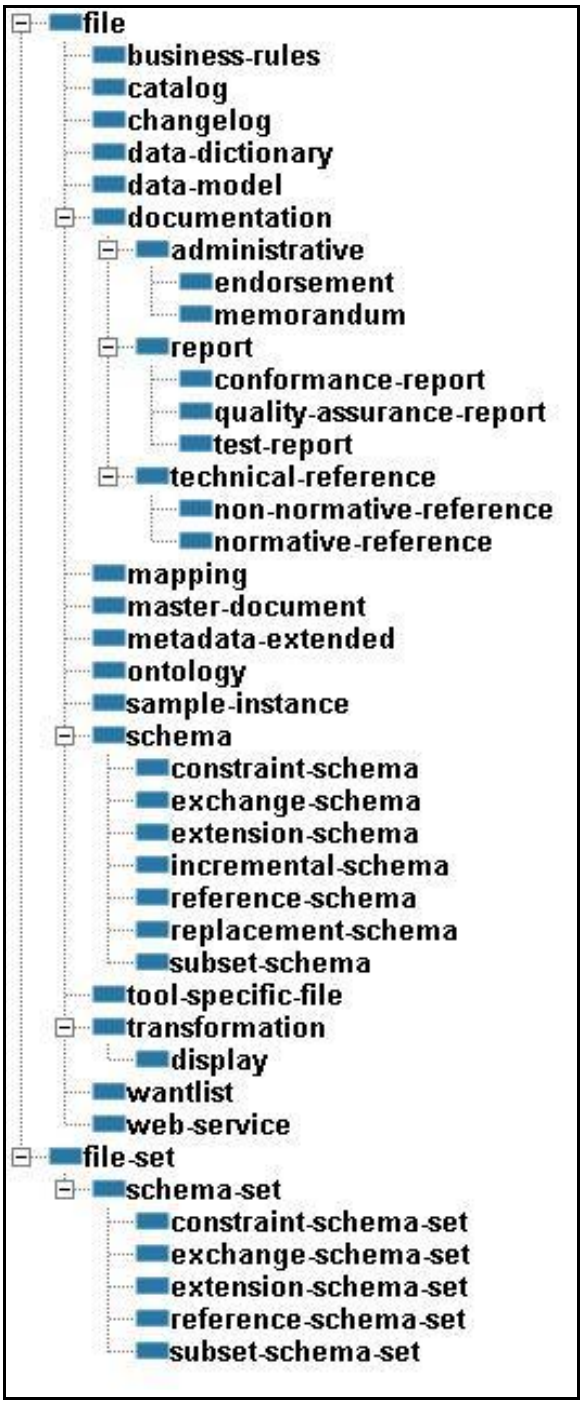
NIEM Purposes

Purpose	Definition
purpose#administrative	Serves the purpose of documentation that relates to business operations, decisions, organizing people and resources, agreements, testimonials, etc. about a model package description.
purpose#business-rules	Serves the purpose of a statement or set of statements (formal or informal) that define and/or constrain some aspect of business data or information.
purpose#catalog	Serves the purpose of identifying, locating, and describing all artifacts in a model package description.
purpose#changelog	Serves the purpose of recording all changes (additions, deletions, modifications) to a model package description relative to a previous version.
purpose#conformance-report	Serves the purpose of a report that evaluates NIEM conformance.
purpose#constraint-schema	Serves the purpose of an XML schema that defines additional constraints on XML instances beyond those that NIEM Naming and Design Rules are capable of enforcing.
purpose#constraint-schema-set	Serves the purpose of a collection of XML schemas which impose additional constraints on the XML instance documents (IEPs) of an IEPD.
purpose#data-dictionary	Serves the purpose of a record of information about data (i.e. metadata) such as names, meaning, relationships to other data, origin, usage, and format.
purpose#data-model	Serves the purpose of a formal representation (e.g., UML) of business data requirements that indicates data semantics, structure, and relationships.
purpose#display	Serves the purpose of formatting information for viewing on a computer monitor.
purpose#documentation	Serves the purpose of describing or explaining any aspect of design, usage, testing, processing, etc. of a model package description and/or its contents.
purpose#endorsement	Serves the purpose of documentation that supports, approves, sanctions, or recommends a model package description.
purpose#exchange-schema	Serves the purpose of an XML schema that is a root schema (and declares a root element) for an IEPD.
purpose#exchange-schema-set	Serves the purpose of a collection of XML schemas which declare root elements for XML instance documents (IEPs) of an IEPD.
purpose#extension-schema	Serves the purpose of an XML schema that extends a NIEM release in accordance with the NIEM Naming and Design Rules.
purpose#extension-schema-set	Serves the purpose of a collection of XML schemas which contain new or extend existing NIEM data components and conform to the NIEM Naming and Design Rules.
purpose#file	Serves the purpose of a block of arbitrary information, or resource for storing information, which is available to a computer program and is usually based on some kind of durable storage.
purpose#file-set	Serves the purpose of a collection of files managed together for a specific purpose or reason.
purpose#incremental-schema	Serves the purpose of an XML schema within a Domain Update that represents incremental changes to an existing NIEM domain.
purpose#mapping	Serves the purpose of describing how one data model or set of data components corresponds to another by identifying semantic equivalence or similarity.
purpose#master-document	Serves the purpose of a primary source of documentation; a readme file; a first documentation source to consult and one that may reference other supplemental documentation.
purpose#memorandum	Serves the purpose of documentation that records events, observations, agreements, or other business related aspects pertaining to a model package description.
purpose#metadata-extended	Serves the purpose of an XML artifact that provides additional metadata beyond that contained in a model package description catalog.

