MTR090357

MITRE TECHNICAL REPORT

MITRE

C2 Core and UCore Message Design Capstone

Interoperable Message Structure

M. David Allen, Catherine Macheret, Mary Ann Malloy

September 2009

This page intentionally left blank.

MTR090357

MITRE TECHNICAL REPORT

MITRE

C2 Core and UCore Message Design Capstone

Interoperable Message Structure

M. David Allen, Catherine Macheret, Mary Ann Malloy

September 2009

Sponsor: ESE Capstone
Dept. No.: E540
Contract No.: FA8721-09-C-009
Project No.: 0709ECSE-DC
Downgrade: N/A
Derived By: N/A
Declassify On: N/A

Abstract

UCore allows users to create multi-layered messages pairing UCore digests with detailed structured payloads. A MITRE Capstone was funded to explore the feasibility of a multi-layered message including a domain specific common core layer. The Capstone Team designed a multi-layered messaging solution where a UCore digest was paired with a notional "common core" payload from the Command and Control (C2) domain, and further extended with a Community of Interest (COI) payload. This approach was tested with an exchange satisfying real air operations planning requirements. Our design goals were to minimize content duplication and to create a message interpretable by both COI and unanticipated users from the domain. This document shares lessons learned from this approach and provides insights about challenges to designing and using multi-layer messages in the real world.

This page intentionally left blank.

Executive Summary

Background

The C2 Core is a Command and Control (C2) data exchange specification built on top of the extendable UCore exchange specification. The C2 Core vision and principles were developed under the auspices of the Technical Framework Architecture and Tools Sub-Working Group (TFTSWG) within the C2 Portfolio. Stakeholders were expected to pilot and evaluate the baseline to determine whether the value propositions are attainable. The value propositions sought through C2 Core are:

- Reduced developer labor from reusing C2 data constructs through a layered interoperability design that promotes reuse
- Broader consumption and derived value from exchanges between primary partners by communities beyond the originally intended exchange partners

MITRE undertook a Capstone effort to test the feasibility of C2 Core design principles using a real message exchange scenario. We implemented a sample information exchange specification (IES) using the layered interoperability approach promoted by UCore and C2 Core, applied it to a real message exchange scenario, and then assessed whether the promised value proposition had in fact been realized. Thus, the ultimate objective was to test the feasibility of developers implementing new information exchanges using the Capstone implementation of the C2 Core design principles.

Experimental Approach

Since the Capstone effort was initiated before the testable baseline was released, the Capstone Team had to fill in some gaps in the technical framework, including developing a notional C2 Core data model sufficient to represent the content required for its use cases. This report has two purposes:

- To document the design decisions the Team made to fill in framework gaps as part of their experiments
- To provide lessons learned based on the exploration of the two use cases that employed notional C2 Core IESs

The UCore, and therefore the C2 Core, promotes a layered architecture for information exchange, as depicted in Figure ES-1. In compliance with this architecture, message content is spread across three structural layers: UCore Digest, C2 Core Payload, and Community of Interest (COI) Payload. However, the C2 Core Concept Document fails to specify a strategy for distributing information across the layers and for creating linkages among them so that content can be consumed meaningfully by at-large or specialized consumers within the C2 domain (and potentially by non-C2 consumers as well). The Capstone Team chose to use each layer to represent only the content appropriate for that layer. Although this introduces some semantic dependency among the layers, it saves labor by enabling IES implementers and message consumers focused on a given layer to reuse models for concepts already provided at any of the higher layers. To minimize information redundancy, the team chose to use an approach called "chaining pointers" in which objects in each layer point to their corresponding objects in the

layers above. Rather than repeating information in each layer, the pointers are used to aggregate all the detail for a specific real world concept or object.

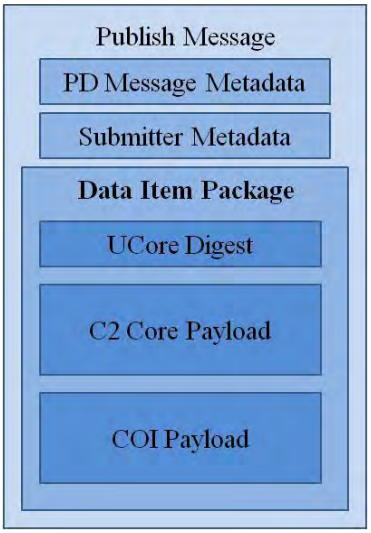

Figure ES-1: Layered Architecture for Information Exchange

Next the team reviewed the Common Mission Definition (CMD) and various other C2 data models to build a notional C2 Core data model in sufficient detail to support creating a message exchange specification for their use cases. Since neither UCore nor C2 Core includes strong guidance for choosing which parts of exchanged information should be represented in the respective message layers, the Team had to develop its own heuristics. Section 8 discusses (down to the eXtensible Markup Language [XML] code level) how the Team chose to distribute content based on the CMD across the three layers and to link it together according to their design decisions.

With all these pieces in place, the Team executed the use case experiments. The scenario for the first use case involved a notional (C2 Core) consumer who needs Air Operations data to coordinate missions for preventing conflicts and friendly fire in a given geographic area. The scenario for the second use case involved a typical Air Operations COI consumer who needs to gather information distributed across all three message layers to determine if there are sufficient resources to carry out a mission. The Team used the Theatre Battle Management Command System (TBMCS) to generate sample CMD messages as inputs for both use cases. These were programmatically transformed into the three-layer message format. The Team noted that substantial testing and debugging time was required ahead of time in manually checking the transformation process as there are no automated tools to do so. Similarly, to consume the messages, the Team had to handcraft programs using Java, Eclipse, and XMLBeans, since there is no standardized Application Programming Interface (API) for consuming layered messages.

Findings

In both use cases, the test programs were successful in correctly producing the expected outputs, indicating that the three-layer messages were sufficiently detailed to be useful for both C2 Core and COI consumers. At the same time, the Capstone Team documented how using messages compliant with an IES based on the C2 Core (and UCore) approach to information exchange is not a simple task. As indicated above, quite a bit of infrastructure needed to be built to support even simple scenarios. While part of this is attributable to the experimental nature of the investigation, the Team includes in this report many well-substantiated reasons why C2 Core (and UCore) proponents need to provide additional design specification details and support tools before value propositions can be realized.

A layered architecture for information exchange, such as that advocated by C2 Core (and UCore), introduces special developer challenges and requires specificity to avoid introducing interoperability break points. The Capstone Team grouped specific challenges into four areas:[1]

- Differences in the modeling paradigms among the three layers, plus a lack of guidance for how to distribute information across layers, increases the risk of inconsistent content representation and poses greater complexity to message consumers.

- The UCore Digest does not always guarantee provision of sufficient content to support stand-alone understanding of messages by unanticipated consumers.

- Additional rules are needed to help define C2 Core and to ensure messages consistently and accurately satisfy applicable rules.

- There is a lack of guidance to IES developers regarding how to link information across layers.

Problems uncovered by the Capstone Team have possible solutions. For example, the C2 Core Specification can provide formal guidance regarding information distribution across the message layers and linking strategies among them. Standardized message access APIs would also help. But without such provisions, it is unlikely that implementing C2 Core layered message production and consumption software will be any less time consuming, expensive, or difficult than non-layered options.

[1] A detailed discussion of these challenges can be found in section 9.

Acknowledgements

This C2 Core UCore Capstone was led by Phil Barry. Members of the project team other than the report authors are David Czulada, Rick Knowles, Carol Mahoney, and Scott Renner.

Technical editing of this paper was expertly performed by Patti Reynolds.

Table of Contents

List of Figures and Tables

This page intentionally left blank.

1 Background

1.1 C2 Core Concept

In January 2009, the Command and Control (C2) Portfolio Manager (CPM) began working on a C2 Common Core. The intent of C2 Core was to address one of the Independent Assessment Team's November 2008 recommendations to the C2 CPM; namely, that the C2 CPM develop a "set of C2-specific extensions to the UCore" to promote net-centric information sharing in the C2 space.

