

2006 CCRTS
THE STATE OF THE ART AND THE STATE OF THE PRACTICE

Data Schemas for Net-Centric Situational Awareness

Classification: Unclassified

C2 Concepts and Organizations
C2 Modeling and Simulation
Cognitive Domain Issues

Dino Konstantopoulos, Ph.D.
Jeffrey Johnston

Point of Contact: Dino Konstantopoulos

The MITRE Corporation
202 Burlington Road
M/S 1614E
Bedford, MA 01730
(781) 266 9526
dino@mitre.org

The views, opinions and/or findings contained in this report are those of the authors and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

For internal use and not an official position of The MITRE Corporation

©2006 The MITRE Corporation

MITRE
Center for Air Force C2 Systems
Bedford, Massachusetts

Abstract

The Cursor-On-Target (CoT) Event data model defines an XML data schema for exchanging time sensitive position of moving objects, or "what", "when", and "where" (WWW) information, between systems. The CoT data strategy, based on a terse XML schema and a set of sub-schema extensions, is especially amenable to exchanging WWW information between bandwidth-limited hardware. The state of the art for such communication exchanges involves the TCP/IP stack, optimized for sending small packets of information very frequently. In contrast, wireline systems enjoy a high bandwidth capacity, and it is often more economical for such systems to exchange a rich amount of information less often rather than a small amount of information very often. Communications between wireline Net-Centric systems usually involves the XML Web Services protocol, and Web Services are more effective when systems exchange information for thousands of moving objects through a single Web Service call rather than relying on thousands of Web Service calls with individual moving object information. In this paper, we extend the CoT data schema so that it is better adapted to Web Service-based communication exchanges in such a manner that the new schema still enjoys the characteristics that are its hallmark: terseness and extensibility.

1. Introduction

The terminology "Cursor On Target" is drawn from a speech given by Gen. John Jumper, CSAF, at the 2002 C2 summit. During this speech, Gen. Jumper suggested an end state for DoD systems in which they interoperate via machine-to-machine mechanisms much like the onboard systems in an F15. The Cursor On Target initiative at MITRE was started as a means to answer Gen. Jumper's challenge to meet this end-state. The resulting XML schema for the exchange of information that underlies system interoperability focused on time-sensitive position exchange needs including spot reporting, blue force tracking, relocation requests, and any time sensitive position information need, to include, targeting information. Time sensitive "what", "when", and "where" (WWW) information availability is especially critical in asymmetric warfare arenas such as the one in Iraq, where agility and responsiveness is key to military superiority, but also to Homeland Security crisis management and response. "What" tells us if this is a friendly or hostile force; a target to be killed or a survivor to be rescued. "Where" has become synonymous with military GPS accuracy of precision coordinates that guide munitions through windows or navigate tanks through zero visibility sandstorms. "When" is becoming increasingly important as we dramatically shrink the sensor-to-shooter timeline for "time-sensitive-targeting" missions.

Version 2.0 (dated 13-June-2003) of the Cursor-On-Target (CoT) event data model defines a schema for exchanging time sensitive position of moving objects, or WWW information, between DoD systems. That CoT data strategy is based on a terse XML schema and a set of sub-schema extensions. The basic CoT approach is to keep things very simple, and very general and extensible. As such, the CoT schemas are organized much like an object hierarchy used in object-oriented programming. There's a simple base class that provides the basics, and several extension schemas (called "detail sub-schema") that flesh out things for specific applications. The first release of the CoT base schema provides only the most rudimentary what-where-when information. It defines only three entities: event, detail, and point, and has only twelve required attributes. It is very small, but it's sufficient for about 90 percent of all information transfers needed in the Battle Management enterprise (blue force tracking, target deconfliction, strike warnings, ISR taskings, etc.). All the other sub-schemas in the package are "special purpose" extensions that let the same CoT objects carry additional detail where necessary. The innovative aspect of this approach is that even the least complex CoT program can understand every CoT message that gets passed around, whereas smarter applications are not limited in the content of the information they can exchange with other systems via the CoT data schema.

Thanks to the terseness of the first release of the CoT schema, that schema is especially amenable to exchanging WWW information between bandwidth-limited hardware. The state of the art for such communication exchanges involves the TCP/IP stack, which is optimized for sending small packets of information very frequently. However, many wireline DoD systems such as the Joint Mission Planning System (JMPS), -the Next Generation consolidated Mission Planning System for all 4 services and all flying platforms-, and Theater Battle Management and Control System (TBMCS) enjoy a high bandwidth capacity, and it is often more effective and economical for such systems to exchange a rich amount of information less often rather than a small amount of information very often. Moreover, modern DoD Net-Centric architectures employ XML Web Services as communication backbone instead of TCP/IP. Web Service based communications are more effective when systems transmit information about hundreds or thousands of moving objects through a single Web Service call, rather than relying on hundreds or thousands of Web Service calls involving individual moving objects. We propose to extend version 2.0 of the CoT data schema so that it can better adapt to Net-Centric and Service Oriented Architectures (SOA), and to do so in such a manner that the new schema still enjoys the CoT characteristics that are its hallmark: terseness and extensibility.

2. Approach

Version 2.0 of the CoT schema is organized much like an object hierarchy used in object-oriented programming. However, the XML tags defined in that schema involve singular elements (event, detail, and point). Even though there is no roadblock per se to stringing many such elements one after the other within a single file or communication stream, there is no provision for accomplishing this in a unique way that XML data validators can enforce: Should one include first a list of unique events, followed by unique

details and points, or should one string one (event, detail, point) triplet one after the other even if many events, details, and points are possibly repeated? In other words, the CoT schema isn't strongly typed for communication exchanges involving WWW information for thousands of moving objects in a single stream. In the same way that one advocates for strong typing in object oriented programming, one can advocate for strong schemas in hierarchical data sets.

Once a strong schema is developed for uniquely and optimally embedding hundreds or thousands of moving body information within a single XML file transmitted through a Web Service call, the relationships between event, detail, and point becomes very relevant: In a file (or transmission) involving a single (event, detail, point) triplet, the relationship is obvious: it is one-to-one. However, with thousands of events, details, and points, complex many-to-many mappings become possible. To clarify, let us pick two XML tags: event and point. One could say that critical semantics of a WWW Common Operating Picture (COP) consist in which points relate to a single event and how many events a single point is related to (e.g. a troop deployment is a single event that leads to many blue force locations, while a moving target may have a history of multiple "combat engagement" events). Thus, it is not enough to just devise an optimal schema for embedding hundreds or thousands of moving object information within a single communication stream; one must also be able to define mappings between the relevant tags. Lastly, the model for Web-Service based communications is a pull model, as opposed to the push model for TCP/IP-based communications. Thus, communication semantics are different and the CoT schema needs to be optimized for Web Services to accommodate transmissions over time: do we communicate an entire battlefield's worth of WWW information for every Web Service call, or is there a more optimal strategy?

In this paper we extend version 2.0 of the CoT schema to version 3.0 in attempting to address the issues raised above. We also provide data implementation examples, and describe a system prototype that implements version 3.0 of the CoT schema.

3. Extending the CoT schema to multiple moving objects within a single XML stream

The purpose of Version 3.0 of the Cursor on Target Schema for Web Services is to extend the 2.0 schema in such a manner that legacy systems and systems that are bandwidth and power-challenged can still communicate single-target information with minimal overhead increase, while bandwidth-capable systems such as JMPS or TBMCS can throttle WWW information for thousands of moving objects in one XML file, exchanged through XML Web Services.

