

MPD Technical Webinar Transcript

Mark Kindl:

On a previous Webinar, the NTAC Coordinator and one of the Co-Chairs of the NTAC introduced the NIEM MPD specification, which defines releases and IEPDs. In this webinar, we will take a closer look at this specification, its purpose, and its content. Let's go to slide two.

I am Mark Kindle, and this slide provides a very brief background sketch. I've been with the NIEM program since its beginning, and I sit on the NIEM Technical Architecture Committee, the NIEM Business Architecture Committee, and the XML structured taskforce. Go to slide three.

The NTAC is currently refining the first version, version 1.0, of the MPD specification and will release a version 1.1 in about a month or two. Note that all NIEM's technical specification guidelines and the resource files associated with them have URIs now, and they can be downloaded at those URIs with a browser. The URI is printed on the cover of each of these specifications. Next slide.

So what is an MPD? It is a set of XML schemas in NIEM with supporting documentation that explains how to use or reuse those schemas. The set of schemas may define one of five MPDs: a release, a core update, a domain update, an IEPD, or an EIEM (an Enterprise Information Exchange Model). You may already know some of these, but not all of them. We'll walk through each in a few moments. Next slide.

So what does this mean? Why does NIEM need the MPD spec? NIEM has several, actually five, different kinds of schema sets that have both similarities and differences. The MPD spec defines some consistency in packaging these sets. This helps to enable automatic processing of these packages. The MPD specification actually has its genesis in the GJXDM IEPD guidelines and in the NIEM IEPD requirements document, which were published some years ago in 2005 and 2006, respectively. NTAC wanted an MPD spec, but realized that there were many similarities to other schema sets in NIEM. But don't worry, there weren't any major changes to the IEPD guidelines, and we will cover that in a little bit. Next slide.

The primary purpose of the MPD spec is to provide consistency of content and structure through a set of principles, some normative rules and some guidelines, but the NTAC also tried to balance this with simplicity and a certain amount of flexibility. As already indicated—as I said earlier—the MPD spec covers five classes of schema sets. I have got them listed here. They are releases, which come in three different flavors, major, minor and micro-releases, core updates and domain updates. All three of these are covered in the NIEM high-level version architecture document. The last two IEPDs, which is probably what everyone is familiar with, and Enterprise Information Exchange Model are essentially user defined documents or user defined sets, and we'll cover these in a lot more depth in a few moments here. Next slide.

We should be on slide seven at this point. The NDR defines NIEM conformance of data components and schemas while the MPD spec defines NIEM conformance of the five fundamental packages. On the date this webinar was recorded in July 2012, an official effective date for the MPD spec has not been established. However, once announced on that date, authoritative sources for these NIEM products are expected to conform to the spec, version 1.1 for developers, and they are not required to revise MPDs that existed before that effective date, but they are encouraged to consider revising MPDs that existed before that date if they are being considered for major changes. Next slide please.

So now we'll go into the five MPD classes in some depth. Next slide should be slide nine.

The first class is essentially the NIEM numbered release. Most people are familiar with this. These are the reference schemas; these are the schemas that contain the components that are reused in IEPDs, essentially. On the left hand side, the three releases on the left hand side 1.0, 2.0, and 2.1 represent reality. We actually have those schemas available. However, on the right hand side, the last three 2.1.1, 2.1.2, and 3.0, that's fiction although it's a possibility. Notice that major releases change core. If you go across from major release to major release, you will see that the core changes and the domains change. For minor releases, only the domains change. So going from a major to a minor release only changes the versions of the domains, not the core; it remains the same. Micro releases are essentially releases that add on core or domain updates, which are updates that have passed stricter quality assurance checks to ensure they are coherent with the major or minor release, to which they are attached. Again, the NIEM high-level version architecture goes into some detail about this. Next slide.

A core update is essentially just that; it is an update to core. Now, notice the URL for the publication area, in the upper corner of the light blue box. This is where core and domain updates are posted. Core updates, and in a moment we'll look at domain updates, are both available for use in IEPDs whenever updates are published. Again, core updates and domain updates are defined by the NIEM high-level version architecture. The only limit on core updates is that they cannot replace parts of the core, they can only supplement, which means they are additive changes. After NIEM 3.0, codelists can be updated through core updates and will make it easier to maintain the model and to keep codelists updated easily. Next slide please.