The C2 Core Technical Framework Architecture and Tools Sub-Working Group (TFTSWG) developed and delivered the C2 Core Development Concept. It articulated portions of a technical framework based on UCore, as well as a value proposition for the C2 Core. A team of working groups prepared a testable baseline of C2 Core, which was delivered on 30 June 2009. It includes notional sample content, naming and design rules, and a technical framework sufficiently complete for pilot and test implementations, although not sufficiently mature for broad implementation in Department of Defense (DoD) programs.

1.2 Purpose of C2 Core Capstone

MITRE performed this C2 Core Capstone to evaluate the C2 Core technical framework and the value proposition it promises to deliver. Because this effort started before the testable baseline was delivered, some of Capstone design choices are different from those made in the 30 June baseline. While the Capstone Team used the developmental technical framework, we had to fill in some design and implementation gaps. The testable baseline chose different options from the Capstone Team, demonstrating that many open questions still need to be resolved by the TFTSWG, and that the C2 Core surely will change. In other words, at this time the testable baseline was not expected to be the final word on technical framework choices.

When the C2 Core Capstone effort started, there were few implementations of multi-layered UCore messages in existence. Therefore, an important goal for the Capstone Team was to implement a sample information exchange specification (IES) using the layered interoperability approach promoted by UCore and C2 Core, and then to assess whether the promised value proposition had in fact been realized.

1.3 Purpose of Document

This document presents technical work performed to evaluate the feasibility of implementing a message design standard based on principles found in the UCore and C2 Core message specifications. The document presents the Capstone Team's design decisions and implementation details. Lessons learned and the successes and challenges from the implementation approach are intended to provide input to the C2 Core TFTSWG. We hope to contribute to the goal of improving information sharing across the C2 community.

1.4 UCore and C2 Core Value Propositions

The C2 Core Capstone message design was motivated by the UCore and C2 Core value propositions for improving information dissemination. These value propositions are as follows:

- UCore promises to reduce labor for creating new information exchanges and to extend the value of partner exchanges to broader communities. The premise is that popular

"universal" information concepts such as when and where should need only to be modeled once, rather than for every new exchange agreement, and likewise software written to consume those concepts should also only be written once and interoperate with messages from all UCore message producers. UCore information exchanges use a few pre-established, broadly applicable data structures in a fixed exchange framework. The framework has the capacity to contain additional detail needed by the primary exchange purpose but in a way that makes messages intended for specific partners partially consumable by a larger community.

- The C2 Core concept builds on this UCore premise. The value proposition for C2 Core is that, over time, data components needed by multiple C2 Communities of Interest (COIs) will be designed once and reused by all. These core data components will be used in the implementation of all new exchanges. When these new exchanges are designed, some (or most) of the data interoperability work is already done, because the participants already understand the core components. The result is reduced cost for the enterprise. More importantly, the "prefabricated" data components from UCore, C2 Core, and the C2 COIs, together with the "assembly instructions" found in the C2 Core technical specifications, permit much faster implementation of new and unanticipated information sharing needs. [2]

UCore and C2 Core both seek not only to reduce labor for exchange partners, but also to extend the information value of those exchanges. They do this by providing a design that allows some of the exchange to be consumed by a broader community than the primary exchange partner.

1.5 Assumptions

This paper assumes technical familiarity with the UCore 2.0 final specification, which can be found at https://www.ucore.gov/, though a brief review is provided in Section 4. This paper also assumes basic familiarity with the technical details of eXtensible Markup Language (XML) and how instance documents are formatted and represented. No formal understanding of the C2 Core is required, since each part of the experiment infrastructure will be explained as needed to understand the use cases.

[2] C2 Core Frequently Asked Questions: "What is the problem statement and value proposition of C2 Core?" – document version 1.2, 30 June 2009. This document is part of the June 30, 2009 testable baseline submission package to the DoD Metadata Registry, which can be accessed here: http://metadata.dod.mil/mdr/ns/C2 CORE/C2_Core.zip

2 Experimental Approach

The goal of the project was to demonstrate UCore and C2 Core value propositions using a layered data interoperability approach. The team chose Air Operations as the experimental COI. More specifically, we narrowed our scope to air operations mission planning. We used Air Operations' Common Mission Definition (CMD) message standards to provide the detail we needed to create a realistic C2 and COI XML schema models. We did not, however, use CMD itself in our message design; it merely provided the requirements for such an exchange. Our experiment scenario begins with the idea that a new IES is needed.

We abstracted notional C2 Core data components from the CMD to create an experimental C2 Core model. The purpose of our C2 Core model was to play the role of the future official C2 Core model in our layering experiment. Finally we created a model for the COI layer of the message.

Thus having all these pieces in place, the Team was ready to execute use case experiments. The scenario for the first use case involved a notional consumer who needs Air Operations data to coordinate missions for preventing conflicts and friendly fire in a given geographic area. A program was created to take as inputs a space/time box together with a series of three-layer direct attack air missions, and to determine whether any of them were within the space/time box. The scenario for the second use case involved a typical Air Operations COI consumer who needs to gather information distributed across all three message layers to determine if there are sufficient resources to carry out a mission. The Theatre Battle Management Command System (TBMCS) was used to generate CMD messages as inputs for both use cases. These were programmatically transformed into the three-layer message format. To consume the layered messages, we created programs using Java, Eclipse, and XMLBeans.

Before discussing the design and implementation in more detail we explain the notion of layered data interoperability and explain how UCore fits into that paradigm.

3 Layered Data Interoperability

A typical technique for representing application data is to create one model with all the complexities and detail needed for that specific application. All requirements are fulfilled in one complex model. However, the information represented in that model is accessible only to consumers who can process the complexities of the whole model. Furthermore, even one change to a minor detail in the data model can break existing consumer software.

Layered data interoperability is an alternative concept used by UCore and further exploited in the C2 Core Development Concept[3]. In this approach, information content is separated into layers, each targeted for different consumers. Figure 1 from the C2 Core Development Concept shows the difference in information abstraction for different layers. The layers become progressively more specific (as the level numbers increase) and so the consumer scope similarly becomes more specific. By supporting higher levels of understanding in a message originally intended for a specific purpose, the potential audience is broadened. Also as the representation becomes more abstract it becomes applicable to more domains; therefore structures used to represent the abstracted version of the information are more reusable.

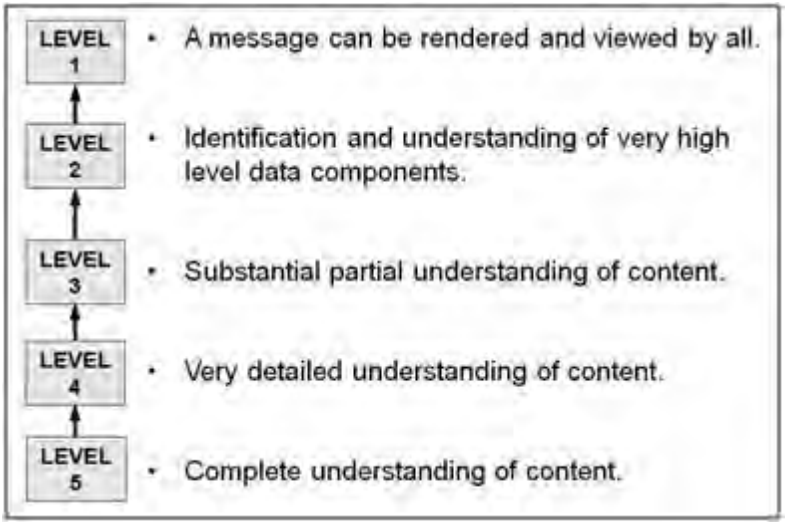

Figure 1: Data Layers

This concept of layered data interoperability is central to the experiment performed in this Capstone task and is the basis for some UCore and C2 Core value propositions.

