A Comparison of Cursor-on-Target, UCore, and NIEM

Introduction

This paper is a comparison of Cursor-on-Target (CoT), UCore, C2 Core, and the National Information Exchange Model (NIEM). All of these specifications have application in XML-based, machine-to-machine exchange of structured data. There are many important differences, which we analyze along the following dimensions:

- Intended subject-area domain
- Information exchange specification (IES) vs. IES framework
- Utility for extensible, loose-coupler IES
- Codelist and taxonomy representation
- Governance and configuration management (CM) for community extensions
- Compatibility with the Geographic Markup Language (GML)
- Compatibility with Intelligence Community (IC) security specifications
- Utility in low-bandwidth communications environment
- Implementation complexity and technology requirements
- Developer and runtime tools

We begin with a description of the comparison dimensions, and then proceed to the analysis of the individual specifications. The final section describes the portion of the tradespace where CoT particularly excels.

Dimensions of Analysis

Intended subject-area domain: Some specifications are intended for and limited to a particular subject area. Others claim universal applicability

IES vs. IES framework: An IES defines a particular data exchange; it explains what developers must know to write code which produces or consumes an instance of that exchange. An IES framework provides rules and predefined data components for creating any number of particular IES.

Utility for extensible, loose-coupler IES: Every IES creates coupling between the producing and consuming applications. With a loose-coupler IES, many applications exchange a little data (measured by subject area, not bits), based on the simple, useful, rough intersection of their data sharing needs. An extensible IES allows a particular community to accept more coupling to achieve more information sharing, through messages that can be processed by systems that use the base IES but which are programmed without awareness of the extension.[1]

Codelist and taxonomy representation: All of the compared specifications permit their messages to include values drawn from codelists and taxonomies. They differ in the way these values are

[1] *Loose Couplers and Unanticipated Use* (2010)

represented in the message, in the way the codelists and taxonomies are defined, and in the way that values are to be interpreted by consuming applications.

Governance and CM for community extensions: All of the compared specifications support some form of "extension". Some provide institutional support for communities to agree on their own subject-area vocabularies and to reuse terms defined by other COIs.

Compatibility with GML: There are three forms of compatibility. The highest level is a data exchange in which each message is a conforming GML document, defined by a GML application schema. Another form is a data exchange with embedded GML content, in which the message as a whole is not a GML document, but does contain valid GML elements. The lowest level is a data exchange with no GML content but containing geospatial information which can be transformed into GML elements.

Compatibility with IC security specifications: The IC has defined and DoD is adopting a number of security-related specifications relevant to the data exchange specifications compared in this paper. These IC specifications are collected into a group known as the "Smart Data Stack", which includes ISM (Information Security Markings), NTK (Need To Know), and TDF (Trusted Data Format).

Utility in low-bandwidth environment: Character-based XML documents are notoriously verbose. This poses difficulties when bandwidth is constrained. The short messages common in tactical networks are especially problematic. These problems may be resolved by Efficient XML Interchange (EXI), the W3C standard for binary/compressed XML, but only if the message XML schemas are properly designed.

Implementation complexity and technology requirements: This dimension includes the sophistication of the tools in the development environment, and the technical competence assumed on the part of the people who operate these tools. "XML technology" refers to many specifications all layered upon the base XML standard. CoT and the others employ different subsets of these specifications.

Developer and runtime tools: Developer tools assist with creating software to produce and/or consume messages. Runtime tools help with processing messages.

Cursor-on-Target

CoT version 2.0 is maintained by the CoT program office at Hanscom AFB. The base layer of the CoT specification has been stable since 2004.

Intended subject-area domain	situation awareness update messages
IES vs. IES framework	single IES
Extensible loose-coupler IES	yes
Codelists and taxonomies	hierarchical string format
Governance and CM	CoT program office
Compatibility with GML	transformable content

Compatibility with IC security	high, for ordinary use
Low-bandwidth utility	high, but not optimal
Implementation complexity	very low
Developer and runtime tools	yes

CoT provides a single IES which is primarily focused on situation awareness update. CoT messages provide the "where and when" for a single "what" entity, something that could plausibly be displayed on a map. The CoT IES is the first well-known example of an extensible loose-coupler. The base layer of data may be extended with additional facts needed by a subset of the CoT community (for example, speed and direction), and can be usefully processed by consumers unaware of the extension. Schema extensions are segregated into a single `<details>` element; taxonomy extensions are described below.