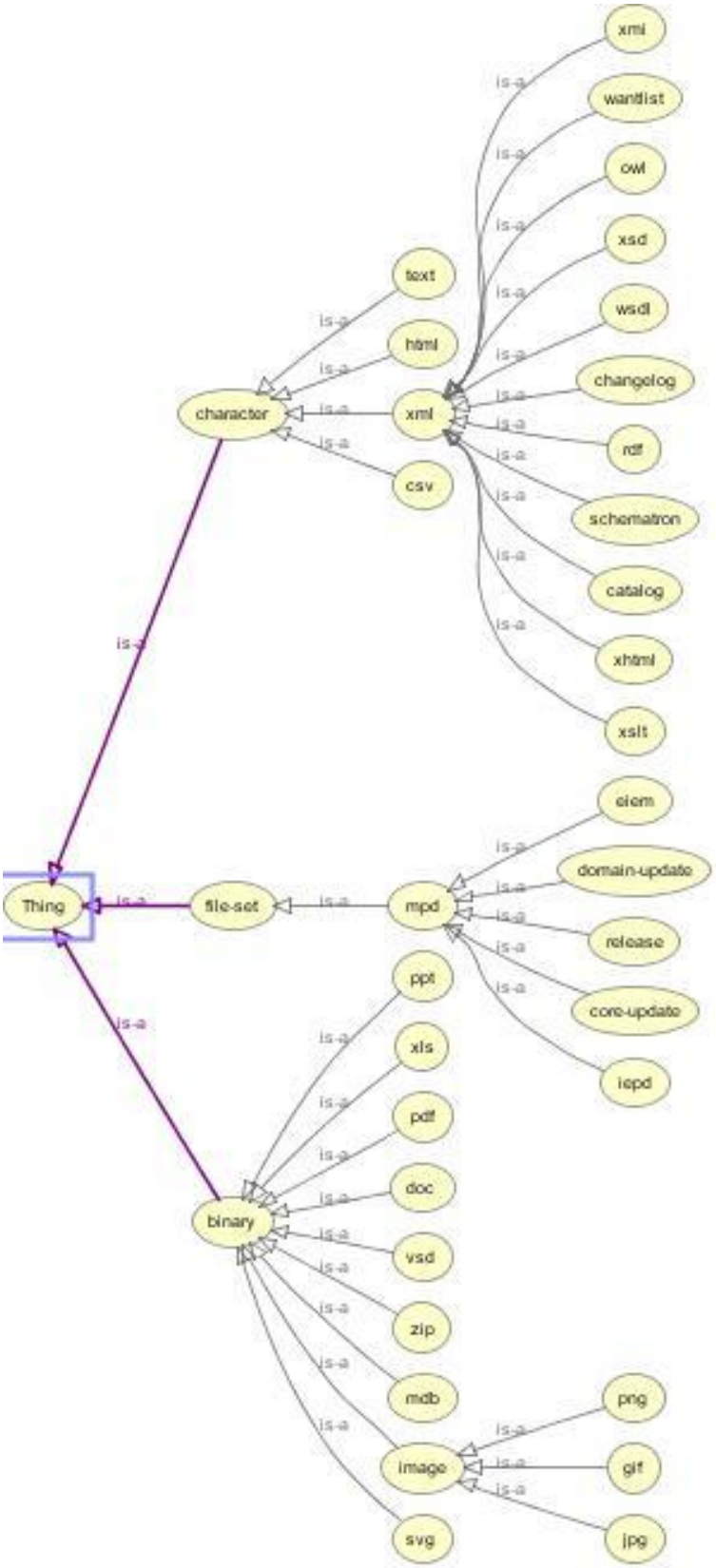
Purpose	Definition
purpose#non-normative-reference	Serves the purpose of technical documentation that provides informal guidance or recommends methods that relate to technical aspects of a model package description.
purpose#normative-reference	Serves the purpose of technical documentation that provides formal specifications, instructions, or rules for techniques and methods that relate to a model package description.
purpose#ontology	Serves the purpose of a standardized representation of knowledge as a set of concepts within a domain, and the relationships between those concepts. It can be used to reason about the entities within that domain, and may be used to describe the domain.
purpose#quality-assurance-report	Serves the purpose of a report that evaluates or measures degree of excellence against some prescribed criteria.
purpose#reference-schema	Serves the purpose of an XML schema that defines and declares NIEM data components that are authoritative.
purpose#reference-schema-set	Serves the purpose of a collection of XML schemas that contain data components that are authoritive for NIEM.
purpose#replacement-schema	Serves the purpose of an XML schema within a domain update that represents a complete replacement of an existing NIEM domain schema.
purpose#report	Serves the purpose of a document that records evaluation results produced through review, testing, or analysis that was executed by automatic processing, manual (human) means, or a combination of both.
purpose#sample-instance	Serves the purpose of an XML instance document that exemplifies a typical instance that validates against an XML schema or schema set.
purpose#schema	Serves the purpose of an XML schema in general.
purpose#schema-set	Serves the purpose of a collection of XML schemas that are maintained together for a logical reason or purpose (usually because of inter dependencies).
purpose#subset-schema	Serves the purpose of an XML schema that has been derived from a NIEM reference schema and represents a subset of the reference schema.
purpose#subset-schema-set	Serves the purpose of a collection of XML schemas which have been derived from and represent a subset of a particular NIEM reference schema set (release).
purpose#technical-reference	Serves the purpose of technical documentation that defines, describes, or explains technical aspects of a model package description.
purpose#test-report	Serves the purpose of a document that records the results of testing.
purpose#tool-specific-file	Serves the purpose of an artifact that is an input to or output from some software tool used to develop contents of or implement a model package description. Such artifacts may be in either a standard open format or proprietary format.
purpose#transformation	Serves the purpose of translating an artifact into another format or representation.
purpose#wantlist	Serves the purpose of an XML instance document that represents a NIEM schema subset and is an input to or output from the NIEM Schema Subset Generation Tool.
purpose#web-service	Serves the purpose of establishing a method for communication between two electronic devices over a network.