We begin by giving a short introduction to the façade of the 2.0 version of the CoT schema, which is depicted in figure 1. The schema is really simple: There are 3 XML tags to keep track of: *event*, *detail*, and *point*. The *event* tag gives the context about the geolocalized information (is it an enemy, is it a tank, how long is that target a valid target?). The *detail* tag allows Communities Of Interest to tack on their own schema information so as to add extra information that is specialized to their domain of interest. And the *point* tag represents the coordinates of this geolocalized point. The schema is depicted in Figure 1 below:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Jason Mathews (Mitre Corp) -->
+ <!-- -->
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
- <xs:element name="event">
+ <xs:annotation>
- <xs:complexType>
- <xs:all>
+ <xs:element ref="point" />
+ <xs:element ref="detail" minOccurs="0" />
+ </xs:all>
+ <xs:attribute name="version" use="required">
+ <xs:attribute name="type" use="required">
+ <xs:attribute name="access" type="xs:string" use="optional">
+ <xs:attribute name="qos" use="optional">
+ <xs:attribute name="opex" type="xs:string" use="optional">
+ <xs:attribute name="uid" type="xs:string" use="required">
+ <xs:attribute name="time" type="xs:dateTime" use="required">
+ <xs:attribute name="start" type="xs:dateTime" use="required">
+ <xs:attribute name="stale" type="xs:dateTime" use="required">
+ <xs:attribute name="how" use="required">
+ </xs:complexType>
+ </xs:element>
- <xs:element name="detail">
+ <xs:annotation>
+ <xs:complexType>
+ </xs:element>
- <xs:element name="point">
- <xs:complexType>
+ <xs:attribute name="lat" use="required">
+ <xs:attribute name="lon" use="required">
+ <xs:attribute name="hae" type="xs:decimal" use="required">
+ <xs:attribute name="ce" type="xs:decimal" use="required">
+ <xs:attribute name="le" type="xs:decimal" use="required">
+ </xs:complexType>
+ </xs:element>
+ </xs:schema>

```

Figure 1: Version 2.0 of the CoT Schema (legacy)

We see that a single event can include many points and many details, but that there is no apparent method, for example, for linking one point to many events. For a detailed discussion of the elements and sub-elements of the schema, we refer you to the following URLs: <http://cot.mitre.org> and <http://www.primidi.com/2004/09/19.html#a972>. We now extend this schema by allowing a sequence of points to be added to the same XML information stream, but we do this by grandfathering the *point* XML element to a *points* XML element as depicted in Figure 2 below:

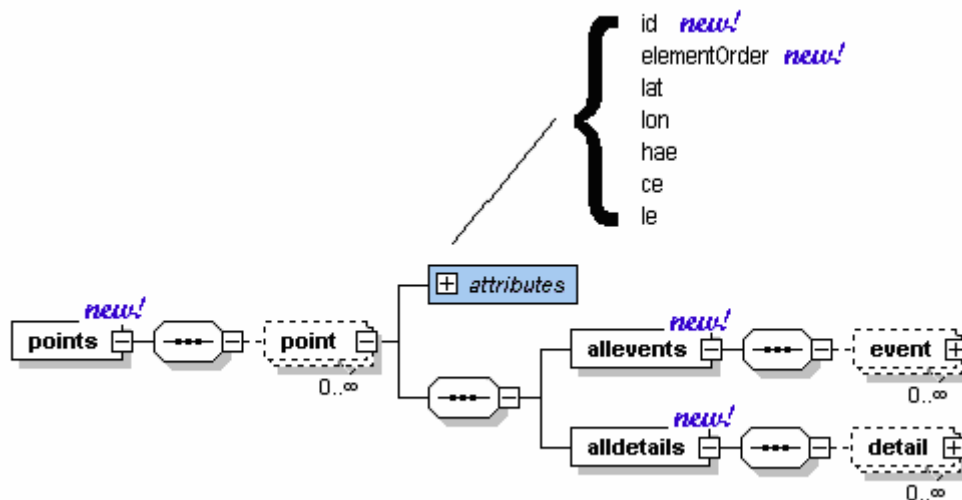


Figure 2: The new *points* XML tag

As we can see, the *point* XML element of version 2.0 was grandfathered into the *points* XML, which allows a sequence of points to become XML children of the *points* XML parent. We will address the new *allevents* and *alldetails* tags and the new *id* and *elementOrder* attributes later on. Now that we have a vector of points in the schema, we need to augment the *event* element to accommodate a vector of points rather than a single point. Further, some of the 10 attributes of the event tag (*version*, *type*, *access*, *qos*, *opex*, *uid*, *time*, *start*, *stale*, *how*) refer to information that may be globally relevant to many points (e.g. “*access*” which determines the security level and intended audience for sharing this information), while others refer to information relevant to individual points (e.g. “*type*” which refers to the nature of the geolocalized point, a tank for example, and “*time*” which refers to the time this point was acquired). Thus, we are led to grandfathering the *event* XML element into the *events* (plural) XML element, much in the same manner we grandfathered the *point* XML element into the *points* XML element. The new *events* XML element is depicted in Figure 3 below, and we once again defer discussion on the new *allpoints* and *alldetails* XML tags:

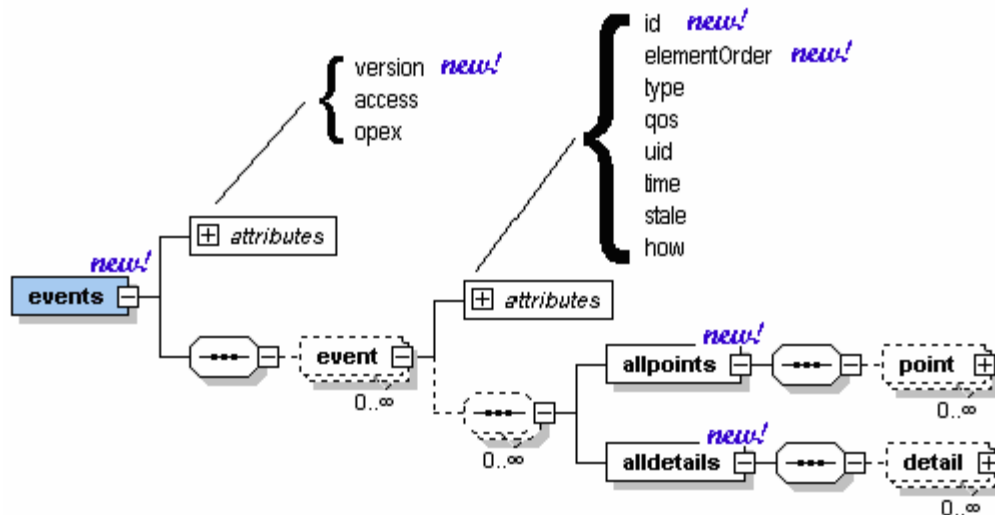


Figure 3: The new *events* XML tag

Note that the *version*, *access*, and *opex* sub-elements now characterize the entire *events* XML element, while the *type*, *qos*, *uid*, *time*, *start*, *stale*, and *how* XML sub-elements and attributes characterize each *point* in the *points* dataset. Please refer to the URLs we gave in the introduction of this document to get more information about each one of these sub-elements, or for the courageous data miners amongst us, drill into the XSD schema file and look at the *annotations* tag for legend clarification.

Finally and predictably, we also grandfather the *detail* tag into a *details* tag in a similar fashion to complete the (event, point, detail) grandfathering trifecta, as depicted in Figure 4 below:

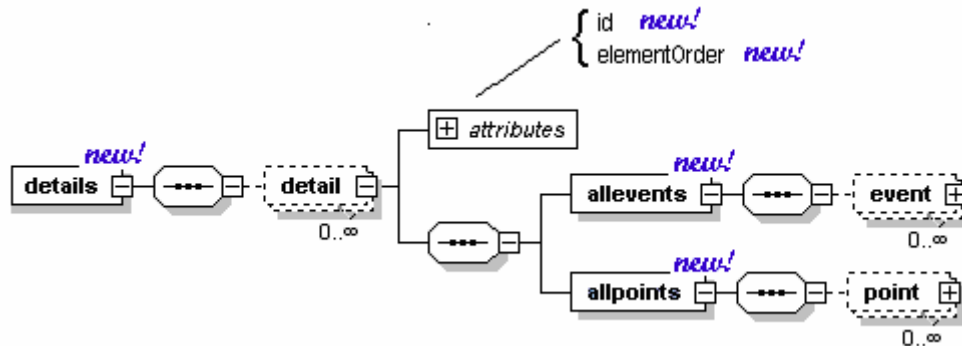


Figure 4: The new *details* XML tag

4. Diffgrams and the Linking of “Event”, “Point”, and “Detail” Information

We can see in Figure 3 above that a universal identifier attribute (*uid*) is associated with each *event*, and thus a natural impulse would be to associate through the *uid* attribute each *point* sub-element in the *points* element with each *event* sub-element in the *events* element (for example by “decorating” each *point* sub-element with the *uid* attribute of an *event* sub-element). However, let’s take a step back and consider that we are devising a schema for potentially *any* Community of Interest (COI), each one of which may have different formats for identifying a point through a *uid* identifier, some more verbose than others. It would thus be a mistake, we feel, to “borrow” this XML sub-element and force a format on the *uid* attribute to accomplish the linking between *event* and *point* (better leave the format unspecified so that each COI can pick one to suit). Moreover, there is another consideration that we need to take into account, and one that is relevant to the task at hand, namely embedding many events, details, and points in a single XML stream.

Realistically, an entire battlefield picture can be “streamed” from one DoD system to another. The authors of the version 2.0 of the CoT schema kept this in mind and thus incorporated into the schema the *stale* attribute. Any geolocalized point can thus be considered “valid” and constitutes actionable information until the current time & date overtakes the time and date identified in the *stale* sub-element. In this way, DoD systems need not be constantly exchanging WWW information in order to keep their Common Operating Picture (COP) synchronized at all times. All they need to do is communicate a new *uid*-identified (event, point) pair as soon as its *stale* date/time has expired. But what if we actually do not know when the validity of a specific point “expires”, when it moves to a new location, when it ceases to exist, or when it is destroyed, in order to decorate it with an appropriate *stale* attribute value? The only solution is to keep on monitoring the point, giving it very short “stale” date/times and continuously transmitting the WWW information between systems. That is clearly not an effective awareness and synchronization strategy. A more effective alternative consists in having an “understanding” between systems that as long as subsets of WWW information remains unchanged, systems need not re-transmit the WWW information for those subsets. Instead, when the WWW information changes for select (event, point) tuples, that information ought to be immediately transmitted between systems. In other words, what is really transmitted between systems sharing WWW information consists of “diffgrams”, or “deltas” which are relative to changed data.

The Microsoft.NET© Framework adopts a similar strategy when transmitting database information between systems: .NET Active Data Objects (ADO.NET©) transmit information diffgrams rather than

transmitting a complete dump of the entire database every time a new database record is inserted, updated, or deleted.

We now begin to see the outlines of a solution that will define our (event, point) tuple linking mechanism and that will allow systems to transmit WWW diffgrams in order to effectively share Common Operating Picture (COP) information with optimal bandwidth usage. We introduce a new XML namespace:

```
targetNamespace=http://cot.mitre.org/ws/provisioning/events
```

We also introduce two new Uniform Resource Names (URN) intended to serve as persistent, location-independent, resource identifiers and designed to make it easy to map other namespaces into URN-space. These are:

```
xmlns:mitredata="urn:schemas-mitre-org:xml-mitredata"
```

```
xmlns:diffgr="urn:schemas-mitre-org:xml-diffgram-v1"
```

Note that the latter one is a mirror of the following URN:

```
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1"
```

that describes the diffgram standard for Microsoft ADO.NET© database rowsets. It is intended that the events CoT schema will be published at the URL above (<http://cot.mitre.org/ws/provisioning/events.xsd>) for all WWW data consumers. Note that alternatively, this schema can be embedded in the XML file being transferred. In keeping with the Web Services-specific data strategy that transmitting a thousand (event, point, detail) triplets in a single XML file is more efficient than a thousand transmissions of every individual (event, point, detail) triplet, embedding the schema information in the XML file being transmitted represents a low overhead, especially if the annotation information is stripped from the embedded CoT schema. Version 3.0-beta-3 of the CoT schema optimized for Web Services is depicted in Figure 4 below: **event**, **point**, and **detail** XML tags are linked within the XML stream through the *diffgr:id* XML attribute, which is different from the *uid* attribute that identifies WWW information in a manner that is appropriate for the Community of Interest at hand. Note that we have introduced an additional attribute: the *mitredata:elementOrder* annotation preserves the row order of each original point in relation to its siblings (so that the XML stream remain context free: tags can be freely reordered by XML parsers without changing data semantics) and identifies the index of a point in a particular WWW dataset. These annotations are all defined in the urn:schemas-mitre-org:xml-mitredata namespace. Figure 5 below details the new *diffgr:id* (required) and *mitredata:elementOrder* (optional) attributes.