CoT messages represent values of the "what is it?" taxonomy as a hierarchically-formatted string that is based in part on MIL-STD 2525. Common string prefixes indicate nodes with a common parent in the taxonomy; for example, the nodes "a-f-A" and "a-f-G" are children of the node "a-f". Taxonomy extensions are created by appending new characters to an existing string. As a result, consuming applications may always process the base portion of a taxonomy value, without any knowledge of extensions.

Any community may create schema or taxonomy extensions for CoT. The CoT program office recognizes important consensus extensions as "stable". Stable extension terms become optional parts of the specification.

CoT messages contain geospatial information which can be converted into GML elements without loss of accuracy or precision through a simple XSLT script.

CoT predates all of the IC security specifications, and CoT messages do not usually include any of the elements or attributed defined in those specifications. However, CoT messages can be easily wrapped as the structured payload in a Trusted Data Object (TDO), which defines its own elements for all of the required markings. This can usually be done without changing the CoT message. In the unusual case of a CoT message with differently-classified portions, it is possible to add ISM markings to a CoT message. This is backward-compatible with the vast majority of CoT consumers. In theory, there can be CoT message with portions that cannot be properly ISM-marked; however, in practice, there are no known situations where such a thing is required.

CoT is designed for situations where a small message size is important, but where the absolute minimum message size is not required. CoT achieves its small message size by choosing attributes instead of elements for most data values, and by choosing very small attribute and element names. Some of the attribute values in a CoT message do not compress well with EXI, and so CoT is not the best choice if absolute minimum message size is essential.

CoT is designed with few assumptions about the development and runtime environment, and requires very little technical understanding of XML technologies on the part of developers. It requires no knowledge of XML specifications beyond XML syntax. Solutions do not depend on namespaces, XML schema, or Schematron.

The CoT distribution includes a debugging tool to assist in creating software to produce and/or consume CoT messages. It also includes a message router which can be configured to perform publish/subscribe dissemination of CoT messages from many producers to many consumers.

UCore 1.0

Three distinct and very different specifications share the name "UCore". The first of these was developed by DoD and IC working together, and was released in October 2007.

Intended subject-area domain	situation awareness update messages
IES vs. IES framework	small number of IES
Extensible loose-coupler IES	yes
Codelists and taxonomies	unspecified
Governance and CM	none
Compatibility with GML	GML application schema
Compatibility with IC security	outdated, but easily fixed
Low-bandwidth utility	low
Implementation complexity	above average
Developer and runtime tools	no

UCore 1.0 was inspired by CoT, and has its origin in a CoT demonstration to the SECAF and DoD CIO in 2006. It defines a small number of IESs, which like CoT provides the "where and when" of a single "what", but which are technically modernized and "born joint". These IESs may be extended by communities and their messages partially processed by software written without knowledge of the extension. There is no governance or CM mechanisms for these extensions.

UCore 1.0 provides the uc:type element to contain a value from a "what is it?" taxonomy. The particular taxonomy is specified via the codeSpace attribute. UCore 1.0 does not itself provide a "what is it?" taxonomy. It does not prescribe a representation for taxonomy values in a message, or a format for defining codelists and taxonomies.

UCore 1.0 was intentionally developed as a GML application schema. It applies an outdated version of the ISM standard, but this could be changed without difficulty. UCore 1.0 messages can be wrapped as the structured payload within a TDO.