Nature



Purpose



Nature class diagram

Appendix H: Rule Summary

[Rule 3-1] Any XML instance that validates against a correct NIEM schema subset will always validate against the entire NIEM reference schema set from which that subset was created.

[Rule 3-2] NIEM subsets may omit elements with zero cardinality and adjust the cardinality of elements in reference schemas from which they are derived, as long as the subset property is maintained.

[Rule 3-3] NIEM subset schemas may omit all `xsd:annotation` elements that are in the NIEM reference schema from which it is derived.

[Rule 3-4] Each schema in a schema subset derived from a schema reference set bears the same (target) namespace as the schema in the reference set on which it is based.

[Rule 3-5] A schema contained in a reference schema set may be omitted from a derived subset, if and only if ALL of the following conditions are true within that schema:

- No elements/attributes declared or types defined in the schema are required for business exchange purposes. AND
- No elements/attributes declared or types defined in the schema are required to support other elements or types within the subset for exchange purposes; in other words, no references to elements or types in the schema exist in any other schema of the subset.

[Rule 3-5.1] If a schema in a reference schema set has been omitted from a derived subset, then every `xsd:import` occurrence of that schema MUST be removed from all schemas within the subset.

[Rule 3-6] An IEPD MUST contain at least one exchange schema artifact that declares at least one top-level root element for IEP instances specified by the IEPD.

[Rule 3-7] An IEPD exchange schema MUST NOT declare any XML element that is not intended for use as an IEP root element.

[Rule 3-8] A constraint schema bears a target namespace that has been previously assigned by a reference schema, extension schema, or exchange schema, or is a schema that is intended to support a constraint schema that has such a target namespace.

[Rule 3-9] A NIEM-conforming IEPD or EIEM MUST contain at least one schema that is either a NIEM reference schema or a subset derived from a NIEM reference schema.

[Rule 3-10] A NIEM IEPD MUST contain at least one valid sample XML instance (i.e., IEP) artifact for each exchange schema element that can be the root of a corresponding IEP.

[Rule 4-1] An MPD MUST contain an XML catalog artifact that validates with the NIEM MPD catalog schema (XSD) and that resides in the root directory of the MPD and bears the file name `"catalog.xml".`

[Rule 4-2] Every MPD MUST be assigned a version number.

[Rule 4-3] All NIEM version numbers adhere to the regular expression:

```
version ::= digit+ ('.' digit+)* (status digit+)?
```

Where:

```
digit ::= [0-9]
status ::= 'alpha' | 'beta' | 'rc' | 'rev'
```

`'alpha'` indicates early development

`'beta'` indicates late development; but changing or incomplete

`'rc'` indicates release candidate; complete but not approved as operational

`'rev'` indicates very minor revision that does not impact schema validation

(The regular expression notation used above is from XML 1.0 (Fifth Edition): http://www.w3.org/TR/2008/REC-xml-20081126/#sec-notation)

[Rule 4-3.1] A higher MPD version number within a version series does NOT imply compatibility between versions. Compatibility between or among MPD versions MUST be explicitly stated in documentation.