[3] The C2 Core Development Concept document (version 1.1) is part of the June 30, 2009 testable baseline submission package to the DoD Metadata Registry, which can be accessed here: http://metadata.dod.mil/mdr/ns/C2CORE/C2_Core.zip

4 UCore Review

UCore specifies a data model for the abstract information layer in its layered message architecture. The layer is called the Digest. Universal Lexical Exchange (ULEX) is the message framework model in which the Digest sits along with message metadata and other message parts. There are linking mechanisms in ULEX to link data across layers. The following subsections present a brief UCore review of the Digest model and the ways in which UCore can be extended to represent more specific information. Refer to the UCore Documentation[4] for more detailed explanations.

4.1 UCore Digest

Figure 2 is a conceptual depiction of the UCore Digest. The UCore Digest model is a high-level representation of key concepts found in most exchanges, namely *Who*, *What*, *Where* and *When* along with some relationship objects used to associate things to one another.

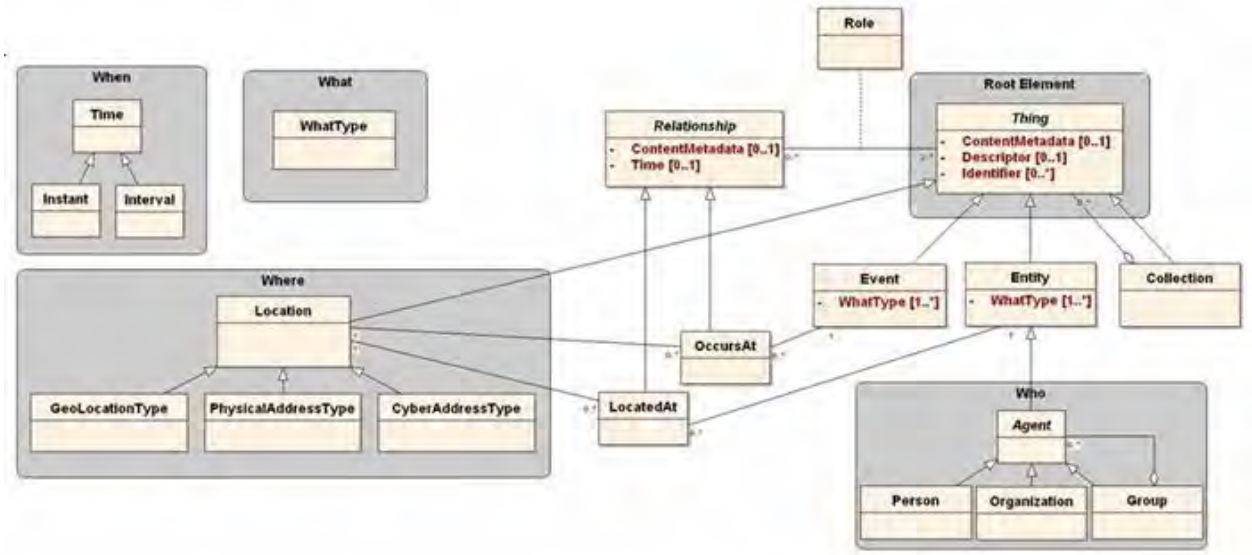

Figure 2: UCore Digest Conceptual Data Model

The UCore concepts *When* and *Where* are represented using accepted pre-existing standards and have enough detail for consumer comprehension. The UCore *What* concept is divided into Event and Entity objects. An Event occurs at a time and location. An Entity can be just about anything left over. Special entity types exist to represent *Who* concepts, such as person and organization. They have name properties and can participate in certain kinds of relationships.

Except for specialized *Who* objects, *What* objects are void of any classification by themselves. To provide more classification detail for *What* objects without relying on schema structure (i.e., named types with special properties), *What* objects can be assigned taxonomy terms in a message instance. This makes the Digest flexible without requiring any change in schema or structure.

[4] Detailed documentation on the UCore conceptual model can be found at https://ucore.gov/ucore/node/20. At the time of this writing, access to the UCore website was open to anyone with a sponsor-approved account.

UCore has its own taxonomy that is expected to be understood by UCore consumers. *What* objects can also be assigned taxonomic terms from other communities who may choose to define their terms in a namespace UCore calls a "codespace." Attaching community codes adds community-specific classification to the object but they are only guaranteed to be understood by that community.

4.2 UCore Extension Options

The UCore specification provides several choices for extending the message content beyond the basic UCore Digest. Using one or more options, C2 Core will have to extend the UCore Digest to represent C2 content in a UCore message. The word "extend" is used generically in the phrase "extend the UCore," but it has a very specific meaning in object-oriented data modeling having to do with deriving new data types from existing data types. Note that this difference in meaning can cause confusion. Only one of the options for extending the UCore is to use the "data type extension" technique. This technique is discussed in section 4.2.2.1.

UCore extension possibilities fall into two categories:

1. Adding content in the Digest layer using existing digest object properties
2. Putting additional content into added message layers, namely Structured Payloads

The Capstone Team reviewed the options and chose several that met the C2 Core goals, and rejected others that we deemed contrary to the C2 Core value propositions.

4.2.1 Adding Content to the UCore Digest

The Digest schema is fixed but flexible. Message instances can extend the meaning of Digest objects by additional assigned domain-specific terms as discussed previously or through added name-value pair strings.

4.2.1.1 Supplementing the UCore Taxonomy

Message producers can enrich the meaning of Digest objects by assigning taxonomic terms from specific domains using the "What Code" property. The UCore base taxonomy will usually be too generic to be useful for the primary business purpose of a message. Figure 3 shows an example person instance having additional taxonomic terms defined by other domains or communities. In this example, the UCore person is classified further in a message instance as a "Crook," and along with other terms meaningful to various communities.

```
<ucore:Digest>
    ...
    <ucore:Person id="jmd">
        <ucore:What ucore:code="Person" ucore:codespace="http://ucore.gov/ucore/2.0/codespace/"/>
        <ucore:What ucore:code="Thief" ucore:codespace="http://yournamespace.com/"/>
        <ucore:What ucore:code="Shoplifter" ucore:codespace="http://another.com/"/>
        <ucore:What ucore:code="Crook" ucore:codespace="http://somewhereelse.com/"/>
        <ucore:What ucore:code="Criminal" ucore:codespace="http://mynamespace.gov/"/>
        <ucore:What ucore:code="Pilferer" ucore:codespace="http://dictionary.reference.com/browse/"/>
    ...
    </ucore:Person>
...
</ucore:Digest>
```

Figure 3: Concrete example of UCore taxonomy extension

Although supplementing the UCore base taxonomy can make the Digest more useful, the domain-specific terms are not meaningful to a basic UCore Digest parser. There is no guarantee that consumers will recognize special code spaces, or that they will be prepared to use the additional classifications in a meaningful way without a prior sharing agreement.

4.2.1.2 Use UCore Simple Properties

UCore Digest objects only have a few generic properties; for most types those are identifiers, what codes, and a comment field. To supplement these, the UCore Digest object has a built in extension technique. This is a property called a "SimpleProperty" which can be used repeatedly to add name-value pairs as "run-time" object properties in the message instance, rather than as part of a revised schema. A message producer can add "Eye Color" – "Blue" to the UCore Entity representing a person.

Most of a domain's additional required detail could be declared using "Simple Properties". However, adding properties in this manner does not provide the normal validation and format predictability that comes from using schema defined properties. Additionally, using these properties in the digest does not guarantee that UCore Digest consumers are prepared to consume them. The Simple Property essentially annotates a UCore Object with arbitrary properties at run time for those consumers prepared to receive them.

4.2.2 Adding Content using Message Layers

One key feature of UCore is its ability to contain additional, separate layers in a single message that can be processed separately by consumers prepared to handle those layers. UCore does not specify how these extra layers are populated, but users have several options. Separate payload schemas will typically be developed to represent the extra content. A payload schema can be designed to be completely independent of the digest and other layers or can somehow be integrated with the digest object types or other payload object types.