```
- <xs:element name="points">
- <xs:complexType>
- <xs:choice maxOccurs="unbounded">
- <xs:element name="point">
- <xs:complexType>
- <xs:sequence>
+ <xs:attribute name="diffgr:id" use="required">
+ <xs:attribute name="mitredata:elementOrder" use="optional">
+ <xs:attribute name="lat" use="required">
+ <xs:attribute name="lon" use="required">
+ <xs:attribute name="hae" type="xs:decimal" use="required">
+ <xs:attribute name="ce" type="xs:decimal" use="required">
+ <xs:attribute name="le" type="xs:decimal" use="required">
- </xs:sequence>
- </xs:complexType>
- </xs:element>
- </xs:choice>
- </xs:complexType>
- </xs:element>
```

Figure 5: The new *diffgr:id* and *mitredata:elementOrder* attributes

As we can see in line 9 of Figure 1 (`<xs:element ref="point" />`), version 2.0 of the CoT schema uses an embedding strategy whereby the entire “point” element is embedded in the “event” element so that there can be no doubts about which point(s) the event is all about. However, this does not represent a context-free XML grammar¹. By taking the point element out of the event element as we accomplished in the previous section, and by linking the two with a *diffgr:id* identifier, it becomes easier to parse a file that may contain thousands of (event, point) tuples and to establish many-to-many semantic relationships between events and points. Indeed, a single event can be comprised of many points, while a single point could potentially belong to many events.

In a similar fashion, we associate *diffgr:id* and *mitredata:elementOrder* attributes to the *event* and *detail* XML tags. One-to-one, one-to-many, many-to-one, and many-to-many relationships between event, point, and detail can now be defined: In figures 2, 3, and 4 and section 3 of this paper, we introduced the *allevents*, *alldetails*, and *allpoints* XML tags, even though we glossed over them and deferred explanation until now: These are vector XML tags that include zero, one, or many XML tags with a single attribute: an integer id that corresponds to the *diffgr:id* of an event, a detail, or a point. In other words, the *allevents*, *alldetails* and *allpoints* XML tags encapsulate pointers to individual *event*, *detail*, and *point* XML tags, thus providing a semantic for describing any kind of relationship between events, details and points. Figure 6 visualizes how the schema implements many-to-many relationships between events, details, and points.

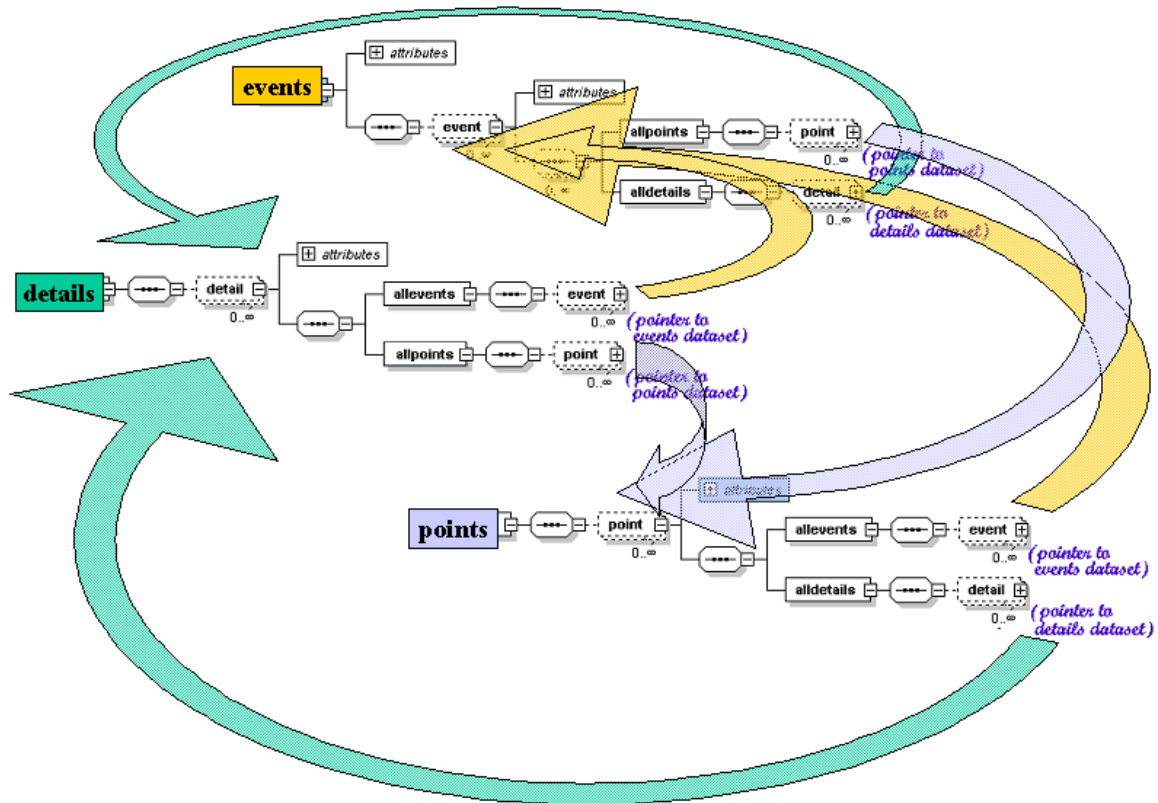


Figure 6: Implementing many-to-many relationships between events, details, and points

In Figure 6, green arrows constitute references to particular details, yellow arrows constitute references to particular events, and teal arrows constitute references to particular points, so that each event can be comprised of one or many points and annotated with meaning by one or many detail elements, each point can belong to one or many events and one or many detail annotations, and each detail can add meaning to one or many points and one or many events.

¹ Context free grammars (CFG) are often the most compact and natural strategies for describing data structures: A CFG provides a tree structure for the data it encapsulates, and tree structures are easier to read and mine by data parsers (the reader is referred to technical literature on context free grammars, often referred to as Extended Backus-Naur Form grammars).

Note that when WWW information is written as a datagram to serialize its contents for transport across a network, it populates the diffgram with all the necessary information to accurately recreate the contents, albeit not the schema itself, including values from both original WWW (event, detail, point) tuples and current (event, detail, point) tuples which may represent updates, inserts, or deletes of already existing information. Moreover, error information and (event, detail, point) tuple ordering information is also conveyed. The diffgram schema derived from the one in use with Microsoft ADO.NET® rowsets is described in the following sections.

5. DiffGram Format

This section describes the diffgram indexing format and can be skipped by readers who are after general ideas rather than technical detail.

The *uid* attribute in the *event* sub-element of the *events* element uniquely identifies an *event* element, while the various *diffgr:id*'s link each event to one or many points and details. Thus, a single *uid* suffices to match an original event to an updated event. In this manner, *uids* allow one to keep track of (event, detail, point) tuples, in order to be able to insert news ones, update old ones, and delete expired ones. However, this would represent a stateful information model from one Web Service call with events, details, and points datasets to another (uid information from one Web Service call must be persisted in a data cache so that it can be compared to the uid information from an ensuing Web Service call). A stateful information model makes good sense in TCP/IP communications, where information payload is at a premium. However, with Internet communications, it is more optimal to minimize communication exchanges and increase the communication payload. We thus opt to add information overhead in each data stream in order to keep the information model stateless: clients need not cache *uids* from one Web Service call to another. We thus choose a diffgram format that affords us to remain stateless (derived from the Microsoft ADO.NET® diffgram) by dividing the payload in two sections: the current data, and the original (or "before") data. This format is detailed in Figure 6 below.

```
<?xml version="1.0"?>
<diffgr:diffgram
  xmlns:mitredata="urn:schemas-mitre-org:xml-mitredata"
  xmlns:diffgr="urn:schemas-mitre-com:xml-diffgram-v1"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <events>
  </events>
  <details>
  </details>
  <points>
  </points>

  <diffgr:before>
  <events>
  </events>
  <details>
  </details>
  <points>
  </points>
  </diffgr:before>