[Rule 4-4] Every MPD MUST be assigned a valid `http` URI.

[Rule 4-5] The URI for an MPD MUST end in its version number.

[Rule 4-6] Each file artifact in an MPD MUST have a corresponding `File` element in the catalog for that MPD.

[Rule 4-7] Each file set artifact in an MPD MUST have a corresponding `FileSet` element in the catalog for that MPD. This `FileSet` element must identify each file artifact that is a member of that file set artifact.

[Rule 4-8] Each artifact identified in the catalog MUST be assigned an `id` in the format of an `NCName` (Non-Colonized Name) as defined by **[W3-XML-Namespaces]**. This is required for both `File` and `FileSet` artifacts.

[Rule 4-9] A URI reference to an individual MPD artifact from another resource is the concatenation of

- The URI of the MPD that contains the artifact.
- The crosshatch or pound character (`"#"`).
- A fragment identifier that is the locally unique `id` of the artifact within the catalog of the MPD itself.

[Rule 4-10] NIEM namespaces MUST NOT be used as URIs for MPD artifacts.

[Rule 4-11] Every MPD that is a reference schema set (i.e., NIEM releases, core updates, and domain updates) MUST contain an XML change log artifact that:

- Validates with the NIEM change log schemas (`mpd-changelog.xsd` and `niem-model.xsd`).

Note: These are the base filenames; the actual filenames also contain a version number. For example: `mpd-changelog-1.0.xsd` is the current version.

- Records changes to previous reference schemas that this MPD represents.
- Bears the file name "`changelog.xml`".
- Resides in the root directory of the MPD.

[Rule 4-12] Every MPD that is an IEPD or EIEM MUST contain a change log artifact that:

- Records changes to previous IEPD or EIEM schemas that this MPD represents.
- Begins with the substring "`changelog`".
- Resides in the root directory of the MPD.

[Rule 4-13] The initial version of an IEPD or EIEM MUST contain a change log artifact with at least one entry for its creation date.

[Rule 4-13.1] If an IEPD or EIEM contains more than one change log artifact, then each change log artifact MUST:

- Have a file name that begins with the substring "`changelog`".
- Reside in the MPD root directory .

[Rule 4-14] An IEPD or an EIEM MUST contain a master document located in the MPD root directory whose filename begins with the substring "`master-document`".

[Rule 4-15] A NIEM IEPD or EIEM master document SHOULD (at a minimum) describe the MPD purpose, scope, business value, exchange information, senders/receivers, interactions, and references to other documentation.

[Rule 6-1] An MPD is packaged as a single compressed archive of files that represents a sub-tree of a file system in standard **[PK-ZIP]** format. This archive MUST preserve and store the logical directory structure intended by its author.

[Rule 6-2] Within an MPD archive, all XSD and XML artifacts MUST be valid against and follow all rules for their respective **[NIEM-NDR]** conformance targets (i.e., subset, constraint, extension, exchange, reference schemas, and XML instances); this includes being well-formed and valid XML Schema documents.

[Rule 6-3] An MPD archive MUST uncompress (unzip) to a one and only one MPD root directory.

[Rule 6-3a] An MPD archive file MUST use file name syntax defined by the regular expression:

```
mpd-filename ::= name '-' version '.' class '.zip'
```

Where:

```
name    ::= alphanum ((alphanum | special)* alphanum)?
alphanum ::= [a-zA-Z0-9]
special  ::= '.' | '-' | '_'
```

```
version  ::= digit+ ('.' digit+)* (status digit+)?
digit    ::= [0-9]
status   ::= 'alpha' | 'beta' | 'rc' | 'rev'
class    ::= 'rel' | 'cu' | 'du' | 'iepd' | 'eiem'
```

All alpha characters SHOULD be lower case to reduce the risk of complications across various file systems. See **[Rule 4-3]** for an explanation of the `status` options.

(The regular expression notation used above is from XML 1.0 (Fifth Edition): http://www.w3.org/TR/2008/REC-xml-20081126/#sec-notation)

[Rule 6-3b] Within an MPD, the `<name>` and `<version>` substrings in the file name MUST match exactly the values for attributes `mpdName` and `mpdVersionID` within its `catalog.xml` artifact.

[Rule 6-3c] Within an MPD, the `<class>` substring in the file name MUST correctly correspond to the value for the attribute `mpdClassCode` within `catalog.xml`. Correct correspondence is:

IF file name `<class>` =	THEN `catalog.xml mpdClassCode` =
`rel`	`release`
`cu`	`core-update`
`du`	`domain-update`
`iepd`	`iepd`
`eiem`	`eiem`

[Rule 6-3d] When represented on the Internet, an MPD archive SHOULD use the following MIME Type:

`application/zip+<class>` where

`<class>` is one member from the set `{rel, cu, du, iepd, eiem}`

Use of the generic zip MIME type `application/zip` is allowed, but discouraged. No other MIME types are allowed when representing MPD archives.

