The NIEM Metamodel and Common Model Format

This paper is a brief high-level explanation of the new NIEM metamodel and Common Model Format (CMF). It describes the relations between runtime data, development-time data models, and the data model for those data models (known as the *metamodel*), and lists the anticipated benefits of this new modeling approach.

Data is a symbolic representation of facts, concepts, or instructions, in a formalized manner suitable for communication, interpretation, or processing by humans or by automatic means. A *data model* specifies the facts to be represented (a subset of all the facts that could possibly be represented), and provides definitions for those facts in terms of the objects, properties, and relationships of interest. Consider the diagram below:

Person			
PersonAge	35	33	24
PersonFullName	Clark Kent	Lois Lane	Jimmy Olsen
data model	*data values*		

The left column illustrates a data model that provides definitions for the Person object type, the PersonAge and PersonFullName properties, and relationships among those (for example, a Person object HAS-A PersonAge attribute). These definitions are used to interpret the data values in the other columns. Many other potential facts (hair color, secret identity, etc.) are not part of this data model, because those facts are not of interest in this scenario.

A data model is itself a symbolic representation of facts. From another point of view, the data model is just data – in this case, data which says that the model has one object type (Person) and two properties (PersonAge, PersonFullName). That data itself has a data model, illustrated below:

Object Type	Person
Property	PersonAge
Property	PersonFullName
data model	*data values*

We call this second model a *metamodel*, because it is a data model for data models. Putting the two diagrams together gives us:

Object Type	Person	↔	Person	
Property	PersonAge	↔	PersonAge	35
Property	PersonFullName	↔	PersonFullName	Clark Kent
data model (describing models)	*data values (for a data model)*		*data model (describing people)*	*data values (for a person)*

From right to left: there is the data, which expresses facts that describe particular things of interest, facts like "Clark Kent" and "35". There is the data model, which specifies the kind of things that will be

described, things like "people" and "names". That data model is itself data. Finally, there is the metamodel, which specifies the kind of things described in a data model, things like "types" and "properties". Rotate this diagram 90 degrees, and we get a three-layer view of data modeling:

Data	"35", "Clark Kent"	Users
Data model	Person, PersonAge, PersonName	Developers
Metamodel	Object type, Property, Relationship	Tool Builders
Layer	*Layer Contents*	*Who Cares*

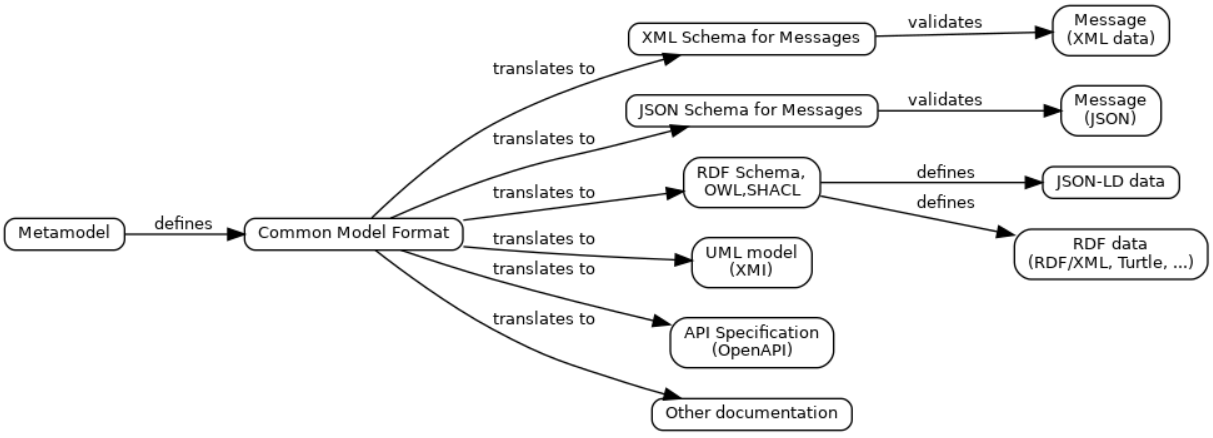
The different layers matter to different people. Users care about the data. The developers who build applications for those users care about the data model. People who create technical specifications and tools for those developers care about the metamodel.

The NIEM technical architecture has always included all three layers. Until now, the metamodel was *implicit,* written between the lines of the Naming and Design Rules. Until now, NIEM used XML Schema as the format for the NIEM data model (core and domains) and for the models in many different IEPDs. Until recently, all NIEM data was XML data.

The NTAC has now introduced an *explicit* metamodel and a *technology-neutral* data model format. This explicit metamodel is the result of applying the NIEM modeling approach. It is a NIEM-conforming message specification (or IEPD). This specification defines the *Common Model Format (CMF).* The CMF is technology-neutral: a CMF data model can be expressed as XML or JSON or any other data serialization that NIEM supports. The data described by the CMF data model can also be expressed in any supported serialization.

Data	message content; for instance, "35", "Clark Kent"	Users
Data model	NIEM model (core and domains), message models	Developers
Metamodel	Common Model Format (CMF) specification	Tool Builders
Layer	*NIEM*	*Who Cares*

The CMF is supported by free and open-source tools which convert CMF data models into technology-specific developer artifacts: XML Schema documents, JSON Schema documents, etc. These tools are under development now.



With the introduction of the Common Model Format, NIEM now supports many data serializations: JSON, RDF, others as needed, in addition to XML. A number of advantages result from this:

- Application developers can implement a NIEM-based data exchange using their preferred technology stack. Many different API frameworks can be supported (OpenAPI, Apache Avro, etc.) The CMF tool suite can produce the interface artifacts they require.
- Message designers can create a single message specification that defines equivalent messages in any supported serialization. At runtime, a message in one serialization can be translated to another, without loss and without specialized translation software.
- NIEM communities become more effective when the shared data definitions they create become useful in many more situations.
- Users receive better information through more and better machine-to-machine data interoperability.

The overall result is reduced time and cost for data interoperability across any enterprise that applies NIEM. This effect will be most pronounced in a large enterprise with many interacting applications.