</diffgr:diffgram>
```

Figure 7: The adopted diffgram format

<diffgr:before> contains the original version of the *events* block or of any <event, detail, point> tuple uniquely identified by the uid attribute in the *event* XML element. The last XML element, <diffgr:before>,

is optional so that Communities of Interest that exchange terse CoT WWW information involving few mobile objects, but do so often (e.g. over the TCP/IP communication protocol), are not straddled with extra communication overhead.

Diffgrams also use annotations to relate elements from the different diffgram blocks that represent different row versions of *event*, *detail*, or *point* elements. Figure 7 describes the DiffGram annotations that are defined in the DiffGram namespace **urn:schemas-mitre-org:xml-diffgram-v1**.

Annotation	Description
Id	Used to pair the elements in the <code><diffgr:before></code> and <code><diffgr:errors></code> blocks to elements in the <code><events></code> block.
HasChanges	Identifies a row in the <code><events></code> block as modified. The hasChanges annotation can have one of the following three values: <p>inserted Identifies an Added (event, point) tuple.</p> <p>modified Identifies a Modified (event, point) tuple that contains an Original (event, point) tuple version in the <code><diffgr:before></code> block. Note that Deleted (event, point) tuples will have an Original row version in the <code><diffgr:before></code> block, but there will be no annotated element in the <code><events></code> block.</p>

Figure8: Diffgram annotations

DiffGram processing logic rules to determine whether an (event, detail, point) tuple is an insert (new event), an update of an already existing event, or a delete of an expired event are described in Figure 8.

Operation	Description
Insert	<p>A diffgram indicates an insert operation when an element appears in the <i>events</i> block but not in the corresponding <i>before</i> block, and the <i>diffgr:hasChanges</i> attribute is specified (<i>diffgr:hasChanges=inserted</i>) on the element. In this case, the diffgram instructs that the (event, point) tuple instance that is specified in the <i>events</i> block consists of a new event.</p> <p>If the <i>diffgr:hasChanges</i> attribute is not specified, the element is ignored by the processing logic and no insert is performed.</p>
Update	<p>The diffgram indicates an update operation when there is an (event, detail, point) tuple in the <i>before</i> block for which there is a corresponding element in the <i>events</i> block (that is, both elements have a <i>diffgr:id</i> attribute with same value) and the <i>diffgr:hasChanges</i> attribute is specified with the value <i>modified</i> on the element in the <i>events</i> block.</p> <p>If the <i>diffgr:hasChanges</i> attribute is not specified on the element in the <code><events></code> block, an error is returned by the processing logic.</p>
Delete	<p>A diffgram indicates an expired (event, detail, point) tuple when an element appears in the <i>before</i> block but not in the corresponding <i>events</i> block. In this case, the diffgram deletes the (event, detail, point) tuple that is specified in the <i>before</i> block from any Common Operating Picture.</p>

Figure 9: Diffgram processing rules logic

6. A New “Details” Schema for the Targeting COI

The *details* schema as defined in Appendix F of the DoD’s Intel Interface Control Document (ICD) dated 12 January 2005 contained redundant information that we now excise with a new updated version of that schema. The *Tasking* XML tag was repeated in the *Weapon* and *Target* tags. We delete these repetitions and we end up with the updated version of the *details* schema as depicted in Appendix A of this document.

7. Conclusion and Sample CoT Diffgrams for the Targeting Community

There is one more important difference between TCP/IP-based communications, and Web-Service-based communications: the former employ a push model whereby the server transmits information to clients, the latter reflect a pull model whereby the client polls the server for data. Theoretically speaking, push models are more adapted to transmitting Battlespace awareness/ Common Operating Pictures (COP) information, since the server retains the knowledge of when important data has changed in order to initiate a new transmission with clients. In such cases, high-bandwidth wireline clients that still wish to use Web Services to exchange COP-type data may want to employ a hybrid TCP/IP + Web Services strategy whereby TCP/IP messages are exchanged to notify clients about new data availability, leaving it up to the client to initiate a Web Service call with the server to receive the new data payload. In practice, clients are the final authority on which data is important (to them), and thus it is ultimately better to leave it up to the clients to initiate a pull for new data rather than having the server force-feeding them data that may be fresh but of marginal consequence. Moreover, clients may fuse data from multiple servers, and fusing is easier when the client is in control of data refresh activity. The picture below, from

<http://cot.mitre.org/resources/usecases/M2MSA/M2MSA.ppt>, is an example of a Situational Awareness (SA) display on FalconView© which represents CoT-based fusing from multiple data sources.

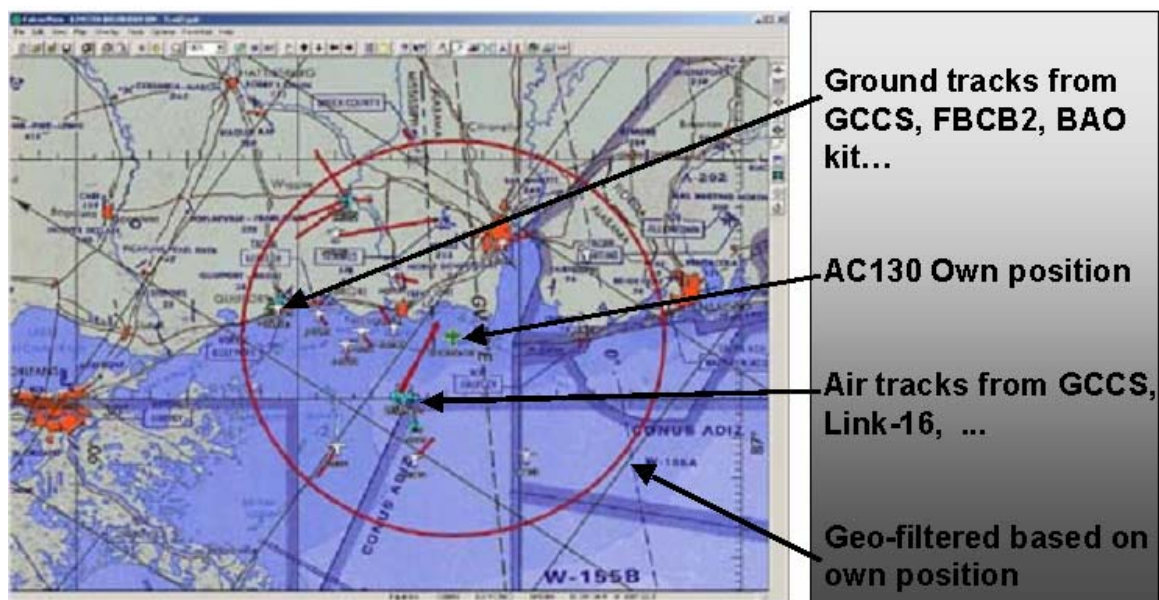


Figure 10: CoT-based Situational Awareness fusion on FalconView

There is a demonstration that is available of the Joint Mission Planning System (JMPS) exchanging Battlespace awareness information with a simulated AWACS server, using TCP/IP and version 2 of the CoT schema, and using Web Services and version 3 of the CoT schema. The Web Services alternative provides a faster system refresh and is a more compelling solution for high-bandwidth wireline systems such as JMPS and TBMCS (better use of system resources).

We conclude by reiterating on the goal of version 3.0 of the CoT schema adapted to Web Services as being two-fold:

- Adapting to information exchange strategies of Net-Centric DoD systems that implement Service Oriented Architecture through the use of Web Services. For these architectures, it is much more efficient to transmit one thousand (event, detail, point) tuples through a single Web Service call, rather than to transmit each individual (event, detail, point) tuple through one thousand Web Service calls. However, as the data payload is increased, so is the schema overhead used to convey data semantics.
- Keeping that overhead at a minimum for bandwidth-limited systems that plan to continue transmitting single (event, detail, point) tuples. These systems employ TCP/IP packets to exchange small amounts of information very often rather than Web Services to exchange larger amounts of information less often.

We now apply version 3.0 of the CoT schema adapted to Web Services to a specific Community of Interest: the targeting community, where (event, detail, point) tuples consist of targets of opportunity that are to be destroyed within specified windows of time. The *detail* schema for the targeting community has been defined in Appendix A of this document. An exemplar of a CoT version 3 compliant XML file containing a single target is provided in Appendix B, proving that version 3 of the CoT schema is almost as terse as version 2 in its classic use of transmitting single (event, detail, point) information. An exemplar of a CoT version 3 compliant XML file containing two targets is provided in Appendix C. Finally, version 3.0 of the CoT schema is included in Appendix D.

Appendix A: A new “detail” element for the targeting Community Of Interest

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- edited with notepad by Dino Konstantopoulos (The Mitre Corporation) -->
- <xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
- <xs:element name="detail">
- <xs:annotation>
- <xs:documentation>...</xs:documentation>
- </xs:annotation>
- <xs:complexType>
- <xs:sequence>
- <xs:element name="PGM">
+ <xs:annotation>
- <xs:complexType>
- <xs:sequence>
- <xs:element ref="TASKING" />
- <xs:element ref="WEAPON" />
- <xs:element ref="TARGET" />
- </xs:sequence>
- </xs:complexType>
- </xs:element>
- </xs:sequence>
- </xs:complexType>
- </xs:element>
- <xs:element name="TASKING">
- <xs:complexType>
- <xs:sequence>
- <xs:element ref="CLASS_LVL" />
- <xs:element ref="DOMAIN_LVL" />
- <xs:element ref="RELEASE_MARK" minOccurs="0" />
- <xs:element ref="CODEWORD" minOccurs="0" />
- <xs:element ref="DECLASS_ON" minOccurs="0" />
- <xs:element ref="DECLASS_ON_DATE" minOccurs="0" />
- <xs:element ref="CONTROL_MARK" minOccurs="0" />
- <xs:element ref="MSN_ID" />
- <xs:element ref="TASK_ORDER_DTG_BEGIN" />
- <xs:element ref="TASK_ORDER_DTG_END" />
- <xs:element ref="CASE_NUM" minOccurs="0" />
- <xs:element ref="DMPI_ID" />
- <xs:element ref="TOT_DATETIME" minOccurs="0" />
- <xs:element ref="TOT_DATETIME_NET" minOccurs="0" />
- <xs:element ref="TOT_DATETIME_NLT" minOccurs="0" />
- <xs:element ref="TASKED_UNIT_NAME" minOccurs="0" />
- </xs:sequence>
- </xs:complexType>
- </xs:element>
- <xs:element name="WEAPON">
- <xs:complexType>
- <xs:sequence>
- <xs:element ref="FUZE_SETTING_ALTITUDE" minOccurs="0" />
- <xs:element ref="FUZE_NAME" minOccurs="0" />
- <xs:element ref="FUZE_DELAY_TIME" minOccurs="0" />
- <xs:element ref="FUZE_ARM_TIME" minOccurs="0" />
- <xs:element ref="FUZE_SETTING_TIME" minOccurs="0" />
- <xs:element ref="FUZE_MODE" minOccurs="0" />
- <xs:element ref="FUZE_DEPTH" minOccurs="0" />
- <xs:element ref="FUZE_COUNT" minOccurs="0" />
- <xs:element ref="TERMINAL_IMPACT_SPEED" minOccurs="0" />
- <xs:element ref="DISPNSR_SPIN_RATE" minOccurs="0" />
- <xs:element ref="DISPNSR_ALTITUDE" minOccurs="0" />
- <xs:element ref="DISPNSR_PAT_DIMENSION" minOccurs="0" />
- <xs:element ref="DISPNSR_PAT_AZIMUTH" minOccurs="0" />
- <xs:element ref="DISPNSR_PAT_TYPE" minOccurs="0" />
- <xs:element ref="DISPNSR_PAT_LENGTH" minOccurs="0" />
- <xs:element ref="DISPNSR_PAT_WIDTH" minOccurs="0" />
- <xs:element ref="DISPNSR_PAT_RADIUS" minOccurs="0" />
- <xs:element ref="PROB_DAMAGE_TOTAL" minOccurs="0" />
- <xs:element ref="PROB_DAMAGE_SORTIE" minOccurs="0" />
- <xs:element ref="EFFECT_IDX_TYPE" minOccurs="0" />
- <xs:element ref="EFFECT_IDX_VALUE" minOccurs="0" />
- <xs:element ref="EFFECT_IDX_VALUE_UM" minOccurs="0" />
- <xs:element ref="DAMAGE_CRITERION" minOccurs="0" />
- <xs:element ref="DMPI_IMPACT_ANGLE" minOccurs="0" />
- <xs:element ref="DMPI_IMPACT_ANGLE_MIN" minOccurs="0" />
- <xs:element ref="DMPI_IMPACT_ANGLE_MAX" minOccurs="0" />
- <xs:element ref="TERMINAL_IMPACT_AZIMUTH" minOccurs="0" />
- <xs:element ref="TERMINAL_IMPACT_AZIMUTH_MIN" minOccurs="0" />
- <xs:element ref="TERMINAL_IMPACT_AZIMUTH_MAX" minOccurs="0" />
- <xs:element ref="WPN_AZIMUTH_AT_DISPENSE" minOccurs="0" />
- <xs:element ref="WPN_QTY" minOccurs="0" />
- <xs:element ref="WPN_NAME" minOccurs="0" />
- <xs:element ref="TERMINAL_AREA_ID" minOccurs="0" />
- </xs:sequence>
- </xs:complexType>
- </xs:element>
```

```

- <xs:element name="TARGET">
  - <xs:complexType>
    - <xs:sequence>
      - <xs:choice>
        - <xs:sequence>
          <xs:element ref="BE_NUMBER" minOccurs="0" />
          <xs:element ref="OSUFFIX" minOccurs="0" />
        </xs:sequence>
        <xs:element ref="UNIT_ID" minOccurs="0" />
      </xs:choice>
      <xs:element ref="DMPI_NAME" minOccurs="0" />
      <xs:element ref="TGT_DTL_NAME" minOccurs="0" />
      <xs:element ref="TARGET_NAME" minOccurs="0" />
      <xs:element ref="COORD" minOccurs="0" />
      <xs:element ref="COORD_DERIV_ACC" minOccurs="0" />
      <xs:element ref="COORD_DERIV_ACC_UM" minOccurs="0" />
      <xs:element ref="COORD_DERIV_ACC_CONF_LVL" minOccurs="0" />
    - <xs:choice>
      - <xs:sequence>
        <xs:element ref="ELEVATION" minOccurs="0" />
        <xs:element ref="ELEVATION_UM" minOccurs="0" />
        <xs:element ref="ELEVATION_DERIV_ACC" minOccurs="0" />
        <xs:element ref="ELEVATION_DERIV_ACC_UM" minOccurs="0" />
        <xs:element ref="ELEVATION_CONF_LVL" minOccurs="0" />
      </xs:sequence>
      - <xs:sequence>
        <xs:element ref="ELEVATION_MSL" minOccurs="0" />
        <xs:element ref="ELEVATION_MSL_UM" minOccurs="0" />
        <xs:element ref="ELEVATION_MSL_DERIV_ACC" minOccurs="0" />
        <xs:element ref="ELEVATION_MSL_DERIV_ACC_UM" minOccurs="0" />
        <xs:element ref="ELEVATION_MSL_CONF_LVL" minOccurs="0" />
        <xs:element ref="GEOIDAL_MSL_SEPARATION" minOccurs="0" />
        <xs:element ref="GEOIDAL_MSL_SEPARATION_UM" minOccurs="0" />
        <xs:element ref="ELEVATION_MSL_ELLIPSOID_REFERENCE" minOccurs="0" />
        <xs:element ref="ELEVATION_MSL_GRID_SPACING" minOccurs="0" />
        <xs:element ref="ELEVATION_MSL_GRID_SPACING_UM" minOccurs="0" />
        <xs:element ref="ELEVATION_MSL_INTERPOLATION_TECHNIQUE" minOccurs="0" />
      </xs:sequence>
    </xs:choice>
    <xs:element ref="SHAPE" minOccurs="0" />
    <xs:element ref="RADIUS" minOccurs="0" />
    <xs:element ref="RADIUS_UM" minOccurs="0" />
    <xs:element ref="AREA_RADIUS" minOccurs="0" />
    <xs:element ref="AREA_RADIUS_UM" minOccurs="0" />
    <xs:element ref="LENGTH" minOccurs="0" />
    <xs:element ref="LENGTH_UM" minOccurs="0" />
    <xs:element ref="AREA_LENGTH" minOccurs="0" />
    <xs:element ref="AREA_LENGTH_UM" minOccurs="0" />
    <xs:element ref="WIDTH" minOccurs="0" />
    <xs:element ref="WIDTH_UM" minOccurs="0" />
    <xs:element ref="AREA_WIDTH" minOccurs="0" />
    <xs:element ref="AREA_WIDTH_UM" minOccurs="0" />
    <xs:element ref="AZIMUTH" minOccurs="0" />
    <xs:element ref="AZIMUTH_REF" minOccurs="0" />
    <xs:element ref="VERTICAL_ORIENT" minOccurs="0" />
    <xs:element ref="HEIGHT" minOccurs="0" />
    <xs:element ref="HEIGHT_UM" minOccurs="0" />
    <xs:element ref="TARGET_SUBCLASS" minOccurs="0" />
    <xs:element ref="TARGET_SUPERCLASS" minOccurs="0" />
    <xs:element ref="LINK_16_TRACK_NUM" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="CLASS_LVL" type="xs:string" />
<xs:element name="DOMAIN_LVL" type="xs:string" />
<xs:element name="RELEASE_MARK" type="xs:string" />
<xs:element name="CODEWORD" type="xs:string" />
<xs:element name="DECLASS_ON" type="xs:string" />
<xs:element name="DECLASS_ON_DATE" type="xs:string" />
<xs:element name="CONTROL_MARK" type="xs:string" />
<xs:element name="MSN_ID" type="xs:string" />
<xs:element name="TASK_ORDER_DTG_BEGIN" type="xs:string" />
<xs:element name="TASK_ORDER_DTG_END" type="xs:string" />
<xs:element name="CASE_NUM" type="xs:string" />
<xs:element name="DMPI_ID" type="xs:string" />
<xs:element name="TOT_DATETIME" type="xs:string" />
<xs:element name="TOT_DATETIME_NET" type="xs:string" />
<xs:element name="TOT_DATETIME_NLT" type="xs:string" />
<xs:element name="TASKED_UNIT_NAME" type="xs:string" />
<xs:element name="FUZE_SETTING_ALTITUDE" type="xs:string" />
<xs:element name="FUZE_NAME" type="xs:string" />
<xs:element name="FUZE_DELAY_TIME" type="xs:string" />
<xs:element name="FUZE_ARM_TIME" type="xs:string" />