[Rule 6-4] Within an MPD archive, the value of each `xsd:import schemaLocation` attribute MUST be a relative path reference that resolves to the correct schema within the sub-tree.

[Rule 6-5] Absolute references to Internet resources MUST use a well-known transfer protocol (`http, https, ftp, ftps`) and MUST resolve (If applicable, documentation that describes how to resolve with security, account, and/or password issues MUST be included).

[Rule 6-6] A published IEPD MUST contain all documents necessary to understand it and allow it to be implemented correctly.

[Rule 6-7] A published IEPD MUST link (through its catalog) to any EIEM it is based on.

[Rule 6-8] Within an MPD archive, if non-NIEM-conforming schemas from other standards are used and referenced within an MPD, then all `xsd:import, xsd:include,` and

`xsd:redefine` constructs used within those schemas MUST be modified as needed to have a value for the `schemaLocation` attribute that is a relative path reference that resolves to the correct schema within the sub-tree.

[Rule 6-9] Within any artifact of an MPD archive, any direct reference to another resource (i.e., another artifact such as an image, schema, stylesheet, etc.) that is required to process or display an artifact SHOULD exist within the archive at the location specified by that reference.

Appendix I: Acronyms and Abbreviations

API – Application Programming Interface

BIEC – Business Information Exchange Component

CSV – Comma Separated Value (file format)

CURIE – Compact Uniform Resource Identifier

EIEM – Enterprise Information Exchange Model

GIF – Graphic Interchange Format

HLTA – High-Level Tool Architecture

HLVA – High-Level Version Architecture

HTML – Hyper Text Markup Language

IEP – Information Exchange Package

IEPD – Information Exchange Package Documentation

IEM – Information Exchange Model

JPG or JPEG – Joint Photographic (Experts) Group

MPD – Model Package Description

NA – Not Applicable

NDR – Naming and Design Rules

NIEM – National Information Exchange Model

NTAC – NIEM Technical Architecture Committee

OWL – Web Ontology Language

PDF – Portable Document Format

PMO – Program Management Office

RDF – Resource Description Framework

SSGT – Schema Subset Generation Tool

SVG – Scalable Vector Graphics

UML – Unified Modeling Language

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

URN – Uniform Resource Name

W3C – World Wide Web Consortium

WSDL – Web Services Description Language

XHTML – Extensible Hyper Text Markup Language

XMI – XML Metadata Interchange

XML – Extensible Markup Language

XSD – XML Schema Definition

XSL – Extensible Stylesheet Language

XSLT – Extensible Stylesheet Language Transformation

Appendix J: Glossary of Terms

The following terms are assumed to be defined in the context of NIEM.

artifact – A single file with a defined purpose or a set of files logically grouped for a defined purpose. An MPD is a collection of related artifacts.

Business Information Exchange Component (BIEC) – A NIEM-conforming XML schema data component definition or declaration (for a type, element, attribute, or other XML construct) reused, subsetted, extended, and/or created from NIEM that meets a particular recurring business requirement for an information sharing enterprise.

constraint schema – A schema which adds additional constraints to NIEM-conformant instances. A constraint schema or a constraint schema set validates additional constraints imposed on an XML instance only after it is known to be NIEM-conforming (i.e., has been validated by reference schemas, subset schemas, extension schemas, and/or exchange schemas). Constraint schema validation is a second-pass validation that occurs independently of and after conformance validation. A constraint schema is not required to be NIEM-conforming. A constraint schema need not validate constraints that are applied by other schemas. See also **constraint schema set**.

constraint schema set – A set of related constraint schemas that work together, such as a constraint schema set built by adding constraints to a schema subset. See also **constraint schema**.

core update – An MPD that it applies changes to a given NIEM core. It is never used to replace a NIEM core; instead, it is used to add new schemas, new data components, new code values, etc. to a particular NIEM core. In some cases, a core update can make minor modifications to existing core data components.

data component – An XML Schema type or attribute group definition; or an XML Schema element or attribute declaration.

domain update – A MPD that contains a schema or set of schemas issued by one or more domains that constitutes new content or an update to content that was previously included in a NIEM release. A domain update may define and declare new versions of content from NIEM releases or other published content. The issuing body vets each update before publishing, but the update is not subject to review by other NIEM bodies. A domain update must be NIEM-conformant, but otherwise it has fewer constraints on quality than does a NIEM release. Domain update schemas contain proposed future changes to NIEM that have not been published in a numbered release and have not been vetted by NIEM governance bodies (except by the domain or domains involved). Domain updates are published to the NIEM Publication Area at http://publication.niem.gov/ and available for immediate use within information exchanges.