We should be on slide 11 at this point. Again, in those publication areas, domain updates are also published here. In fact, if you add NIEM/domains to the back end of that URL, you will get to the domain area of the publication area. The red arrows indicate that domain updates can supplement or replace schemas in a release, but it's not part of a release unless an official release—micro release—has been published. In that case, refer to slide nine, and you will see what I mean.

The red arrows point to the schemas in release 2.1 that domain updates in the publication area supplement or replace, but we do not remove the old schemas from the NIEM 2.1 release; those are permanent. However, as far as IEPDs are concerned, you can use the domain updates and the core updates that are in the publication area with the appropriate major or minor release to which they are attached in a micro release. Next slide.

It's probable that NIEM users are most familiar and most comfortable with the IEPD concept. The IEPD is of course just a definition for a NIEM information exchange. And this slide illustrates a relatively simple and common IEPD structure. For IEPDs, not much is changed by the MPD specs; so you still see a subset, and you have an option of extending the IEPD with extension schemas. You have an exchange schema, which is essentially the root of the exchange; you have the option to use constraint schemas or constraint schema sets, which can further constrain the subset; and then you have a catalog, a change log, some documentation, and also sample instances that validate to the schemas. The most significant changes to the IEPD actually relate to the catalog, and also there is some acknowledgement in the MPD spec about some of the more complex options. We will talk about some of those in a little bit. Next slide.

This slide depicts what is referred to as the enterprise information exchange model, and this is the newest concept, which is actually still under development in NTAC, but it is incorporated at least on some level in the MPD spec. You can get an idea of what an Enterprise Information Exchange Model is if you get a hold of a copy of the Business Information Exchange Component Concept paper, that will provide a non-technical explanation of business information components and EIEMs, and you can find it at its URI which is <http://reference.niem.gov/niem/guidance>, and you won't have any problem tracking it down there. Enterprise Information Exchange Model is a local reference model from which IEPDs are built by the EIEM's authoritative source or author. As an authoritative source, you may want to define your own master subset and your own tailored extensions and then reuse these components throughout all the IEPDS you develop, which we refer to usually as a family. Note that one big difference in the Enterprise Information Exchange Model is that there is no exchange schema. That is because this is not an exchange, it is for a family of exchanges or a family of IEPDs. Next slide please.

So what has and what hasn't changed in the IEPD by virtue of the MPD spec? We will focus on that on this slide. I am just going to highlight about half of these, starting at the top. Number one, the basic IEPD framework, as I said earlier, remains the same: subsets, constraint schemas, extension schemas, and exchange schemas. Second bullet, there is a new concept we refer to as the base schema set. It will be in section 3.5 in the MPD spec, and it essentially defines all of the schemas with the exception of the constraint schemas. So a base schema set is all the common schemas—subset, extension, exchange together—but that does not incorporate a constraint schema set, which is considered a separate set from the base schema set. The fourth bullet, we've

unified what used to be called metadata file and the catalog into a single catalog artifact, and we've eliminated the duplication that existed before. And the seventh bullet, the URI is used to allow you to address individual IEPDs and MPDs as well as the individual artifacts within them. We essentially have a way to uniquely identify each artifact, each artifact set, and the MPD itself through URIs, and this helps with a lot of things including the catalog, the ability to refine other MPDs that are related to one you are building but is outside of yours. And there are many advantages to this. The ninth bullet, flexible directory organization. There are very few constraints on directory organization. Instead, the catalog becomes the normative specification for the structure of an MPD, and we'll take a look at that more in depth here in a little bit. And finally, we provide a little bit of guidance on some of the more complex IEPDs. For example, you are now—its explicitly mentioned in the MPD—that you are allowed to have multiple root elements within an exchange schema or multiple exchange schemas, for example. And there are other permutations that provide some flexibility in MPDs or IEPDs in particular. Go to the next slide.