```



```
<xs:element name="FUZE_SETTING_TIME" type="xs:string" />
<xs:element name="FUZE_MODE" type="xs:string" />
<xs:element name="FUZE_DEPTH" type="xs:string" />
<xs:element name="FUZE_COUNT" type="xs:string" />
<xs:element name="TERMINAL_IMPACT_SPEED" type="xs:string" />
<xs:element name="DISPNSR_SPIN_RATE" type="xs:string" />
<xs:element name="DISPNSR_ALTITUDE" type="xs:string" />
<xs:element name="DISPNSR_PAT_DIMENSION" type="xs:string" />
<xs:element name="DISPNSR_PAT_AZIMUTH" type="xs:string" />
<xs:element name="DISPNSR_PAT_TYPE" type="xs:string" />
<xs:element name="DISPNSR_PAT_LENGTH" type="xs:string" />
<xs:element name="DISPNSR_PAT_WIDTH" type="xs:string" />
<xs:element name="DISPNSR_PAT_RADIUS" type="xs:string" />
<xs:element name="PROB_DAMAGE_TOTAL" type="xs:string" />
<xs:element name="PROB_DAMAGE_SORTIE" type="xs:string" />
<xs:element name="EFFECT_IDX_TYPE" type="xs:string" />
<xs:element name="EFFECT_IDX_VALUE" type="xs:string" />
<xs:element name="EFFECT_IDX_VALUE_UM" type="xs:string" />
<xs:element name="DAMAGE_CRITERION" type="xs:string" />
<xs:element name="DMPI_IMPACT_ANGLE" type="xs:string" />
<xs:element name="DMPI_IMPACT_ANGLE_MIN" type="xs:string" />
<xs:element name="DMPI_IMPACT_ANGLE_MAX" type="xs:string" />
<xs:element name="TERMINAL_IMPACT_AZIMUTH" type="xs:string" />
<xs:element name="TERMINAL_IMPACT_AZIMUTH_MIN" type="xs:string" />
<xs:element name="TERMINAL_IMPACT_AZIMUTH_MAX" type="xs:string" />
<xs:element name="WPN_AZIMUTH_AT_DISPENSE" type="xs:string" />
<xs:element name="WPN_QTY" type="xs:string" />
<xs:element name="WPN_NAME" type="xs:string" />
<xs:element name="TERMINAL_AREA_ID" type="xs:string" />
<xs:element name="BE_NUMBER" type="xs:string" />
<xs:element name="OSUFFIX" type="xs:string" />
<xs:element name="UNIT_ID" type="xs:string" />
<xs:element name="DMPI_NAME" type="xs:string" />
<xs:element name="TGT_DTL_NAME" type="xs:string" />
<xs:element name="TARGET_NAME" type="xs:string" />
<xs:element name="COORD" type="xs:string" />
<xs:element name="COORD_DERIV_ACC" type="xs:string" />
<xs:element name="COORD_DERIV_ACC_UM" type="xs:string" />
<xs:element name="COORD_DERIV_ACC_CONF_LVL" type="xs:string" />
<xs:element name="ELEVATION" type="xs:string" />
<xs:element name="ELEVATION_UM" type="xs:string" />
<xs:element name="ELEVATION_DERIV_ACC" type="xs:string" />
<xs:element name="ELEVATION_DERIV_ACC_UM" type="xs:string" />
<xs:element name="ELEVATION_CONF_LVL" type="xs:string" />
<xs:element name="ELEVATION_MSL" type="xs:string" />
<xs:element name="ELEVATION_MSL_UM" type="xs:string" />
<xs:element name="ELEVATION_MSL_DERIV_ACC" type="xs:string" />
<xs:element name="ELEVATION_MSL_DERIV_ACC_UM" type="xs:string" />
<xs:element name="ELEVATION_MSL_CONF_LVL" type="xs:string" />
<xs:element name="GEOIDAL_MSL_SEPARATION" type="xs:string" />
<xs:element name="GEOIDAL_MSL_SEPARATION_UM" type="xs:string" />
<xs:element name="ELEVATION_MSL_ELLIPSOID_REFERENCE" type="xs:string" />
<xs:element name="ELEVATION_MSL_GRID_SPACING" type="xs:string" />
<xs:element name="ELEVATION_MSL_GRID_SPACING_UM" type="xs:string" />
<xs:element name="ELEVATION_MSL_INTERPOLATION_TECHNIQUE" type="xs:string" />
<xs:element name="SHAPE" type="xs:string" />
<xs:element name="RADIUS" type="xs:string" />
<xs:element name="RADIUS_UM" type="xs:string" />
<xs:element name="AREA_RADIUS" type="xs:string" />
<xs:element name="AREA_RADIUS_UM" type="xs:string" />
<xs:element name="LENGTH" type="xs:string" />
<xs:element name="LENGTH_UM" type="xs:string" />
<xs:element name="AREA_LENGTH" type="xs:string" />
<xs:element name="AREA_LENGTH_UM" type="xs:string" />
<xs:element name="WIDTH" type="xs:string" />
<xs:element name="WIDTH_UM" type="xs:string" />
<xs:element name="AREA_WIDTH" type="xs:string" />
<xs:element name="AREA_WIDTH_UM" type="xs:string" />
<xs:element name="AZIMUTH" type="xs:string" />
<xs:element name="AZIMUTH_REF" type="xs:string" />
<xs:element name="VERTICAL_ORIENT" type="xs:string" />
<xs:element name="HEIGHT" type="xs:string" />
<xs:element name="HEIGHT_UM" type="xs:string" />
<xs:element name="TARGET_SUBCLASS" type="xs:string" />
<xs:element name="TARGET_SUPERCLASS" type="xs:string" />
<xs:element name="LINK_16_TRACK_NUM" type="xs:string" />
</xs:schema>
```


Appendix B: A simple targeting CoT XML file containing a single target

```
<?xml version="1.0" encoding="UTF-8" ?>
- <TT1 xmlns="CoTWS.xsd">
- <diffgr:diffgram xmlns:mitredata="urn:schemas-mitre-org:xml-mitredata" xmlns:diffgr="urn:schemas-mitre-org:xml-diffgram-v1">
- <events version="3.0">
  <!-- Note that type and how tags were picked at random, while uid, time, start, and stale tags were determined
  from the details section -->
  - <event diffgr:id="event1" elementOrder="1" type="a-f" uid="TEST DMPI NAME TT: 1" time="21155925ep2005"
  start="21155925ep2005" stale="22155925ep2005" how="m-i">
    - <allpoints>
      <point diffgr:id="point1" />
    </allpoints>
    - <alldetails>
      <detail diffgr:id="detail1" />
    </alldetails>
  </event>
</events>
- <points>
  <!-- Note that lat and lon below were converted from degrees/minutes/seconds to decimal degrees -->
  <!-- Mean Sea Level (MSL) values still need to be converted to Height Above Ellipsoid (HAE) values !!! -->
  <!-- Below is the formula for the HAE to MSL correction, as reported by
  http://www.beg.utexas.edu/coastal/poster/vertdatadadjustant.htm -->
  <!-- Hae1 = HAElidar + NG99SESS + MSL correction -->
  <!-- Circular Error (CE) and Linear Error (LE) also need to be determined from PGM data as well !!! -->
  - <point diffgr:id="point1" elementOrder="1" lat="36.216667" lon="115.216667" hae="0" ce="0" le="0">
    - <allevents>
      <event diffgr:id="event1" />
    </allevents>
    - <alldetails>
      <detail diffgr:id="detail1" />
    </alldetails>
  </point>
</points>
- <details>
  - <detail diffgr:id="detail1" elementOrder="1">
    - <allevents>
      <event diffgr:id="event1" />
    </allevents>
    - <allpoints>
      <point diffgr:id="point1" />
    </allpoints>
    - <PGM>
      - <TASKING>
        <CLASS_LVL>U</CLASS_LVL>
        <DOMAIN_LVL>CO</DOMAIN_LVL>
        <CONTROL_MARK>/>
        <MSN_ID>0</MSN_ID>
        <TASK_ORDER_DTG_BEGIN>21155925ep2005</TASK_ORDER_DTG_BEGIN>
        <TASK_ORDER_DTG_END>22155925ep2005</TASK_ORDER_DTG_END>
        <DMPI_ID>DMPI IDENT ATO: 0 TT#: 1</DMPI_ID>
        <TOT_DATETIME>20050921215900</TOT_DATETIME>
        <TASKED_UNIT_NAME>TST BOMBER 1</TASKED_UNIT_NAME>
      </TASKING>
      - <WEAPON>
        <FUZE_SETTING_ALTITUDE>0</FUZE_SETTING_ALTITUDE>
        <FUZE_DELAY_TIME>0.001</FUZE_DELAY_TIME>
        <FUZE_MODE>A</FUZE_MODE>
        <TERMINAL_IMPACT_SPEED>0</TERMINAL_IMPACT_SPEED>
        <DISPNSR_PAT_AZIMUTH>0</DISPNSR_PAT_AZIMUTH>
        <DISPNSR_PAT_LENGTH>0</DISPNSR_PAT_LENGTH>
        <DISPNSR_PAT_WIDTH>0</DISPNSR_PAT_WIDTH>
        <DAMAGE_CRITERION>0</DAMAGE_CRITERION>
        <DMPI_IMPACT_ANGLE>90</DMPI_IMPACT_ANGLE>
        <TERMINAL_IMPACT_AZIMUTH>0</TERMINAL_IMPACT_AZIMUTH>
        <WPN_AZIMUTH_AT_DISPENSE>0</WPN_AZIMUTH_AT_DISPENSE>
        <WPN_QTY>0</WPN_QTY>
        <WPN_NAME>R_UGM-109C</WPN_NAME>
      </WEAPON>
    </PGM>
    - <TARGET>
      <DE_NUMBER>0</DE_NUMBER>
      <OSUFFIX>0</OSUFFIX>
      <DMPI_NAME>TEST DMPI NAME TT: 1</DMPI_NAME>
      <COORD>360130000N1150130000W</COORD>
      <COORD_DERIV_ACC>1</COORD_DERIV_ACC>
      <COORD_DERIV_ACC_UM>FT</COORD_DERIV_ACC_UM>
      <COORD_DERIV_ACC_CONF_LVL>90</COORD_DERIV_ACC_CONF_LVL>
      <ELEVATION_MSL>-1500</ELEVATION_MSL>
      <ELEVATION_MSL_UM>FT</ELEVATION_MSL_UM>
      <ELEVATION_MSL_DERIV_ACC>2</ELEVATION_MSL_DERIV_ACC>
      <ELEVATION_MSL_DERIV_ACC_UM>FT</ELEVATION_MSL_DERIV_ACC_UM>
      <ELEVATION_MSL_CONF_LVL>90</ELEVATION_MSL_CONF_LVL>
      <TARGET_NAME>TEST TARGET NAME TT: 1</TARGET_NAME>
      <VERTICAL_ORIENT>1</VERTICAL_ORIENT>
      <SHAPE>PT</SHAPE>
      <TARGET_SUBCLASS>1</TARGET_SUBCLASS>
      <TARGET_SUPERCLASS>1</TARGET_SUPERCLASS>
      <LINK_16_TRACK_NUM>ZZ000</LINK_16_TRACK_NUM>
    </TARGET>
  </PGM>
</detail>
</details>
</diffgr:diffgram>
</TT1>
```