There are two ways to integrate schemata in one layer with schemata in another layer for the purpose of representing more detailed information concepts. Object type specialization is one way. A new specialized type inherits the original type's properties and meaning and takes on additional properties and a more specific meaning. In XML this is done with the XML Schema extension element[5]. We are calling the second way to integrate schema components type augmentation. In this method new types are created with the intention of being linked to other types like puzzle pieces completing a bigger picture. Each one of the pieces or data components has a unique set of properties used to augment other pieces or data components.

Both of these methods "extend" the capability to represent information beyond the base UCore. The UCore Digest object is meant to represent only some portion of a concept in the digest layer. Extended objects with additional detail about a concept appear in other layers (the payloads).

4.2.2.1 Type Specialization

A consequence of deriving new types from an existing digest type is that the new type inherits its parent properties. The result of using an extended digest type in the next payload layer is that the digest level information gets duplicated in the payload. Now suppose the message needs a

[5] See http://www.w3schools.com/schema/el_extension.asp for an example of this approach

second payload to contain information for an even more specific purpose. If type specialization is used again, the third layer containing the new type will have the properties of its parent and its parent's parent, thus continuing to duplicate information in the prior message layers.

4.2.2.2 Type Augmentation

The term "augmentation" here means linking additional properties and/or relationships to pre-existing data objects without changing their pre-existing structures. Pieces of a concept are created separately and linked together with pointers. The primary utility for data coupling described in Sections 5.3 and 8.2 is to allow a concept to be distributed into separate structures for different audiences and also to be combined as needed to form a whole concept. The advantage here is that the original structure is still recognizable by its original consumer while at the same time declaring additional detail for a new consumer. This approach is one way to achieve interoperable messages across multiple consumer groups. For this reason the Capstone used this approach. The application of this approach to the Capstone use case is described in the next section.

5 Capstone Message Design

The message architecture chosen for the Capstone experiment aimed to demonstrate the UCore and C2 Core value propositions. The UCore architecture separates some message content represented in a standard generic model from the remainder of the message content represented using domain specific models and agreed to by specific exchange partners. Both UCore and C2 Core would like to capitalize on this interoperable layered data architecture. The key premise for this approach is each layer along with the layers above it serves a certain community. Starting from the bottom layer going up the stack the community served becomes broader. This sort of layering enables some portion of the message to be understood by a community broader than the original exchange partners.

High-level architectural decisions are discussed in this section. Many lower-level design decisions are discussed in Section 8.

5.1 Employing Layered Message Architecture

The Capstone C2 Core message design uses the ULEX containers to create a separation between a "universal" community, a C2 community, and a COI community. Beyond the UCore Digest, the message content is separated into content applicable to the "at large" C2 community and content for the "specific" C2 community. A C2 Core message using UCore will have a UCore Digest and two payload layers: a C2 layer and a separate special COI layer as illustrated in Figure 4. The latter two layers are both implemented as ULEX structured payloads. The goal is to produce messages that have some value to the C2 community in general while also satisfying the messaging needs of a specific C2 COI.

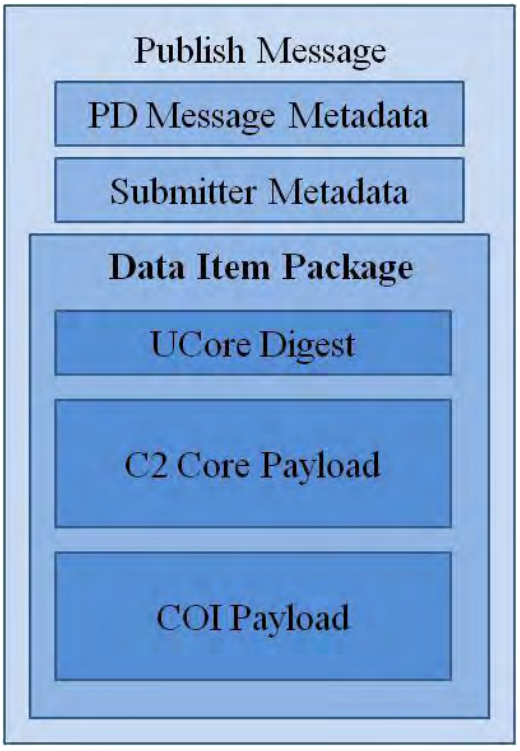

Figure 4: Layered Message Structure

With this architecture, message content is spread across the various layers such that information appropriate for UCore consumers appears in the UCore Digest, information appropriate for the C2 Core consumers appears in both the Digest and the C2 Structured Payload, and information appropriate for COI consumers appears in all three layers.

The C2 Core Development Concept Document (version 1.1)[6] promotes a layered architecture. Some parts of the document state that the C2 layer will mix generic C2 content and COI content together[7]. A third layer is suggested only for extending the COI specification to represent unique details beyond the scope of the COI IES.

In other parts of the document, the COI layer and the C2 Core layer are depicted separately[8]. This configuration was favored by the Capstone Team because it allows unanticipated consumers to choose as many or as few of the layers as they understand. Each layer is well separated and documented with payload-specific metadata. That is, this configuration serves three potential groups of consumers:

1. Those outside the C2 domain who understand only UCore
2. Those in the C2 domain who understand both UCore and C2 Core
3. COI members who understand all three layers

Separating the C2 Core layer from the COI layer follows the pattern of UCore, doing "UCore the UCore way," and providing maximum opportunity to demonstrate the value proposition of "layered interoperability."

5.2 Information Distributed Across Layers

There are several ways to promote message interoperability across multiple user groups using the layered architecture concept. Starting with the UCore Digest and moving down the stack of layers, information becomes more community-specific and has fewer target consumers. One way to populate separate layers is to treat each layer as a semantically independent sub-message. Independent sub-messages can be processed and understood without information from any of the other layers. This design satisfies interoperability requirements for multiple user groups because the target consumer can continue to process his own layer regardless of what appears in the other layers.

Many applications of UCore to date have used the separate layers for semantically independent content. This leads to one layer duplicating message details already specified by other layers with potentially different data structures. This duplication promotes proliferation of multiple schemas for the same concepts. The Capstone Team rejected this semantically independent layer approach for the following reasons:

[6] The C2 Core Development Concept document (version 1.1) is part of the June 30, 2009 testable baseline submission package to the DoD Metadata Registry, which can be accessed here: http://metadata.dod.mil/mdr/ns/C2CORE/C2_Core.zip

[7] For example, section 10.1.1 ("Architectural Framework Overview") and Figure 4 specifically within that section, which depicts C2 Core content and COI extensions within the same StructuredPayload.

[8] C2 Core Development Concept v1.1, Figure 5, "Architectural Layered Interoperability"

- ***It fosters redundancy***: Creating independent layers can lead to repeating the same information up to three times (i.e., one time for each layer) with varying degrees of generality. The message size is increased, as is the potential for contradictory information. Additionally, since each layer has an independent message, every consumer is guaranteed to discard two of the three layers.

- ***It requires extraneous effort***: The message producer is forced to devise message parts that have no benefit for his primary target consumer. This is likely to impede adoption of the standard.

- ***It inhibits reuse***: The message designer will tend to re-create schemas for concepts already represented in other layers. For example, the Capstone Team observed this in the first draft version of C2 data components, where *time* and *location* structures are re-created regardless of the existing UCore *time* and *location* representation. This apparently was done to make consumption of the C2 layer possible without information in the UCore Digest layer.

Rather than treating the layers as semantically independent, the Capstone effort used each layer to represent only the content appropriate for that layer, and avoided duplicating information already represented in another layer. The layers are semantically dependent; consumption of one layer requires the consumption of all the layers higher in the stack. The C2 Payload contains only the information directed to the C2 community at large and does not contain information already represented in the UCore Digest layer. For example, the C2 Payload can use *time* and *location* representations in the UCore Digest.

The COI Payload, the third layer in the message stack, is reserved for message details outside the scope of the C2 Payload and UCore Digest target consumers. Separating COI information from C2 information makes the C2 layer consumable by all C2 participants, while letting the COI extend C2 Core in a separate layer to include COI-specific information.

Keeping the layers separate but semantically dependent is desirable for three key reasons:

- In the future, COIs need not spend time re-modeling concepts already provided for them by UCore and C2 Core; instead they can use “prefabricated data components” as referred to by the C2 Core value proposition.