This slide is simply a high-level summary of a more detailed appendix F in the MPD spec. Many of these artifacts are probably already familiar to most NIEM users. All of them are listed and defined in the appendix, although I would recommend that you refer to that appendix for the details. But this gives you some idea of what goes into an MPD or what can go into an MPD. You need to realize though that artifacts on the left may be mandatory or optional and it depends on what class of MPD you're building, so an exchange schema is not part of a release, but it is a mandatory part of an IEPD, for example. Next slide.

Now, what I want to do is look at the concept of representing the MPD within the catalog. I mentioned the URI scheme—and by the way, this slide and the slides that follow, the next several, are probably the significant changes or represent the most significant changes in the MPD spec, and most of the rules in the MPD spec actually focus on these, so that is why we will take a closer look. We will also follow up these slides with an example, so you will get some idea of what we are talking about here. This is a relatively fast summary so you'll certainly want to review the spec. You can see that the MPD has a URI itself, which is an http style URI. In addition, every artifact—and this includes artifact sets—every artifact and artifact set had a local unique identifier, an ID. Therefore, the URI for any given artifact in an MPD will be the concatenation of the MPD's URI and the pound sign and followed by the artifact ID. So, the idea here is that the authoritative source or for example, an IEPD, will assign the URI for that IEPD, and you will also assign the local unique IDs to the artifacts. And then, everything now has a URI when viewed from the outside. And so they can, every artifact within an IEPD can be referred to by another IEPD, for example. Within the catalog, you will essentially define a data structure for the IEPD in XML using the URIs, a couple of elements for files, and file sets that are all defined for the catalog schema—catalog.xsb. There is also a nature and purpose lexicon which provides additional information for each artifact. And all of this means that catalog becomes normative for the structure of the MPD—or in this case an IEPD—while the operating system directory structure is no longer necessary for processing the MPD. Next slide, slide 17.

We will continue here with concepts for representing MPDs. I have already said now that an MPD, a file, and a file set are all represented by URIs. The lexicon incorporates a nature property and a purpose relationship. The nature property is on every file, file set, and on the MPD itself. The purpose is essentially a descriptive relationship that can occur between an MPD and a file within that MPD, a MPD and a file set within that MPD, and between a file set and its file members. There is also a relative path name property and that property occurs on file and it also occurs on an optional folder which is a little beyond the scope of this, but you can read it in the MPD spec. But a folder is not actually a part of the formal MPD structure, it is simply a pointer to the location of an artifact in the operating system's file system. The baseline metadata included for search and discovery is also included within the catalog, and there is an optional way to extend metadata if the IEPD author determines he would like to extend it. Next slide.

And now we will turn to an example. The next two slides are a continuous example of a sample data structure that represents an IEPD or at least parts of it. And it's represented in a catalog XML file. This is essentially, obviously, a diagrammatic representation. This structure, by the way, is based on the resource description framework, but it uses XML not RDF. The red circle in the upper left corner represents the IEPD itself, and you notice it's got a couple of properties. It's got nature, which defines it as an IEPD, and it's got an ID, which is actually its URI. The blue and green circles represent artifact sets and single artifacts respectively. Each has an ID property, which is locally unique and when concatenated with the URI for the MPD, forms a URI for that artifact or artifact set. The purpose relationships are the yellow boxes, and you'll notice they are between the circles which are the artifacts and the MPDs itself and the sets. It defines the relationship between those artifacts. And then the nature, which is signified by "n," as a property of the artifacts is also there. The purpose and the nature are pre-defined artifacts; in other words, there are sets of labels to draw from for these that are in the MPD spec. And finally, there is an RP, which is a relative path name and all that is a pointer to the location to the artifact within the operating system's file system or in the directory structure. Bottom line is, this structure is completely independent of the operating system's file structure, as long as those relative path names point to the correct location of the files. It is tied to the file structure, but it is independent of it. It does not matter what kind of file structure you make for your IEPD, as long as the catalog points to correct locations for those files, the catalog will define normatively the structure for that IEPD. Next slide.