Appendix C: A simple targeting CoT file containing two targets

(Note: in this example, there are two unrelated (event1, point1, detail1) and (event2, point2, detail2) tuples)

```
<?xml version="1.0" encoding="UTF-8" ?>
- <TT2 xmlns="CoTWS.xsd"
- <diffgr:diffgram xmlns:mitredata="urn:schemas-mitre-org:xml-mitredata" xmlns:diffgr="urn:schemas-mitre-org:xml-diffgram-v1">
- <events version="3.0">
  <!-- Note that type and how tags were picked at random, while uid, time, start, and stale tags were determined
  from the details section -->
  - <event diffgr:id="event1" elementOrder="1" type="a-f" uid="TEST DMPI NAME TT: 2" time="2115592Sep2005"
  start="2115592Sep2005" stale="2215592Sep2005" how="m-i">
    - <allpoints>
      <point diffgr:id="point1" />
    </allpoints>
    - <alldetails>
      <detail diffgr:id="detail1" />
    </alldetails>
  </event>
  <!-- Note that type and how tags were picked at random, while uid, time, start, and stale tags were determined
  from the details section -->
  - <event diffgr:id="event2" elementOrder="2" type="a-f" uid="TEST TARGET NAME TT: 4" time="2115592Sep2005"
  start="2115592Sep2005" stale="2215592Sep2005" how="m-i">
    - <allpoints>
      <point diffgr:id="point2" />
    </allpoints>
    - <alldetails>
      <detail diffgr:id="detail2" />
    </alldetails>
  </event>
</events>
- <points>
  <!-- Note that lat and lon below were converted from degrees/minutes/seconds to decimal degrees -->
  <!-- Mean Sea Level (MSL) values still need to be converted to Height Above Ellipsoid (HAE) values !!! -->
  <!-- Below is the formula for the HAE to MSL correction, as reported by
  http://www.beg.utexas.edu/coastal/poster/vertdataadjustmnt.htm -->
  <!-- Hae1 = HAE1idar + NG998888 + MSL correction -->
  <!-- Circular Error (CE) and Linear Error (LE) also need to be determined from PGM data as well !!! -->
  - <point diffgr:id="point1" elementOrder="1" lat="36.883333" lon="115.883333" hae="0" ce="0" le="0">
    - <allevents>
      <event diffgr:id="event1" />
    </allevents>
    - <alldetails>
      <detail diffgr:id="detail1" />
    </alldetails>
  </point>
  <!-- Note that lat and lon below were converted from degrees/minutes/seconds to decimal degrees -->
  <!-- Mean Sea Level (MSL) values still need to be converted to Height Above Ellipsoid (HAE) values !!! -->
  <!-- Below is the formula for the HAE to MSL correction, as reported by
  http://www.beg.utexas.edu/coastal/poster/vertdataadjustmnt.htm -->
  <!-- Hae1 = HAE1idar + NG998888 + MSL correction -->
  <!-- Circular Error (CE) and Linear Error (LE) also need to be determined from PGM data as well !!! -->
  - <point diffgr:id="point2" elementOrder="2" lat="36.883333" lon="115.883333" hae="0" ce="0" le="0">
    - <allevents>
      <event diffgr:id="event2" />
    </allevents>
    - <alldetails>
      <detail diffgr:id="detail2" />
    </alldetails>
  </point>
</points>
- <details>
  - <detail diffgr:id="detail1" elementOrder="1">
    - <allevents>
      <event diffgr:id="event1" />
    </allevents>
    - <allpoints>
      <point diffgr:id="point1" />
    </allpoints>
  </detail>
  - <PGM>
    - <TASKING>
      <CLASS_LVL>U</CLASS_LVL>
      <DOMAIN_LVL>CO</DOMAIN_LVL>
      <MSN_ID>99999998</MSN_ID>
      <TASK_ORDER_DTG_BEGIN>2115592Sep2005</TASK_ORDER_DTG_BEGIN>
      <TASK_ORDER_DTG_END>2215592Sep2005</TASK_ORDER_DTG_END>
      <DMPI_ID>DMPI IDENT ATO: 0 TT#: 2</DMPI_ID>
      <TOT_DATETIME>20050921215900</TOT_DATETIME>
      <TASKED_UNIT_NAME>TST Bomber 2</TASKED_UNIT_NAME>
    </TASKING>
    - <WEAPON>
      <FUZE_SETTING_ALTITUDE>99998</FUZE_SETTING_ALTITUDE>
      <FUZE_DELAY_TIME>172799.998</FUZE_DELAY_TIME>
      <DISPNSR_SPIN_RATE>9998</DISPNSR_SPIN_RATE>
      <DISPNSR_PAT_AZIMUTH>358</DISPNSR_PAT_AZIMUTH>
      <WPN_AZIMUTH_AT_DISPENSE>358</WPN_AZIMUTH_AT_DISPENSE>
      <WPN_QTY>98</WPN_QTY>
      <WPN_NAME>CBU-103/B</WPN_NAME>
    </WEAPON>
  </detail>
</details>
```

```

- <TARGET>
  <UNIT_ID>ZZZZZZZZZ</UNIT_ID>
  <DMPI_NAME>TEST DMPI NAME TT: 2</DMPI_NAME>
  <COORD>360530000N1150530000W</COORD>
  <COORD_DERIV_ACC>1</COORD_DERIV_ACC>
  <COORD_DERIV_ACC_UM>FT</COORD_DERIV_ACC_UM>
  <COORD_DERIV_ACC_CONF_LVL>50</COORD_DERIV_ACC_CONF_LVL>
  <ELEVATION>+29999</ELEVATION>
  <ELEVATION_UM>FT</ELEVATION_UM>
  <ELEVATION_DERIV_ACC>1</ELEVATION_DERIV_ACC>
  <ELEVATION_DERIV_ACC_UM>FT</ELEVATION_DERIV_ACC_UM>
  <ELEVATION_CONF_LVL>50</ELEVATION_CONF_LVL>
  <TARGET_NAME>TEST TARGET NAME TT: 2</TARGET_NAME>
  <VERTICAL_ORIENT>88</VERTICAL_ORIENT>
  <SHAPE>PT</SHAPE>
  <TARGET_SUBCLASS>ZZZZZZZZZ</TARGET_SUBCLASS>
  <TARGET_SUPERCLASS>ZZZZZ</TARGET_SUPERCLASS>
  <LINK_16_TRACK_NUM>ZZ000</LINK_16_TRACK_NUM>
</TARGET>
</PGM>
</detail>
- <detail diffgr:id="detail2" elementOrder="2">
  - <allevents>
    <event diffgr:id="event2" />
  </allevents>
  - <allpoints>
    <point diffgr:id="point2" />
  </allpoints>
- <PGM>
  - <TASKING>
    <CLASS_LVL>U</CLASS_LVL>
    <DOMAIN_LVL>CO</DOMAIN_LVL>
    <MSN_ID>99999996</MSN_ID>
    <TASK_ORDER_DTG_BEGIN>2115592Sep2005</TASK_ORDER_DTG_BEGIN>
    <TASK_ORDER_DTG_END>2215592Sep2005</TASK_ORDER_DTG_END>
    <DMPI_ID>DMPI IDENT ATO: 0 TT#: 4</DMPI_ID>
    <TOT_DATETIME>20050921215900</TOT_DATETIME>
    <TASKED_UNIT_NAME>TST Bomber 4</TASKED_UNIT_NAME>
  </TASKING>
  - <WEAPON>
    <FUZE_SETTING_ALTITUDE>99996</FUZE_SETTING_ALTITUDE>
    <FUZE_DELAY_TIME>172799.996</FUZE_DELAY_TIME>
    <FUZE_MODE>A</FUZE_MODE>
    <TERMINAL_IMPACT_SPEED>99996</TERMINAL_IMPACT_SPEED>
    <DISPNSR_SPIN_RATE>9996</DISPNSR_SPIN_RATE>
    <DISPNSR_PAT_AZIMUTH>356</DISPNSR_PAT_AZIMUTH>
    <DAMAGE_CRITERION>97</DAMAGE_CRITERION>
    <DMPI_IMPACT_ANGLE>87</DMPI_IMPACT_ANGLE>
    <TERMINAL_IMPACT_AZIMUTH>356</TERMINAL_IMPACT_AZIMUTH>
    <WPN_AZIMUTH_AT_DISPENSE>356</WPN_AZIMUTH_AT_DISPENSE>
    <WPN_QTY>96</WPN_QTY>
    <WPN_NAME>CBU-103/B</WPN_NAME>
  </WEAPON>
  - <TARGET>
    <BE_NUMBER>ZZZZZZZZZ</BE_NUMBER>
    <OSUFFIX>ZZZZZ</OSUFFIX>
    <DMPI_NAME>TEST DMPI NAME TT: 4</DMPI_NAME>
    <COORD>360530000N1150530000W</COORD>
    <ELEVATION>+29997</ELEVATION>
    <ELEVATION_UM>FT</ELEVATION_UM>
    <TARGET_NAME>TEST TARGET NAME TT: 4</TARGET_NAME>
    <VERTICAL_ORIENT>86</VERTICAL_ORIENT>
    <SHAPE>PT</SHAPE>
    <TARGET_SUBCLASS>ZZZZZZZZZ</TARGET_SUBCLASS>
    <TARGET_SUPERCLASS>ZZZZZ</TARGET_SUPERCLASS>
    <LINK_16_TRACK_NUM>ZZ000</LINK_16_TRACK_NUM>
  </TARGET>
</PGM>
</detail>
</details>
</diffgr:diffgram>
</TT2>