Enterprise Information Exchange Model (EIEM) – An MPD that contains NIEM-conforming schemas that define and declare data components to be consistently reused in the IEPDs of an enterprise. An EIEM is a collection of BIECs organized into a subset and one or more extension schemas. Constraint schemas and non-NIEM-conforming external standards schemas with type adapters are optional in an EIEM.

exchange schema – A NIEM-conformant schema which specifies a document in a particular exchange.

extension schema – A NIEM-conformant schema which adds domain or application specific content to the base NIEM model.

harmonization – Given a data model, *harmonization* is the process of reviewing its existing data definitions and declarations; reviewing how it structures and represents data; integrating new data components; and refactoring data components as necessary to remove (or reduce to the maximum extent) semantic duplication and/or semantic overlap among all data structures and definitions resulting in representational quality improvements.

IEP root element – The single top-level element in an IEP (instance). In the absence of any other XML wrapping of an IEP, a root element declared in an exchange schema is an IEP document element.

Information Exchange Model (IEM) – One or more NIEM-conforming XML schemas that together specify the structure, semantics, and relationships of XML objects. These objects are consistent XML representations of information. Currently, five IEM classes exist in NIEM: (1) release, (2) core update, (3) domain update, (4) Information Exchange Package Documentation (IEPD), and (5) Enterprise Information Exchange Model (EIEM).

Information Exchange Package (IEP) – An XML document that is a correct instantiation of a NIEM IEPD, and therefore, validates with the XML schemas of that IEPD.

Information Exchange Package Documentation (IEPD) – An MPD that contains NIEM-conforming schemas that define one or more recurring XML data exchanges.

Information Sharing Enterprise – A group of organizations with business interactions that agree to exchange information, often using multiple types of information exchanges. The member organizations have similar business definitions for objects used in an information exchange and can usually agree on their common BIEC names and definitions.

major release – A NIEM release in which the NIEM Core schema has changed since previous releases. The first integer of the version number indicates the major release series; for example, versions 1.0, 2.0, and 3.0 are different major releases.

micro release – A NIEM release in which neither the NIEM Core nor the domain schemas have changed from the previous minor release, but one or more new schemas have been added (without impact to the domain or Core schemas). A third digit greater than zero in the version number indicates a micro release (for example, v2.1.1 – this release does not exist as of this date).

minor release – A NIEM release in which the NIEM Core has not changed from previous releases in the series, but at least one or more domain schemas have changed. A second digit greater than zero in the version number indicates a minor release (for example, v2.1). Note also that major v2.0 and minor v2.1 are in the same series (i.e., 2) and contain the same NIEM Core schema.

Model Package Description (MPD) – An organized set of files that contains one and only one of the five classes of NIEM IEM, as well as supporting documentation and other artifacts. An

MPD is self-documenting and provides sufficient normative and non-normative information to allow technical personnel to understand how to use and/or implement the IEM it contains. An MPD is packaged as a ZIP **[PK-ZIP]** file.

MPD root directory – The top level file directory relative to all MPD artifacts and subdirectories.

nature – An indication of the type of an MPD artifact. This further indicates how it should be processed by software tools.

purpose – A property of an MPD artifact that indicates its usage or function. This determines what to do with this artifact and/or how it should be processed by software tools.

reference schema – An XML Schema document that meets all of the following criteria:

- It is explicitly designated as a reference schema. This may be declared by an MPD catalog or by a tool-specific mechanism outside the schema.
- It provides the broadest, most fundamental definitions of components in its namespace.
- It provides the authoritative definition of business semantics for components in its namespace.
- It is intended to serve as the basis for components in IEPD and EIEM schemas, including subset schemas, constraint schemas, extension schemas, and exchange schemas.
- It satisfies all rules specified in the **[NIEM-NDR]** for reference schemas.

See also **reference schema set**.

reference schema set – A set of related reference schemas, such as a NIEM release. See also **reference schema**.

release – A set of schemas published by the NIEM Program Management Office (PMO) at http://release.niem.gov/niem/ and assigned a unique version number. Each schema defines data components for use in NIEM information exchanges. Each release is independent of other releases, although a schema may occur in multiple releases. A release is of high quality, and has been vetted by NIEM governance bodies. A numbered release may be a major, minor, or micro release.

schema coherence – A schema set is coherent when it has the following properties: (1) the schema set does not refer to a schema outside the set (i.e., the set is closed), and (2) the set does not include two different versions of the same component in an incompatible way.

schema subset (or subset schema set) – A NIEM-conformant set of subset schemas, derived from a set of reference schemas, but which specifies instances that may consist of a portion of the reference schema set. See also **subset schema**.

subset schema – An XML Schema document that meets all of the following criteria:

- It is explicitly designated as a subset schema. This may be declared by an MPD catalog or by a tool-specific mechanism outside the schema.

• It has a target namespace previously defined by a reference schema. That is, it does not provide original definitions and declarations for schema components, but instead provides an alternate schema representation of components that are defined by a reference schema.

• It does not alter the business semantics of components in its namespace. The reference schema defines these business semantics.