UCore 1.0 messages are not well suited for the low-bandwidth environment. None of the XML component names were chosen for brevity. Some of the GML components in UCore 1.0 do not compress well with EXI, especially the codeSpace and srsName attributes. This is a serious problem in short messages. For example, the EXI bitstream for the GML element

```
<gml:Point srsName="http://metadata.ces.mil/mdr/ns/GSIP/crs/WGS84E_2D">
    <gml:pos>-35.112 70.011</gml:pos>
</gml:Point>
```

will require nearly 400 bits for the srsName attribute value. All of those bits are wasted in a CoT-like IES that always uses the WGS84 coordinate reference system. In a message with an information content on the order of 500 bits, this overhead is usually unacceptable.[2]

UCore 1.0 message designers need to understand how to build conforming GML application schemas. To create extended schemas, they must understand the xsi:type construct in XML Schema. UCore 1.0 does not provide any developer or runtime tools of its own.

UCore 2.0

This version was developed by four Departments: DoD, DoJ, DHS, and DNI. UCore 2.0 was released in March 2009.

Intended subject-area domain	cross-Department data exchange
IES vs. IES framework	single IES
Extensible loose-coupler IES	no
Codelists and taxonomies	OWL format
Governance and CM	none
Compatibility with GML	embedded GML
Compatibility with IC security	outdated, but easily fixed
Low-bandwidth utility	low
Implementation complexity	high
Developer and runtime tools	no

UCore 2.0 defines a single IES which provides a messaging framework in which the message payload is packaged together with message metadata (sender, timestamp, etc.) and a "who, what, where, when" digest of the payload contents. Unlike CoT and UCore 1.0, this digest reports the "where and when" of multiple entities. UCore 2.0 is intended to be useful for the "unanticipated consumer" and for cross-Department data exchange, by providing a small message which anyone can understand; consumers needing more details can read documentation to understand the payload.

Although UCore 2.0 can be extended, there are no examples of UCore 2.0 used as an extensible loose-coupler IES. The message framework and digest do not form a core IES which can be extended by communities but which can still be usefully processed by applications written without knowledge of the extension.[3] Some pilot implementations use the digest as a discovery metadata summary of the payload, resembling a DDMS metacard. Others use the digest as the starting point for a particular IES, one which cannot be usefully interpreted by software written to the UCore specification alone.

UCore 2.0 provides the What element and the codeSpace attribute for "what is it?" taxonomy values. UCore 2.0 defines a simple taxonomy and encourages communities to define their own taxonomies which either extend the UCore taxonomy or relate to it. UCore and community

[2] A more detailed discussion of this concern is found in *Subset Schemas and UCore Reuse Rules* (2011)
[3] *Command and Control On UCore: Message Design Experiment* (2010)

taxonomies are defined in OWL syntax. Consuming applications must have these OWL definitions during development or runtime; applications without knowledge of a taxonomy extension cannot understand and correctly process values from that taxonomy in a message.

A UCore 2.0 message is not a GML document, but does contain embedded GML components. UCore 2.0 applies an outdated version of the ISM standard, but this could be changed without difficulty. UCore 2.0 messages can be wrapped as the structured payload within a TDO.

UCore 2.0 component names were not chosen for brevity. The embedded GML components and the components defined by UCore 2.0 make extensive use of attribute values which do not compress well. UCore 2.0 messages are thus not well suited for the low-bandwidth environment, especially when messages are short.

Implementations capable of coping with the complete UCore 2.0 specification are necessarily complex. The digest is a graph structure with potentially ambiguous interpretations.[4] The upward and downward sameAs links between digest and payload are complicated. Correct processing of OWL taxonomies (perhaps available only at runtime) and message data using taxonomy values is also complicated. A simple UCore 2.0 implementation is possible only if significant simplifying assumptions are made about the messages that will be received.

UCore 2.0 provides a conformance testing tool intended for developers.

UCore 3.0

This version was developed by DoD alone. UCore 3.0 was released in April, 2012.

Intended subject-area domain	all data exchanges
IES vs. IES framework	IES framework
Extensible loose-coupler IES	no
Codelists and taxonomies	OWL format
Governance and CM	none
Compatibility with GML	GML application schema or embedded GML
Compatibility with IC security	mixed
Low-bandwidth utility	low
Implementation complexity	above average
Developer and runtime tools	no

UCore 3.0 provides a framework for constructing any number of conforming IESs. It defines three reusable data components (ThingType, LocatedThingType, and RelationshipType) to be used or extended in every exchange. UCore 3.0 is "universal" in that it is supposed to be applied to every data exchange in the DoD.