- Consumers need not create new software to understand C2 content in a COI message, ensuring that as much of the message is useful to as many consumers as possible.

- Message redundancy is minimized, improving performance.

Separate layers maintain interoperability for three different communities without requiring redundant information and duplicative design and development efforts. The data coupling described next provides the necessary information for aggregating the semantically dependent parts of the message.

5.3 Data Coupling Across Layers

ULEX "pointers" are used to connect information across the Digest, C2 Core Payload, and COI Payload message layers. The following example consists of notional content for explanation purposes only.

Figure 5 provides an example of a UCore Digest that specifies a male person named "John Doe." In Figure 6 the C2 Core layer further specifies that he is an infantry officer assigned to the 108th Infantry Division. Figure 7 shows how a COI Payload might further specify some aspect of his training readiness; but the COI Payload does not repeat his name, his sex, or his unit assignment. Instead, the COI Payload contains a `<ulexlib:SameAsDigestReference>` element linking his extra data (i.e., training, readiness, relationship to operation) back to his UCore Digest data (i.e., name, sex). In some cases, `<ulexlib:SameAsPayloadReference>` may also be used to link a COI Payload element to a C2 Core Payload element.

```
<ucore:Digest>
  <ucore:Person id="P1">
    <ucore:What ucore:code="Person"
    ucore:codespace="http://ucore.gov/ucore/2.0/codespace/"/>
    <ucore:Name>
      <ucore:FullName>
        <ucore:Value>John Doe</ucore:Value>
      </ucore:FullName>
    </ucore:Name>
    <ucore:Sex>male</ucore:Sex>
  </ucore:Person>
</ucore:Digest>
```

Figure 5: Notional UCore Digest

```
<ulex:StructuredPayload>
  <ulex:StructuredPayloadMetadata>
  <c2core:message xmlns:c2core="c2core:1.0">
    <c2core:Officer>
      <ulexlib:SameAsDigestReference ulexlib:nvref="P1"/>
      <c2core:Rank>O2</c2core:Rank>
      <c2core:Unit>108th Infantry Division</c2core:Unit>
    </c2core:Officer>
  </c2core:message>
</ulex:StructuredPayload>
```

Figure 6: Notional C2 Core Portion

```
<ulex:StructuredPayload>
  <ulex:StructuredPayloadMetadata>
  <COIMessage>
    <Resource>
      <ulexlib:SameAsDigestReference ulexlib:nvref="P1"/>
      <Training course="Infantry Officer Basic Course" completed="true"/>
    </Resource>
  </COIMessage>
</ulex:StructuredPayload>
```

Figure 7: Notional COI Payload

The purpose of linking data across the layers is to support aggregation of message parts which have each been represented in the format expected by the target consumer. This design supports both message interoperability and flexibility.

6 The Capstone Subject Domain

The technical use cases employed in the Capstone explored the capability to design, produce, and consume a new information exchange using the message design described in this document.

The Capstone Team began with the CMD schema for providing realistic domain detail. The CMD is used by Air Operations to describe missions requested by Air Tasking Orders (ATOs) and sent to operational units tasked to fulfill the missions. The mission definition content includes a mission customer, tasked unit, location and target, along with the individual tasks making up the mission. In addition, required equipment configurations are described, as is rudimentary route information used by the tasked unit to create flight plans. Figure 8 is a high level CMD concept model provided by CMD documentation.[9]

The usefulness of CMD to the Capstone is that it represents real data requirements that are implemented in production environments today. The information exchange for this use case was built as if CMD did not exist – essentially building a new information exchange. CMD is useful in that it saves modeling time by providing definitions for the COI Payload and candidate requirements for a C2 Core model. Most importantly, it provides the complexity of a real world operational data sharing problem.

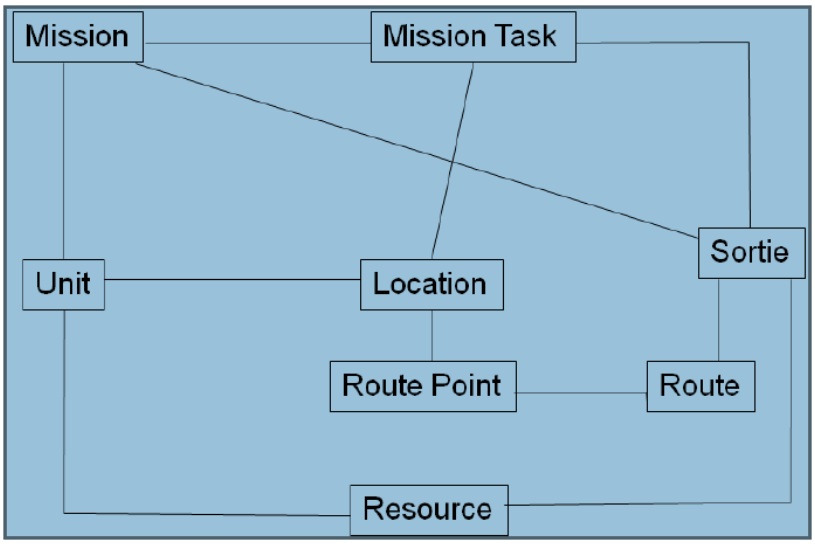

Figure 8: CMD Conceptual Model

The Capstone Team concentrated on one type of air operations mission, the "direct air attack" mission[10]. CMD covers many other air operation mission types, each having a different set of detailed information. The timeline for the Capstone effort only allowed two use cases to be examined, and the Team reasoned that the value proposition could be tested with a single mission type. The specifics of these use cases are detailed in sections 8.3.2.3 and 8.3.2.4. CMD data used for the experiment was generated from the TBMCS in the form of a CMD XML document. The Capstone experiment included transforming that document to the three-layer

[9] MITRE Technical Report (MTR080334) Common Mission Definition, Emre Gulbay, November 2008
[10] Specifically, CMD messages whose primary mission type code was "ATK".

message format described earlier. Then queries were posed against the content in this new format to determine whether it could support community needs.

After selecting and understanding the air operations mission domain, the next step was to create the C2 Core model needed to represent some of the mission information of interest to the broader C2 community.

7 The Capstone Notional C2 Core Model

As stated earlier, the Capstone effort started before the release of the C2 Core testable baseline. Therefore, the team fabricated a C2 Core data model with a high-level resemblance to the C2 Core testable baseline conceptual data model.

While the intent of the Capstone was not to produce high-quality content for later inclusion in the official C2 Core, it was considered important to create highly plausible content. This meant meeting two primary criteria:

- The data model should represent a real, joint C2 need.
- The concepts represented by the data model should be in active use by more than one COI.

Various C2 data models were reviewed for input into the notional C2 Core data model, in addition to CMD. More discussion of this survey of C2 content appears in Appendix A. For this experiment, the Capstone Team only needed to create enough C2 Core data components to represent some C2 level mission content. At the same time, the Capstone notional model had to be sufficient for creating a message exchange specification from beginning to end and to uncover the pros and cons of the UCore – C2 Core message architecture.

Figure 9 shows the C2 Core notional model developed for these purposes. It is composed of concepts which are popular across the C2 community and conceivably can be understood by consumers outside of air operations. The UCore Digest layer and C2 Payload layer were used to declare the information depicted here. This layer approach is discussed further in later sections.