• It is intended to express the limited vocabulary necessary for an IEPD or EIEM and to support XML Schema validation for an IEPD.

• It satisfies all rules specified in the Naming and Design Rules [**NIEM-NDR**] for subset schemas.

See also **schema subset**.

Appendix K: References

[FEA-DRM]: The Federal Enterprise Architecture Data Reference Model, Version 1.0, September 2004. Available from http://xml.gov/documents/completed/DRMv1.pdf
A more recent DRM Version 2.0, 17 November 2005 is available from
http://www.whitehouse.gov/omb/assets/egov_docs/DRM_2_0_Final.pdf

[GJXDM-IEPD]: GJXDM Information Exchange Package Documentation Guidelines, Version 1.1, Global XML Structure Task Force (GXSTF), 2 March 2005. Available from
http://it.ojp.gov/documents/global_jxdm_IEPD_guidelines_v1_1.pdf

[ISO-Schematron]: Schema Definition Languages (DSDL) – Part 3: Rule-based validation – Schematron, ISO/IEC 19757-3:2006(E), First edition, 1 June 2006. Available from
http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip

[NIEM-BIEC]: Business Information Exchange Components (BIEC), Version 1.0, NIEM Technical Architecture Committee (NTAC), March 2011. Available from
http://reference.niem.gov/niem/guidance/business-information-exchange-component/1.0/

[NIEM-Conformance]: NIEM Conformance, Version 1.0, NIEM Technical Architecture Committee (NTAC), 15 September 2008. Available from
http://reference.niem.gov/niem/specification/conformance/1.0/

[NIEM-ConOps]: NIEM Concept of Operations, Version 0.5, NIEM Program Management Office, 9 January 2007. Available from
http://reference.niem.gov/niem/guidance/concept-of-operations/

[NIEM-DomainUpdate]: Draft NIEM Domain Update Specification, Version 1.0, NIEM Technical Architecture Committee (NTAC), 5 November 2010. Available from
http://reference.niem.gov/niem/specification/domain-update/1.0/

[NIEM-HLTA]: NIEM High-Level Tool Architecture, Version 1.1, NIEM Technical Architecture Committee, 1 December 2008. Available from
http://reference.niem.gov/niem/specification/high-level-tool-architecture/1.1/

[NIEM-HLVA]: NIEM High Level Version Architecture (HLVA), Version 1.0, NIEM Technical Architecture Committee, 2008. Available from
http://reference.niem.gov/niem/specification/high-level-version-architecture/1.0/

[NIEM-IEPD]: Requirements for a National Information Exchange Model (NIEM) Information Exchange Package Documentation (IEPD) Specification, Version 2.1, June 2006. Available from
http://niem.gov/files/NIEM_IEPD_Requirements_v2_1.pdf

[NIEM-Implementation]: NIEM Implementation Guidelines, NIEM Program Management Office. Available from http://niem.gov/implementationguide.php

[NIEM-Intro]: Introduction to the National Information Exchange Model (NIEM), Version 0.3, NIEM Program Management Office, 12 February 2007. Available from http://reference.niem.gov/niem/guidance/introduction/

[NIEM-NDR]: NIEM Naming and Design Rules (NDR), Version 1.3, NIEM Technical Architecture Committee (NTAC), 31 October 2008. Available from http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/

[NIEM-UserGuide]: NIEM User Guide Volume 1, U.S. Department of Justice, Office of Justice Programs, (date unknown). Available from http://reference.niem.gov/niem/guidance/user-guide/vol1/

[RFC2119-KeyWords]: Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, IETF RFC 2119, March 1997. Available from http://www.ietf.org/rfc/rfc2119.txt

[RFC3986-URI]: Berners-Lee, T., et al., Uniform Resource Identifier (URI): Generic Syntax, Request for Comments 3986, Network Working Group, January 2005. Available from http://tools.ietf.org/html/rfc3986

[W3-AssocResourcesNS]: Associating Resources with Namespaces, TAG Finding, World Wide Web Consortium, 25 June 2008. Available from http://www.w3.org/2001/tag/doc/nsDocuments-2008-06-25/

[W3-CURIE-Syntax]: CURIE Syntax 1.0 – A syntax for expressing Compact URIs, Version 1.0, W3C Candidate Recommendation, 16 January 2009. Available from http://www.w3.org/TR/2010/NOTE-curie-20101216/

[W3-OWL]: OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004. Available from http://www.w3.org/TR/2004/REC-owl-ref-20040210/

[W3-RDF]: Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation 10 February 2004. Available from http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[W3-XML]: Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation 26 November 2008. Available from http://www.w3.org/TR/2008/REC-xml-20081126/

[W3-XML-InfoSet]: XML Information Set (Second Edition), W3C Recommendation 4 February 2004. Available from http://www.w3.org/TR/2004/REC-xml-infoset-20040204/

[W3-XML-Namespaces]: Namespaces in XML (Second Edition), World Wide Web Consortium 16 August 2006. Available from http://www.w3.org/TR/2006/REC-xml-names-20060816/.