This is just a continuation of that same file structure. The idea now is that this structure is relatively easy to understand and process automatically. It is also relatively easy to write an XSLT, to translate it to a browser-displayable index for the files in the IEPD. And there is actually an example of an XSLT that does this and what its output looks like in the MPD spec, in the appendix. Notice that this part of the IEPD, which contains documentation also has a subordinate set member for a purpose, which is a little different from set member. This is for the XSLT generated indices or indexes, and what you find out is with HTML documentation, you often have many small, insignificant supplemental files that don't require direct indexing, and they will be referenced by other HTML files anyway. So this purpose, that is subordinate set member, helps identify insignificant files that

you may choose not to index in an XSLT generated index; you can just ignore them because they will be picked up by other HTML files anyway. Let's go to the next slide.

Now the primary reason the catalog was made to be independent of the operating system's file systems was because of feedback from the field that indicated so many different permutations that people wanted to use of the operating system directory structure, and there were just too many to make it consistent with any set of rules. However, there are some constraints and rules on what developers may do with the directory structure, and these are most of them. For example, the MPD is supposed to use the PK-Zip format, all the artifacts needed to process, to display or to validate a sample XML instance must be physically present in the MPD or in the IEPD archive.

They cannot be pointed to, they cannot be referenced out somewhere on the Internet. If it is required for display, for validation, or to process, it's got to be there. That does not mean that you cannot refer to things from this IEPD or this MPD on the Internet. For example, you may reference documents that might be interesting or associated with the IEPD as long as they aren't the master document. There is one and only one root directory for the MPD, so that is why this is rooted at the top. The catalog and the change log files must be files at the top of that directory, or in other words they must be immediately under the root. There is a standard syntax—and I won't go into it—but there is a standard syntax or MPD naming which includes the name, the version, and the class. So for example, at the very top on the right side, you see "my-iepd". "My" is actually the name and "iepd" is the class and then the version number is to the right. The syntax is defined in the MPD spec. It is a requirement that all XSD import values must be relative path references to the schema inside the MPD. This is so that when you unzip the file, everything is referenced correctly. And finally, that picture on the right side is non-normative. That is a suggested directory structure if you have nothing else to do or you have no other intent, but that is not normative. What's normative is the catalog. The catalog must be correct, and the MPD spec goes into a fair amount of detail to make sure that everyone understands how the catalog works. Next slide please.

What I've listed here is aspects of the MPD spec that are probably relevant for most NIEM users. Now, version 1.1 is not released yet, but you can get an early view of what will be in it—a lot of what will be in it—by reviewing the version 1.0 spec. And I've got the URL for the 1.1 spec on the last slide, you just have to change the version and you will find the 1.0. You won't find the 1.1, obviously, because they just said it is not released yet. Just be aware that the version 1.1 will be the first to be effective once the date has set. We're basically skipping 1.0. Some of the more relevant items I'll mention up here. Section 2 incorporates basic concepts and terminology. This is a pretty good section to start with. Section 1 tends to be a lot of background, but you make sure you understand Section 2, with the concepts and terminology because you'll need that further on into the specs. You want to make sure you understand the IEPD catalog and its importance to the tools that would process it and why it's important. You want to make sure you understand how to structure an IEPD and that you understand how to use the nature and purpose lexicons that's fundamental to the structure of an IEPD because that's how you define the catalog. And finally, make sure you read Section 3.5—The Base Schema Sets, that's new and you will not see it in version 1.0. Next slide please.

We've listed here all of the appendices in the back of the MPD spec. Recall earlier I spoke of resources associated with specifications and guidelines. This is essentially a summary of several of those resources. Every resource file has a URI as well as the specifications and the guideline documents, so you can go download those at these URIs, just by using them as URLs. These essentially are the files for developers, in other words, they don't contain hidden characters that the MS word or PDF files would contain, so these are clean and these will validate. Next slide.

So, in conclusion, I reiterate the NIEM MPD specification version 1.1 is not released yet, but will be in a month or two, and this is the URL where you can get it once it's been released. If you want, take a look at version 1.0, simply change that URL at the far right side to 1.0 and you'll be able to download that. And if you have questions or comments, please send them to the Help Desk. You've got the email address, as well as the website and phone numbers. Thank you.