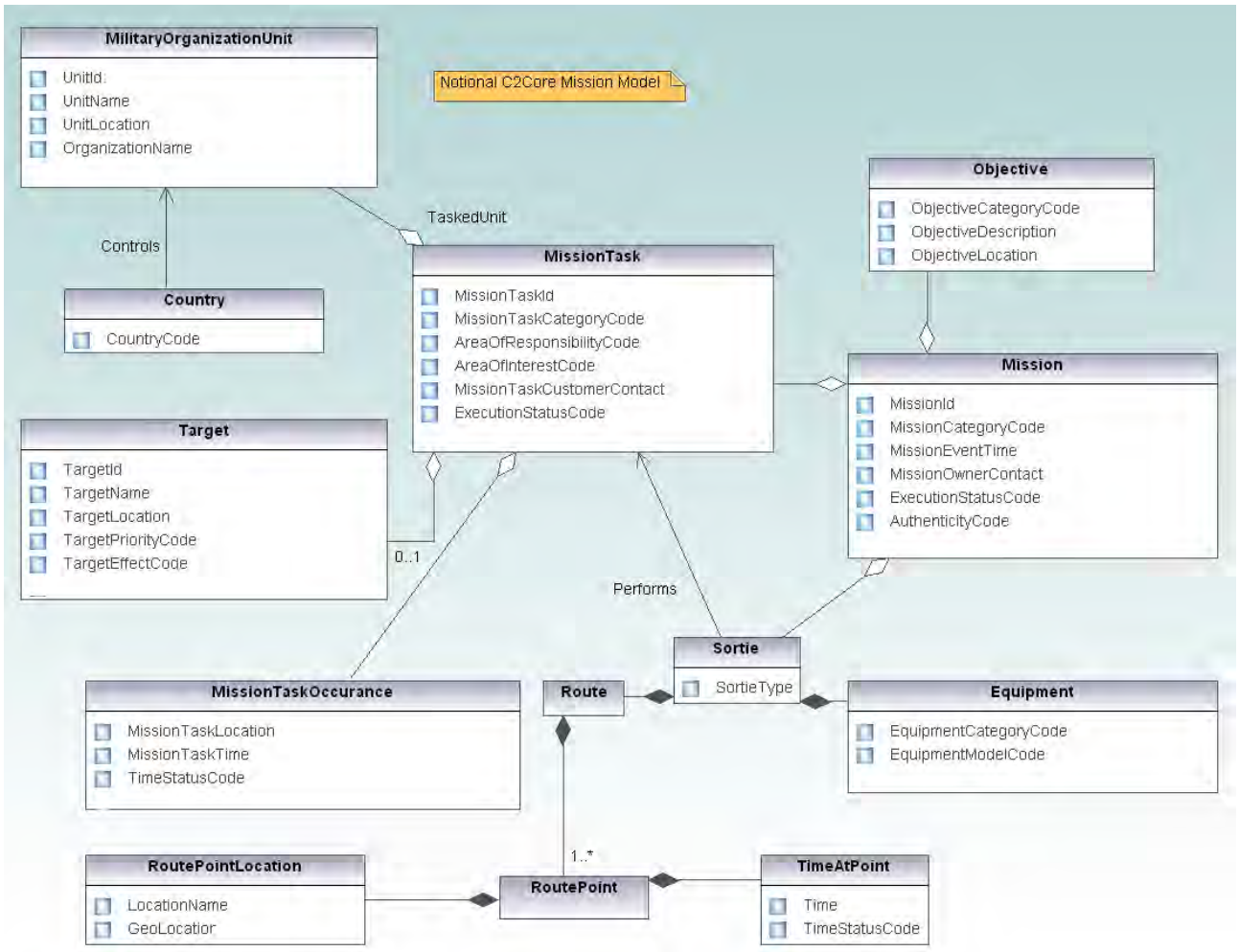

Figure 9: Notional C2 Core Model

Note that many of the direct attack mission details found in the CMD did not become part of the C2 Core model. For example, a detailed list of aircraft configuration items used to define a task resource is critical to the air operations COI, but may not be of interest to warrant inclusion in the C2 Core. As a result, that data appears in the COI specific payload layer using a COI-specific payload data model. Exactly which components are suitable for C2 Core and which are not is a judgment call that will be made over time by the C2 Core content working group; the important principle for the purpose of this experiment is that not everything belongs in the C2 Core, and that there will inevitably be the need for additional components in a COI-specific extension.

8 Applying C2 Core UCore Message Design to Air Mission Subject Domain

The remainder of this document describes the process and results of applying this C2 model and the left over Air Operations Direct Attack detail to the three layer message architecture described above.

8.1 Populating Message Layers

Because UCore Digest is generic and flexible enough to declare *where* and *when* and a little bit of *what* and *who* for any content in Air Operations Direct Attack messages, the message producer is left with choosing what to put in the UCore Digest and how to organize it[11]. To promote consistent messages, rules are needed for deciding which content goes in the Digest and which content goes elsewhere. One Capstone objective required that the rules take into account reducing or eliminating redundancy. This can be done by distributing message content across the message layers and omitting information represented in one layer from any of the other layers.

8.1.1 Message Content in UCore Digest Layer

The UCore specification does not include strong guidance for choosing which part of the message information should be represented in the UCore Digest or how it will be used. The Capstone Team experimented with several different guidelines for choosing Digest content.

The first approach was to create Digest objects for every piece of data from a CMD message that can possibly fit without misrepresenting the semantics of the information. The Capstone Team found though that the content became too vague to be meaningful, especially with respect to relationships between objects[12]. Indiscriminately populating the Digest with as many objects and relationships as possible did not seem to increase information value. This calls into question the utility of this rule considering the cost in message size and effort to populate and re-couple the information with data in the other message layers.

A second possible strategy was to populate the Digest with only those objects that can be located in time or space, because those concepts are well modeled in UCore. Even this rule puts data components into the Digest that do not appear to be useful to the UCore target consumer without C2 or Air Operations level detail. Given the overhead to tease apart and re-assemble content, information ought to merit elevation to the Digest.

The team ultimately decided to populate the Digest with what might be characterized as significant information that is potentially useful to an audience broader than C2. Choosing which content is significant and useful in its Digest format will have to be left to the judgment of the

[11] UCore provides many modeling options whose consequences are not always clear. For example, imagine modeling a UCore Digest that represents a company with three employees. One producer might model the company as being affiliated with a collection of employees. That collection in turn would be connected to each employee. Another producer might model the company as having direct relationships to each employee, with no intervening collection. UCore does not provide any guidance to indicate which is right or wrong; the consumer may require a priori knowledge to interpret such Digests.

[12] Frequently, complex relationships were collapsed into simple `AffiliatedWith` relationships. For example, the populated Digest might state that a `PlannedEvent` (in reality an attack) was `AffiliatedWith` an entity (in reality a target). While true, it was unclear who could make use of generalities like that.

message producer. Without a better understanding of the consumers of the UCore Digest, the choices that ensure value to the ad hoc consumer are not clear. On the other hand, unless some information is provided in a broadly understood way, the message is only understood by the narrower targeted audience.

UCore relationships were used to represent, at some level of generality, associations between the Digest objects. While avoiding redundancy was an important objective, there were cases in which a C2 Core relationship needed to be represented both in the Digest and the C2 layer. The UCore relationship is needed to make more sense out of the Digest for the Digest only consumers. But because the UCore relationship is not specific enough to represent all the meaning of the C2 Core association, the C2 layer will repeat and clarify the relationship.

Although UCore relationships can often be expected to be insufficient for the C2 consumer, there is one C2 association in our notional data model that is sufficiently represented by UCore. In CMD, a military organizational unit is associated with the country to which it reports. The UCore "Controls" relationship used between the Military Organizational Unit and the Country sufficiently represents that information so nothing further is declared in any of the other layers.

There is no standard way to show the C2 content selected for the UCore Digest other than looking at the associated XML instance document.[13] Figure 10 intends to illustrate distribution of C2 Core information across the UCore Digest and C2 layers. The C2 Core conceptual model shown earlier in Figure 9 is overlaid with UCore Digest objects showing how C2 content is mapped to the UCore Digest. Each UCore box is labeled with the corresponding Digest object name that is used to represent the information covered by the box. The objects and properties not covered by a UCore box go into the C2 layer.

Some property data types are noted in the figure next to the property names; simple string and integer data types are not noted. UCore time and location data types were used in the C2 layer if they met the needs of the C2 properties. It made good sense to reuse UCore types where possible to maximize the UCore and C2 Core value propositions. This kind of genuine reuse is labor saving and supports interoperability.