```

Appendix D: Version 3.0 of the CoT schema adapted to Web Services

(Note: as of this reading, in order to open this schema successfully with Altova's XMLSpy product, please replace all occurrences of **diffgr:id** with **id** and all occurrences of **mitredata:elementOrder** with **elementOrder**)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema id="CursorOnTargetWebServices" targetNamespace="http://cot.mitre.org/ws/provisioning/events"
xmlns="http://cot.mitre.org/ws/provisioning/events" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:mitredata="urn:schemas-mitre-org:xml-mitredata" xmlns:diffgr="urn:schemas-mitre-org:xml-diffgram-v1"
attributeFormDefault="qualified" elementFormDefault="qualified">
  <xs:attribute name="version" default="3">
    <xs:simpleType>
      <xs:restriction base="xs:decimal">
        <xs:minInclusive value="3"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:element name="events">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>Events Definition</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="event" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
              <xs:element name="allpoints">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="point" minOccurs="0"
maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:attribute name="diffgr:id"
use="required">
                          <xs:simpleType>
                            <xs:restriction base="xs:integer">
                              <xs:minInclusive value="1"/>
                            </xs:restriction>
                          </xs:simpleType>
                        </xs:attribute>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="alldetails">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="detail" minOccurs="0"
maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:attribute name="diffgr:id"
use="required">
                          <xs:simpleType>
                            <xs:restriction base="xs:integer">
                              <xs:minInclusive value="1"/>
                            </xs:restriction>
                          </xs:simpleType>
                        </xs:attribute>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
```

```

<xs:attribute name="diffgr:id" use="required">
  <xs:annotation>
    <xs:documentation>Identifier for linking
this event to point(s) and/or detail(s).</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:simpleType>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="mitredata:elementOrder" use="optional">
  <xs:annotation>
    <xs:documentation>Identifier for
reconstituting the ordering of all event elements of the events set.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:simpleType>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="type" use="required">
  <xs:annotation>
    <xs:documentation>

```

The "type" attribute is a composite of components delimited by the semi-colon character. The first component of this composite attribute is defined below. Future versions of this schema will define other components which we expect will aid in machine filtering. Despite the exclusion of definitions for additional components in this version of the schema, users of this schema should expect and design an optional trailing field delimited by the semi-colon character. This field can be ignored.

component1;optional field

The first component (component1) is a hierarchically organized hint about type. The intention is that this hierarchy be flexible and extensible and facilitate simple filtering, translation and display. To facilitate filtering, the hierarchy needs to present key fields in an easily parsed and logical order. To facilitate this, this component is a composite of fields separated by the "-" punctuation character, so a valid type would be: x-x-X-X-x. Using a punctuation for field separation allows arbitrary expansion of the type space, e.g., a-fzp-mlk-gm-...

Field meanings are type specific. That is, the third field of an "atom" type may represent air vs. ground while the same field for a "reservation" type may represent purpose.

The "Atoms" portion of the type tree requires some additional explanation past the taxonomy defined below. The "Atoms" portion of the type tree contains CoT defined fields and part of the MIL-STD-2525 type definition. To distinguish MIL-STD-2525 type strings from CoT defined fields, the MIL-STD-2525 types must be represented in all upper case. Differentiation of type namespace with upper/lower case facilitates extension of CoT types and MIL-STD-2525 types without name space confliction. An example:

a-f-A-B-C-x

The organization of CoT and MIL-STD-2525 types can be determined from the taxonomy below, but additional details are provided here.

The "Atoms" portion of the "type" tree contains the "Battle Dimension" and "Function ID" fields taken from MIL-STD-2525. "Battle Dimension" is a single character taken from MIL-STD-2525.

P - Space
A - Air
G - Ground
S - Sea Surface

U - Sea Subsurface
X - Other

The typical 2525 representation for "Function ID" is three groups of two characters separated by a space (e.g. "12 34 56"). The CoT schema maps this to a "-" delimited list of characters. (e.g. "1-2-3-4-5-6"). The concatenation of the "Battle Dimension" and "Function ID" fields from the MIL-STD-2525 specification represented in the CoT schema will be as follows:

battle dimension-func id char1-func id char2- ... -func id char6

for example: a-h-G-U-C-D-M-L-A

is a hostile Ground-based "AIR DEFENSE MISSILE MOTORIZED (AVENGER)"

When an appropriate MIL-STD-2525 type exists, it should be used. If there is a MIL-STD-2525 representation which is close, but may be refined, a CoT extension to the 2525 type can be appended. For example...

for example: a-h-G-U-C-D-M-L-A-i might represent

hostile Ground-based "AIR DEFENSE MISSILE MOTORIZED (AVENGER)" of Israeli manufacture. Again, the CoT extension uses lower case. Conceptually, this extension defines further branching from the nearest MIL-STD-2525 tree point.

If no appropriate 2525 representation exists, a type definition can be added to the CoT tree defined here. The resulting definition would be represented in all lower case. For example

a-h-G-p-i

might define atoms-hostile-Ground-photon cannon-infrared.

The taxonomy currently looks like this: Note that the coding of the sub fields are determined entirely by the preceding fields!) The current type tree is defined here.

+--- First position, this event describes
|
V

a - Atoms - this event describes an actual "thing"

+--- CoT affiliation of these atoms
|
V

p - Pending
u - Unknown
a - Assumed friend
f - Friend
n - Neutral
s - Suspect
h - Hostile
j - Joker
k - Faker
o - None specified
x - Other

+--- Battle dimension
| Taken from MIL-STD-2525 "Battle Dimension" (upper case)
|
V

P - Space
A - Air
G - Ground
S - Sea Surface
U - Sea Subsurface

X - Other

+--- Function (dimension specific!)
| Taken from MIL-STD-2525 function fields (upper case)
|
V
...
U-C-D-M-L-A - AIR DEFENSE MISSILE MOTORIZED (AVENGER)
...
U-C-A-A-A-T - ANTI ARMOR ARMORED TRACKED
...

+--- The event describes ...

|
V

b - Bits - Events in the "Bit" group carry meta information about raw data sources. For example, range-doppler radar returns or SAR imagery represent classes of information that are "bits". However, tracks derived from such sources represent objects on the battlespace and this have event type "A-..."

The intention with the "Bit" type is to facilitate the identification of germane information products. This hierarchy is not intended to replace more detailed domain-specific meta information (such as that contained in NITF image headers or the GMTI data formats), rather it is intended to provide a domain-neutral mechanism for rapid filtering of information products.

+--- Dimension

|
V

i - Imagery

e - Electro-optical
i - Infra red
s - SAR
v - video
...

r - Radar

m - MTI data
...

d - Sensor detection events

s - Seismic
d - Doppler
a - Acoustic
m - Motion (e.g., IR)

m - Mapping

p - Designated point (rally point, etc.)
i - initial points
r - rally points
...

r - Reservation/Restriction/References

Events in this category are generally "notices" about specific areas. These events are used for deconfliction and conveyance of significant "area" conditions. Generally, the "point" entity will describe a conical region that completely encloses the affected area. The details entity will provide more specific bounds on precisely the region affected.

u - Unsafe (hostile capability)

o - Occupied (e.g., SOF forces on ground)

c - Contaminated (NBC event)

c - chemical
x - agents, direction,
y
z

f - Flight restrictions

t - Tasking (requests/orders)

Events in this category are generalized requests for service. These may be used to request for data collection, request mensuration of a specific object, order an asset to take action against a specific point. Generally, the "details" entity will identify the general or specific entity being tasked.

s - Surveillance

r - Relocate

e - Engage

m - Mensurate

c - Capability (applied to an area)

s - Surveillance

r - Rescue

f - Fires

d - Direct fires

i - Indirect fires

l - Logistics (supply)

f - Fuel

...

c - Communications

Examples:

a-f-A-U-C-V-R-A -> atoms-friendly-air-attack rotary wing

a-a-S-C-L-B-B -> atoms-assumed friend-sea surface-battleship

c-c -> communications capability

b-i-i -> indication that there exists infrared imagery

t-r -> tasking to relocate

</xs:documentation>

</xs:annotation>

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="\w+(-

\w+)*;(^[^;]*)?"/>

</xs:restriction>

</xs:simpleType>

</xs:attribute>

<xs:attribute name="qos" use="optional">

<xs:annotation>

<xs:documentation>

format - digit-character-character as defined below

The QoS attribute will determine the preferential treatment events receive as they proceed through the kill chain. The field has several distinct but related components.

A "priority" value indicates queuing and processing order for competing events. At a processing bottleneck (e.g., bandwidth limited link) high priority events will be processed before lower priority events. Priority determines queuing order at a bottleneck.

9 - highest (most significant) priority

0 - lowest (least significant) priority

A "overtaking" value indicates how two events for the same uid are reconciled when they "catch up" to one another. The more recent event (by timestamp) may supersede the older event (deleting the old event) when it catches it, or it may follow the old event so that event order is preserved, or it may be routed independently of previous events.

r - replace - new event replaces (deletes) old event

f - follow - new event must follow previous events

i - independent - new event processed independently of old events

An "assurance" value indicates how much effort must be placed in delivering this particular event. Events from sources that

continually send updates (blue force tracks) or that are sent for information purposes only require a lower level of delivery effort. Events that are singletons (sent only once) and are critical require guaranteed delivery.

g - guaranteed delivery (message never dropped even if delivered late)
d - deadline (message dropped only after "stale" time)
c - congestion - message dropped when link congestion encountered

Thus, a valid QoS field looks like:

qos="1-r-c"

Note that different events with the same UID may have differing QoS values. This enables a graceful degradation in the presence of congestion. For example, a blue force tracker may output a sequence of events with like

< ... qos="1-r-c" ... > <= frequent, low priority updates
< ... qos="1-r-c" ... >
< ... qos="1-r-c" ... >
< ... qos="5-r-d" ... > <= occasional "push" priority update
< ... qos="1-r-c" ... >

...

< ... qos="9-r-g" ... > <= A "Mayday" position report

</xs:documentation>

</xs:annotation>

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="d-\w-\w"/>

</xs:restriction>

</xs:simpleType>

</xs:attribute>

<xs:attribute name="uid" type="xs:string" use="required">

<xs:annotation>

<xs:documentation>

The "uid" attribute is a globally unique name for this specific piece of information.

Several "events" may be associated with one UID, but in that case, the latest (ordered by timestamp), overwrites all previous events for that UID.

</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="time" type="xs:dateTime" use="required">

<xs:annotation>

<xs:documentation>

The CoT XML schema includes three time values:

time, start, and stale. "time" is a time stamp placed on the event when generated. start and stale define an interval in time for which the event is valid. Example: For the scenario where we have intel reports about a meeting of terrorist operatives later in the day: An event might be generated at noon (time) to describe a ground based target which is valid from 1300 (start) until 1330 (stale). All time fields are required. In version 1.1 of the CoT schema, the time and stale attributes together defined and interval of time for which the event was valid. In V2.0, time indicates the "birth" of an event and the start and stale pair define the validity interval.

The "time" attribute is a time stamp indicating when an event was generated.

The format of time, start, and stale are in standard date format (ISO 8601):

CCYY-MM-DDThh:mm:ss.ssZ; e.g., 2002-10-05T17:01:14.00Z.

</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="start" type="xs:dateTime" use="required">

<xs:annotation>

<xs:documentation>

format - DTG

The "start" attribute defines the starting time of the event's validity interval. The start and stale fields together define an interval in time. It has the same format as time and stale.

</xs:documentation>

</xs:annotation>