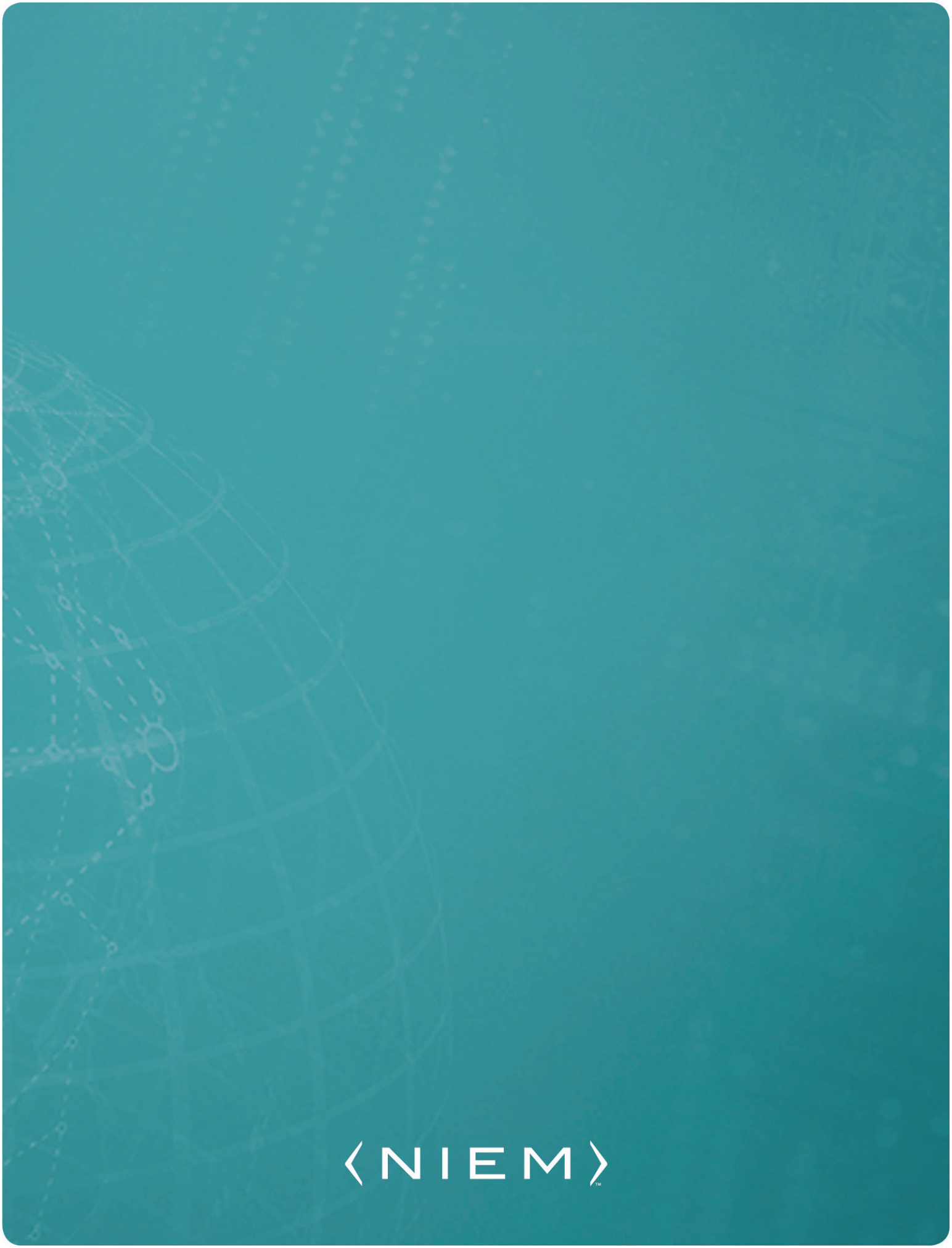
[W3-XML-SchemaDatatypes]: XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004. Available from http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

[W3-XML-SchemaStructures]: XML Schema Part 1: Structures Second Edition, W3C Recommendation 28 October 2004. Available from http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

[W3-XSLT]: XSL Transformations (XSLT), Version 1.0, W3C Recommendation 16 November 1999. Available from http://www.w3.org/TR/1999/REC-xslt-19991116

[W3-XSLT2]: XSL Transformations (XSLT), Version 2.0, W3C Recommendation 23 January 2007. Available from http://www.w3.org/TR/2007/REC-xslt20-20070123/

[PK-ZIP]: APPNOTE.TXT - .ZIP File Format Specification, Version: 6.3.2, Revised: 28 September 2007, Copyright (c) 1989 - 2007 PKWare Inc. Available from http://www.pkware.com/documents/casestudies/APPNOTE.TXT


NIEM