[13] The Capstone Team also developed rendering software for any UCore Digest XML instance. It presents all the UCore Digest objects as nodes in a graph where the connections are UCore relationships. At the time of writing, services created for the Capstone to render UCore Digests and UCore layers are accessible at http://bombadil.dyndns.org:8080/cgi-bin/ucore.pl

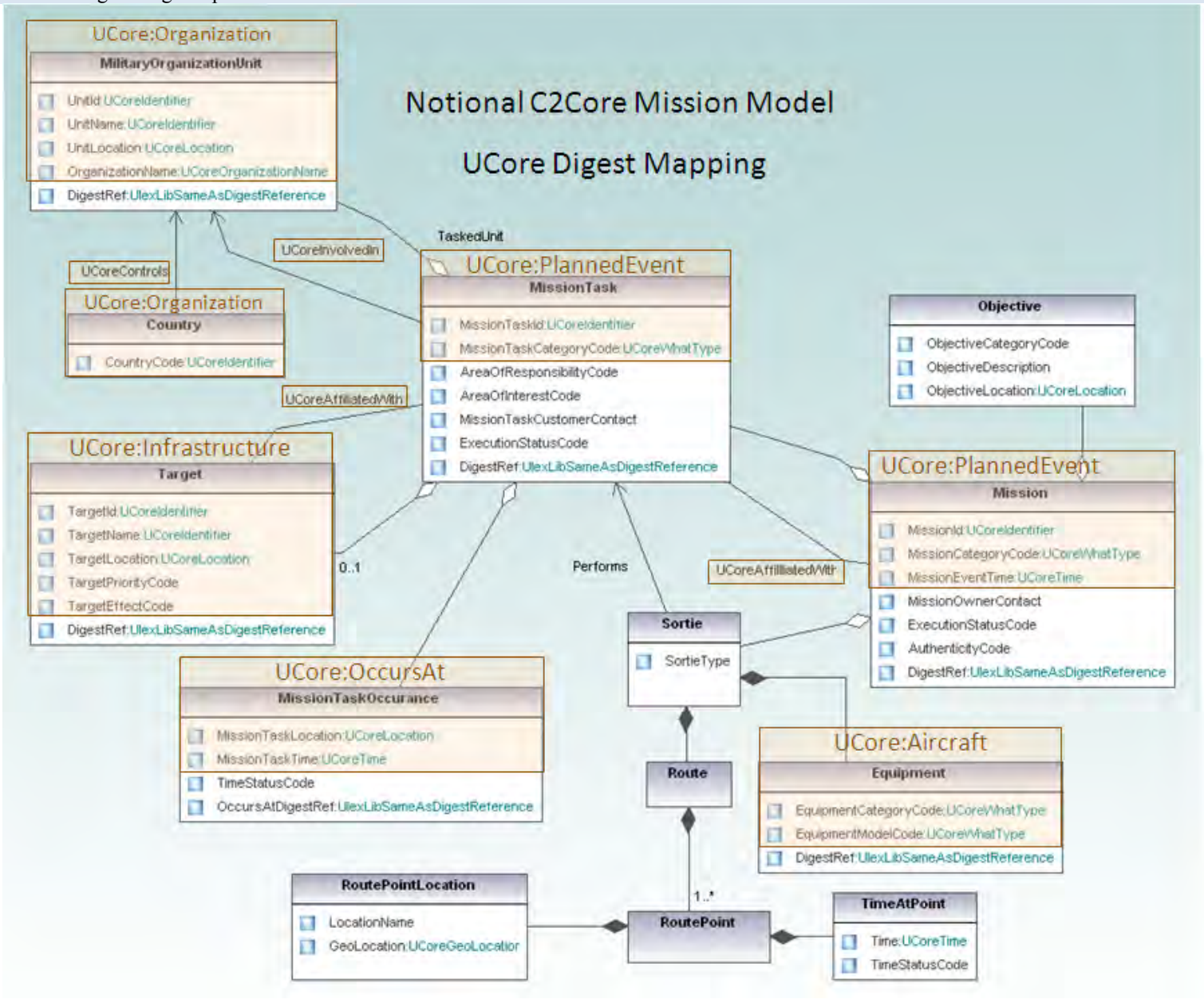

Figure 10: Mapping C2 Core to UCore Digest

The representation of C2 concepts in the UCore Digest is limited to high level UCore taxonomy assignments. Without an additional C2 taxonomy, the objects were too vague, and so we added C2 and Air Operations COI taxonomy codes. Figure 11 shows an XML snippet of the mission event in the UCore Digest revealing the taxonomy codes we chose. There is no clear choice for assigning classification terms beyond the term "Mission." It is not obvious when a classification term should instead be represented as a Mission property in the C2 layer.

```
<ucore:Digest>
  <!--
  ========= Mission Digest ==========
  -->
  <ucore:Event id="MzYyaTAxMDM5MHUwMDIsNDE1MDFnaGlq">
    <!-- Identifiers -->
    <ucore:Identifier ucore:code="MissionKey" ucore:label="Mission Key"
                      ucore:codespace="http://xml.dod.mil/aop/cmd/">MzYyaTAxMDM5MHUwMDIsNDE1MDFnaGlq</ucore:Identifier>
    <ucore:Identifier ucore:code="Number" ucore:label="TBMCS-ABP Mission Alias">ACM3/1501</ucore:Identifier>
    <ucore:Identifier ucore:code="Number" ucore:label="TBMCS-WWD Mission Alias">362i010390u009s</ucore:Identifier>
    <!-- What Codes -->
    <ucore:What ucore:code="PlannedEvent" ucore:codespace="http://ucore.gov/ucore/2.0/codespace/"/>
    <ucore:What ucore:code="Mission" ucore:codespace="http://us.jfcom.mil/c2core/codespace/"/>
    <ucore:What ucore:code="ManeuverToSecure" ucore:codespace="http://us.jfcom.mil/c2core/codespace/"/>
    <ucore:What ucore:code="ATK" ucore:codespace="http://xml.dod.mil/aop/cmd#ATOMissionTypeCodes/PrimaryMissionTypeCode"/>
    <ucore:What ucore:code="AIR" ucore:codespace="http://xml.dod.mil/aop/cmd/#MissionCategoryCode"/>
    <!-- This is represented in C2Core as Authenticity Property of Mission, so I should probably remove it from here?
    <ucore:What ucore:code="LIVE" ucore:codespace="http://xml.dod.mil/aop/cmd/#AuthenticityCode"/>
     -->
  </ucore:Event>
```

Figure 11: C2 Mission Mapped to UCore Planned Event

In the end, no matter what information is selected for the UCore Digest, those choices affect what needs to be represented in the rest of the message layers. The C2 Core Payload schema depends on the consistency of the message producer's choices for the Payload schema designer to be sure the produced schema is sufficient for representing the remainder of the information content. Although the choices made for the Digest dictate the design of the remaining layers, they are not recorded and enforced by any particular formalism in UCore.

The Capstone Team made expedient decisions that made sense in the context of the use case, but it is important to remember that other producers in other contexts would likely come to different conclusions. So part of the role of the C2 Core Specification is to provide strict guidance for populating the UCore Digest with C2 content. Without that, the Digest can be populated many different ways, leading to difficult and incoherent information exchanges and potential interoperability breaking points.

8.1.2 C2 Payload Layer

Once the information selection is made for the UCore Digest, the C2 Payload layer can be modeled in XML Schema Definition (XSD). The C2 Core Payload schema is needed by the message producer and the C2 and COI consumers for the C2 message layer. Therefore the model need only represent that information which is not going in the Digest layer.