UCore 3.0 does not itself prescribe a standard IES. There is an example situation-awareness IES in the distribution, but this is provided only as an illustration. It is theoretically possible to build

[4] *Everything You Wanted To Know About UCore,* (2010)

an extensible loose-coupler IES based on UCore 3.0, but this has not been demonstrated, and would not be part of the specification. Taxonomy value representation, taxonomy definition format, and taxonomy extensions are the same as in UCore 2.0. There is no established governance or CM for community extensions.

UCore 3.0 messages may be GML documents, or may simply embed GML components. UCore 3.0 applies a current version of the ISM standard, and can be easily updated as the ISM standard changes. UCore 3.0 messages may be wrapped as the structured payload within a TDO. However, there is a conflict concerning NTK metadata. UCore 3.0 includes NTK metadata within its messages, while the TDO format calls for NTK metadata to appear within the TDO headers. This conflict can be easily resolved in a future release of UCore 3, TDO, or both.

UCore 3.0 component names were not chosen for brevity. Both the embedded GML components and components defined by UCore 3.0 make extensive use of attribute values which do not compress well. UCore 3.0 messages are not well suited for the low-bandwidth environment, especially when messages are short.

Correct processing of OWL taxonomies and message data is complicated, as in UCore 2. Design of a UCore 3 schema sometimes requires understanding the rules for GML application schemas, as in UCore 1. Apart from that, the design and implementation of a UCore 3 data exchange is unremarkable. The naming and design rules for some of the incorporated standards (e.g., TSPI) are quite complex, but simple "cookbook" implementations are possible without fully understanding those rules.

There are no developer or runtime tools for UCore 3.0.

NIEM

Development and maintenance of the National Information Exchange Model (NIEM) is directed by DHS, DoJ, and HHS. Version 2.1 was released in September 2009. The development plan for version 3.0 is established, and release is expected in September 2013.

Intended subject-area domain	machine-to-machine data exchange
IES vs. IES framework	IES framework
Extensible loose-coupler IES	no
Codelists and taxonomies	XML Schema, OASIS Genericode
Governance and CM	NIEM Business Architecture Committee
Compatibility with GML	embedded GML
Compatibility with IC security	outdated (v2); high (v3)
Low-bandwidth utility	high
Implementation complexity	average
Developer and runtime tools	yes

NIEM provides a collection of reusable data components and a set of rules for composing them into any number of exchange specifications. NIEM also provides governance and a version control architecture for several subject-area domain vocabularies, each controlled by its own

community of interest. NIEM does not itself define any IES. It is possible to build an extensible loose-coupler IES based on NIEM (for example, LEXS). However, NIEM itself does not define any IES.

NIEM does not specify a particular format for codespace or taxonomy values in a message. The code/codespace pattern in UCore 2 and 3 is allowed, but uncommon. There is no explicit mechanism for taxonomy extensions. Codespaces in NIEM 2.1 are captured in XML Schema format. NIEM 3 adds an optional OASIS Genericode representation for codelists. In either version, the codespace definitions must be available during development or runtime, else the consuming application will not understand how to process the message.

NIEM governance extends to the recognition of NIEM domains/COIs, which independently control their subject-area content. Governance also extends to the harmonization of domain content into the NIEM Core. A version control architecture ensures that content defined by one body may be reused by another, without fear of untimely change.

NIEM messages are not GML documents, but may contain embedded GML. NIEM 2.1 applies an outdated version of the ISM standard, but this will be corrected in the 3.0 release. NIEM messages can be wrapped as the structured payload within a TDO.

NIEM component names are not chosen for brevity. However, properly designed NIEM messages usually compress well with EXI. Suitability for low-bandwidth environment will be determined by the individual message design; nothing intrinsic to the NIEM framework poses any difficulty.