```

</xs:attribute>
<xs:attribute name="stale" type="xs:dateTime" use="required">
  <xs:annotation>
    <xs:documentation>

```

The "stale" attribute defines the ending time of the event's validity interval. The start and stale fields together define an interval in time. It has the same format as time and start.

```

</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="how" use="required">
  <xs:annotation>
    <xs:documentation>

```

format = character-character

The "how" attribute gives a hint about how the coordinates were generated. It is used specifically to relay a hint about the types of errors that may be expected in the data and to weight the data in systems that fuse multiple inputs. For example, coordinates transcribed by humans may have digit transposition, missing or repeated digits, estimated error bounds, etc. As such, they may require special attention as they propagate through the kill chain (e.g., they may require an additional review). Similarly, machine generated coordinates derived solely from magnetic sources may be subject to known anomalies in certain geographical areas, etc.

- h - human entered or modified (someone typed the coordinates)
- e - estimated (a swag by the user)
- c - calculated (user probably calculated value by hand)
- t - transcribed (from voice, paper, ...)
- p - cut and paste from another window
- m - machine generated
- i - mensurated (from imagery)
- g - derived from GPS receiver
- m - magnetic - derived from magnetic sources
- s - simulated - out of a simulation
- f - fused - corroborated from multiple sources
- c - configured - out of a configuration file
- p - predicted - prediction of future (e.g., a from a tracker)
- r - relayed - imported from another system (gateway)

As with other compound fields, the elements of the how field will be delimited by the field separator character "-". E.g, A coordinate mensurated from imagery would have a how field of "m-i".

```

</xs:documentation>
  </xs:annotation>
</xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:pattern value="\w-|\w"/>
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="version" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:decimal">
      <xs:minInclusive value="3"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="access" type="xs:string" use="optional">
  <xs:annotation>
    <xs:documentation>
      The access field is intended to indicate who has access to this
      event. (e.g. unrestricted, nato, army, coalition...)
      It is currently defined as a string, and is optional in V2.0.
      Future version of the event schema will provide formal
      definition of this field.
    </xs:documentation>

```

```

        </xs:annotation>
      </xs:attribute>
    <xs:attribute name="opex" type="xs:string" use="optional">
      <xs:annotation>
        <xs:documentation>

```

The opex field is intended to indicate that the event is part of a live operation or an exercise. For backward compatibility, absence of the opex indicates "no statement", which will be interpreted in an installation specific manner.

opex="o-<name>" or "e-<nickname>"

o = operations
e = exercise

```

        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
</xs:element >
<xs:element name="details">
  <xs:complexType>
    <xs:annotation>
      <xs:documentation>
        A list of Detail elements, each of which has schema defined outside of this document...
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="detail" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:annotation>
            <xs:documentation>

```

format = XML schema defined outside of this document

Detail entities...

The "detail" entity is intended to carry information that is specific to smaller communities of producers and consumers and require more intimate knowledge of the operating domain. For example, mensurated "target" events may come from dramatically different sources and need to propagate dramatically different "detail" information. A close-air-support mission will augment target details with initial point and callsign details to facilitate coordination of weapon delivery. In contrast, a mission planning system may augment planned targets with target catalog information and weapon fuzing requirements.

```

# For a SOF-generated target, details aid weapon delivery
&lt;event ... uid="yosh#laser#21" type="a-h-G-f" ...
  &lt;detail&gt;
    &lt;nine-line callsign="jagged shadow" ip="charlie" equip="BareBack"
...&gt;
    remarks="Square building, NE corner of clearing"
  &lt;/nine-line&gt;
  &lt;/detail&gt;

# Targets from a mission planning system have different details
&lt;event ... uid="mgb@raindrop.020402.1" type="a-h-G-f" ...
  &lt;detail&gt;
    &lt;target
      be="1234ca5678" osuffix="xyzyz" dmpi="MB0002" shape="180,45,12"
      cep="90" fuzing="impact" ...
    /&gt;
  &lt;/detail&gt;

```

The detail entity may contain a number of community-specific entities, and these entities may be added and removed as the events propagate through the kill chain.

```

&lt;event ... type="a-h-G-f" ...
  &lt;detail&gt;
    &lt;nine-line callsign="jagged shadow" ... /&gt;
    &lt;docs target-number="AB1234" tailnumber="TY33" platform="F16"...

```

```

/&gt;
  &lt;link16 track="1402" rr="0035" weapon="00135" c2node="00037" ... /&gt;
&lt;/detail&gt;

```

Because the "details" portion of the event are of interest only to a subset of subscribers, that entity may be mentioned by reference when the event is communicated. This reduces the congestion when events are transmitted over bandwidth limited links and also prevents the retransmission of static data elements.

```

&lt;event ... uid="yosh#laser#21" type="a-h-G-f" ...
  &lt;detail link="http://cot.hanscom.af.mil/targets?uid=yosh#laser#21" /&gt;
  ...

```

The latter example shows a complete URL being provided as the link. In practice, it is more likely that URLs (or other links) will be implicit based on event type and will reference virtual servers, not specific hosts. This will provide for information redundancy and reduce communications bottlenecks. Use of implicit links will also reduce the amount of redundant information transferred in the published events. An initial query may result in a redirection to the server with specific detailed knowledge of that event.

This event:

```

&lt;event ... uid="yosh#laser#21" type="a-h-G-f" ...
  &lt;detail link="targets" /&gt;
  ...

```

indicates that there are more details available via a link to the "target" virtual server. An initial query to "http://targets.af.mil?yosh#laser#21" may resolve to any number of hosts depending on the topological location of the source of the query and the organization responsible for the detailed information. E.g., "http://targets.af.mil?yosh#laser#21" may resolve to "sol.aoc.usafe.mil/targets?yosh#laser#21" while a similar query to "http://targets.af.mil?mgb@raindrop.020402.1" may resolve to "mission-planning.pentagon.mil..."

```

</xs:documentation>
</xs:annotation>
<xs:sequence>
  <xs:element name="allevents">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="event" minOccurs="0"
maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="diffgr:id"
use="required">
              <xs:simpleType>
                <xs:restriction base="xs:integer">
                  <xs:minInclusive value="1"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="allpoints">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="point" minOccurs="0"
maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="diffgr:id"
use="required">

```

```

<xs:restriction base="xs:integer">
  <xs:minInclusive value="1"/>
</xs:restriction>

</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="diffgr:id" use="required">
  <xs:annotation>
    <xs:documentation>Identifier for linking
this detail to event(s) and/or point(s).</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="mitredata:elementOrder" use="optional">
  <xs:annotation>
    <xs:documentation>Identifier for
reconstituting the ordering of all detail elements of the dataset.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:anyAttribute processContents="skip"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="points">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="point" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="allevents">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="event"
minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                      <xs:attribute
name="diffgr:id" use="required">
                        <xs:simpleType>
                          <xs:restriction base="xs:integer">
                            <xs:minInclusive value="1"/>
                          </xs:restriction>
                        </xs:simpleType>
                      </xs:attribute>
                    </xs:complexType>
                  </xs:sequence>
                </xs:complexType>
              </xs:sequence>
            </xs:complexType>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

minOccurs="0" maxOccurs="unbounded">
    <xs:element name="alldetails">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="detail"
                    <xs:complexType>
                        <xs:attribute
                            name="diffgr:id" use="required">
                                <xs:simpleType>
                                    <xs:restriction base="xs:integer">
                                        <xs:minInclusive value="1"/>
                                    </xs:restriction>
                                </xs:simpleType>
                                </xs:attribute>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="diffgr:id" use="required">
            <xs:annotation>
                <xs:documentation>Identifier for linking
                    this point to event(s) and/or detail(s).</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="1"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="mitredata:elementOrder" use="optional">
            <xs:annotation>
                <xs:documentation>Identifier for
                    reconstituting the ordering of all point elements of the dataset.</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="1"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="lat" use="required">
            <xs:annotation>
                <xs:documentation>Latitude based on
                    WGS-84 ellipsoid in signed degree-decimal format (e.g. -33.350000). Range -90 -> +90.</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:decimal">
                    <xs:minInclusive value="-90"/>
                    <xs:maxInclusive value="90"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="lon" use="required">
            <xs:annotation>
                <xs:documentation>Longitude based on
                    WGS-84 ellipsoid in signed degree-decimal format (e.g. 44.383333). Range -180 -> +180.</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:decimal">
                    <xs:minInclusive value="-180"/>
                    <xs:maxInclusive value="180"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:element>

```

```

</xs:attribute>
<xs:attribute name="hae" type="xs:decimal" use="required">
  <xs:annotation>
    <xs:documentation>HAE acronym for
Height above Ellipsoid based on WGS-84 ellipsoid (measured in meters).</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="ce" type="xs:decimal" use="required">
  <xs:annotation>
    <xs:documentation>
Circular Error around point defined by lat and lon fields in meters. Although
named ce, this field is intended to define a circular area around the event point, not
necessarily an error (e.g. Describing a reservation area is not an
"error"). If it is appropriate for the "ce" field to represent
an error value (e.g. event describes laser designated target), the
value will represent the one sigma point for a zero mean
normal (Guassian) distribution.
</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="le" type="xs:decimal" use="required">
  <xs:annotation>
    <xs:documentation>
Linear Error in meters associated with the HAE field. Although named le, this
field is intended to define a height range about the event point, not
necessarily an error. This field, along with the ce field allow for the
definition of a cylindrical volume about the point. If it is appropriate
for the "le" field to represent an error (e.g. event describes laser
designated target), the value will represent the one sigma point for
a zero mean normal (Guassian) distribution.
</xs:documentation>
  </xs:annotation>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```