The design and implementation of a NIEM data exchange is unremarkable. The naming and design rules for NIEM are lengthy and difficult to comprehend, but simple "cookbook" implementations are possible without fully understanding those rules.

NIEM provides developer tools to assist in IES design, and to test the conformance of an IES and individual messages.

C2 Core

The C2 Data and Services Steering Committee (C2 DSSC) in the DoD controls the development of this standard. Version 2.0 was released in October 2011.

Intended subject-area domain	machine-to-machine data exchange
IES vs. IES framework	IES framework
Extensible loose-coupler IES	no
Codelists and taxonomies	XML Schema
Governance and CM	version control architecture
Compatibility with GML	embedded GML
Compatibility with IC security	high

Low-bandwidth utility	high
Implementation complexity	average
Developer and runtime tools	yes

C2 Core is deliberately patterned after NIEM. It reuses almost all of the NIEM technical framework, but none of its subject-area content. The reusable data components in C2 Core are intended to define concepts needed in more than one of the C2-related COIs. These data components, together with extensions defined by individual COIs, can be composed into any number of C2-related exchange specifications. It is possible to build an extensible loose-coupler IES based on C2 Core (for example, the AF Request Manager pilot). However, C2 Core does not itself define any IES.

C2 Core has not yet determined to follow NIEM in its application of Genericode to codelists and taxonomies. Codespace definitions must be available during development or runtime, else the consuming application will not understand how to process the message. C2 Core copies NIEM's version control architecture. At present there is no explicit governance of domains/COIs.

C2 Core messages are not GML documents, but may contain embedded GML. C2 Core incorporates a current version of ISM, and can be easily updated as that standard changes. C2 Core messages can be wrapped as the structured payload within a TDO. There is no difference between C2 Core and NIEM concerning low-bandwidth, implementation complexity, or tools.

Comparison matrix

	CoT	UCore 1.0	UCore 2.0	UCore 3.0	NIEM	C2 Core
Subject area	primarily SA update	SA update	cross-Department exchange	all data exchanges	M2M data exchange	M2M data exchange
IES/framework	single IES	few IESs	single IES	framework	framework	framework
Provides loose coupler IES	yes	yes	no	no	no	no
Codelists / taxonomies	hierarchical string format	unspecified	OWL	OWL	Schema, Genericode	Schema
Governance & CM	CoT program office	none	none	none	NBAC	version control architecture
GML	transformable content	GML application schema	embedded GML	GAS or embedded GML	embedded GML	embedded GML
Compatibility with IC security	high, for ordinary use	outdated, easily fixed	outdated, but easily fixed	mixed	v2: outdated v3: high	high

Low bandwidth utility	high, but not optimal	low	low	low	high	high
Implementation complexity	very low	above average	high	above average	average	average
Tools	yes	no	no	no	yes	yes

Where Cursor-on-Target excels

CoT may be used in a great many data exchanges, but it especially excels in a particular "ecological niche", one with the following four properties:

- No sophisticated developers or technology. CoT makes minimal use of XML and has no requirements at all for proficiency in other XML technologies (schema, EXI, etc.)
- Bandwidth important but not crucial. CoT reduces the size of its runtime data by using XML attributes instead of elements, by choosing short tag names, and by squeezing all the parts of its "what" taxonomy into a single string value. When bandwidth is not a concern, these choices are unnecessary. When bandwidth is truly crucial, character-based XML is driven out by hand-crafted binary formats or EXI serialization.
- Simple what-where-when data exchange. CoT is great for supplying the geotemporal location of a single entity, especially the sort of entity that can be usefully depicted on a map.
- Good conceptual fit to the CoT taxonomy. Every taxonomy is a hierarchy, and hierarchies are great, but only if it's *your* hierarchy.

Over time, the benefit from low technology requirements will wane, and the single, fixed CoT taxonomy will limit CoT's spread. However, developers will rationally choose CoT for data exchanges in this niche for years to come.

Acknowledgements

This paper was improved by helpful comments from Mike Cokus, Glenda Hayes and Michael Kristan.