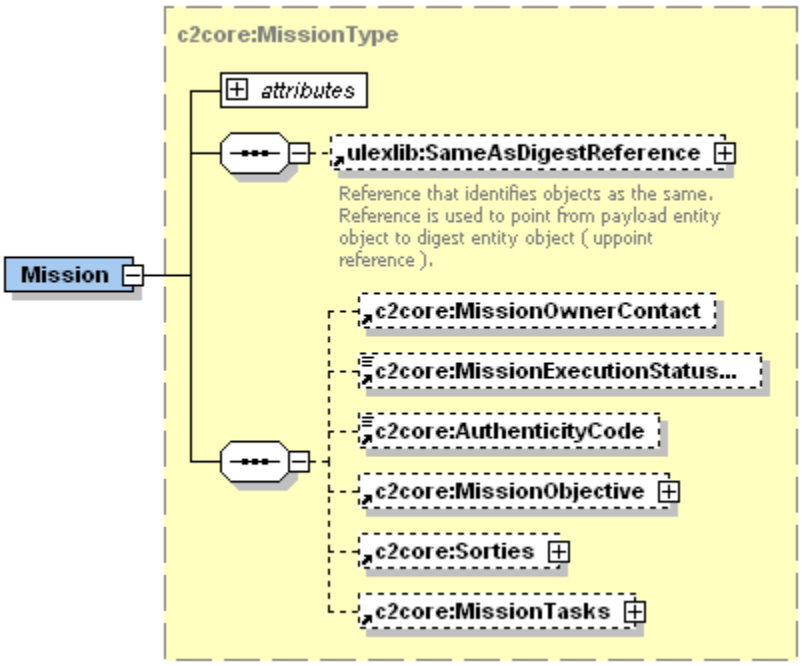

Figure 12: C2 Payload Mission Model

As stated in the C2 Core Development Concept document[14], the C2 Core model is supposed to be a set of building blocks or data components. It is not clear how those building blocks are to be assembled. For example, the concept document does not suggest whether an instance document will be a flat list of C2 objects and C2 relationships much like the UCore Digest or whether they will be hierarchically organized with a single root object. The Capstone's C2 Payload XSD model shown in Figure 12 exploits the hierarchical nature of XML as opposed to the UCore Digest's relatively flat structure. This seemed to be the typical XML modeling approach and is most expedient given the formalism in which the model is built.[15] So wherever there is a composition relationship in the C2 Core notional Unified Modeling Language (UML) model depicted in Figure 9, the schema uses containment to represent the association.

The XML code snippet in Figure 13 provides some idea of an instance of the C2 message layer. It shows part of a C2 layer Mission instance. There is a pointer back to the Digest planned event data component, a few properties indicating that the mission has been tasked (TSK) and is live rather than simulated. The mission objective is a child component of mission. The rest of the mission instance is not in this figure.

[14] Section 7.1.2, "C2 Core specifications will be designed with the idea of reusable components in mind"
[15] There is more on this topic in Section 9, Technical Findings.


```
<c2c:Mission ulexlib:id="Mission_1_p">
  <ulexlib:SameAsDigestReference ulexlib:nvref="MzYyaTAxMDM5MHUwMDIsNDE1MDFnaGlq"/>

  <c2c:MissionExecutionStatusCode>TSK</c2c:MissionExecutionStatusCode>
  <c2c:AuthenticityCode>LIVE</c2c:AuthenticityCode>
  <!-- ======== C2 Core Mission Objective ======== -->
  <c2c:MissionObjective ulexlib:id="MissionObjective_1_p">
    <c2c:MissionObjectiveCategoryCode>DestroyTarget</c2c:MissionObjectiveCategoryCode>
    <c2c:MissionOjectiveDescription>
      Increase security in the region by distroying suspected enemy control center.
    </c2c:MissionOjectiveDescription>
  </c2c:MissionObjective>
```

Figure 13: C2 Layer Mission XML Element

The C2 message layer schema reuses some UCore data types rather than re-creating those types as shown by the XML message snippet in Figure 14. The C2 route reuses the UCore GeoLocation data type for latitude and longitude data and UCore TimeInstance data type for the time at route point data.

```
<c2c:Sorties>
  <c2c:Sortie ulexlib:id="Sortie_1_p">
    <c2c:Equipment ulexlib:id="Equipment_1_p">
      <ulexlib:SameAsDigestReference ulexlib:nvref="Aircraft_1"/>
    </c2c:Equipment>
    <c2c:Route ulexlib:id="Route_1_p">
      <c2c:RoutePoints ulexlib:id="RoutePoints_1_p">
        <c2c:RoutePoint ulexlib:id="RoutePoint_1_1_p">
          <c2c:RoutePointLocation ulexlib:id="RoutePointLocation_1_1_p">
            <c2c:LocationName>BOUAKE</c2c:LocationName>
            <ucore:GeoLocation>
              <ucore:Point>
                <gml:Point gml:id="gmlRoute_1_Point_1_Location"
                           srsName="http://metadata.dod.mil/mdr/ns/GSIP/crs/WGS84E_2D">
                  <gml:pos>7.738888888888888 -5.073611111111111</gml:pos>
                </gml:Point>
              </ucore:Point>
            </ucore:GeoLocation>
          </c2c:RoutePointLocation>
          <c2c:TimeAtPoint>
            <c2c:TimeStatusCode>Planned</c2c:TimeStatusCode>
            <ucore:TimeInstant>
              <ucore:Value>2009-02-26T07:38:00.000Z</ucore:Value>
            </ucore:TimeInstant>
          </c2c:TimeAtPoint>
        </c2c:RoutePoint>
      </c2c:RoutePoints>
    </c2c:Route>
```

Figure 14: C2 Route Reusing UCore Data Types

The example UCore Digest declares a few relationships between objects. The XML instance snippet in Figure 15 shows an "OccursAt" relationship stating that a mission task occurs at a certain location and at a certain time. The CMD schema requires qualifying this time as "Planned", "Scheduled", or "Actual." The "OccursAt" relationship has no qualifier property. To

remedy this, the C2 message layer adds this qualifier to the time data with an element called "Occurance" and a property called "TimeStatusCode." The "Occurance" element has a link property used to link back to the Digest "OccursAt" data component. This is shown in Figure 16.

```
<ucore:OccursAt id="MissionTask_1_OccursAt_1">
  <!-- Note: CMD distinguishes scheduled, planned or actual time.  This is declared in the C2 Payload. -->
  <ucore:Time>
    <ucore:TimeInterval>
      <ucore:StartTime>
        <ucore:Value>2009-02-26T10:22:00.000Z</ucore:Value>
      </ucore:StartTime>
      <ucore:EndTime>
        <ucore:Value>2009-02-26T12:15:00.000Z</ucore:Value>
      </ucore:EndTime>
    </ucore:TimeInterval>
  </ucore:Time>
  <ucore:EventRef ref="MissionTask_1"/>
  <ucore:LocationRef ref="MissionTaskLocation_1"/>
</ucore:OccursAt>
```

Figure 15: UCore Digest `Occurs` Relationship

```
<c2c:Occurance ulexlib:id="Occurance_1_p">
  <ulexlib:SameAsDigestReference ulexlib:nvref="MissionTask_1_OccursAt_1"/>
  <c2c:TimeStatusCode>Planned</c2c:TimeStatusCode>
</c2c:Occurance>
```

Figure 16: C2 Layer Task `Occurs` Qualifier

This distribution of information is awkward, but it does prevent a UCore Digest consumer from encountering unexpected information and structure. There are trade-offs associated with an interoperable layered architecture. In addition to some awkward modeling, one might question whether the time qualifier is critical and therefore wonder if omitting it from the UCore layer is risky. This concern is relevant to discussions about the value and expected use of the UCore Digest to UCore-only consumers.

8.1.3 Air Operations COI Layer

The Air Operations (AO) COI specific information deemed irrelevant for the UCore and C2 layers includes the administrative state of the mission, mission identifiers used by AO systems, specifics about sorties such as aircraft call signs assigned for a route, references to specific ATOs, and so on.

The AO COI layer schema in Figure 17 shows the top level set of XML tags defined in the AO COI XSD. Most of the tags in this example have the same name (but different namespace) as those in the C2 layer, but the contents underneath are those things not declared in the UCore Digest and C2 layer. Having a similar containment structure also happens to imply relationships that are also implied in the C2 containment hierarchy. This duplication is necessary but possibly problematic for the consumer software having to deal with duplicative associations, implied or explicit.

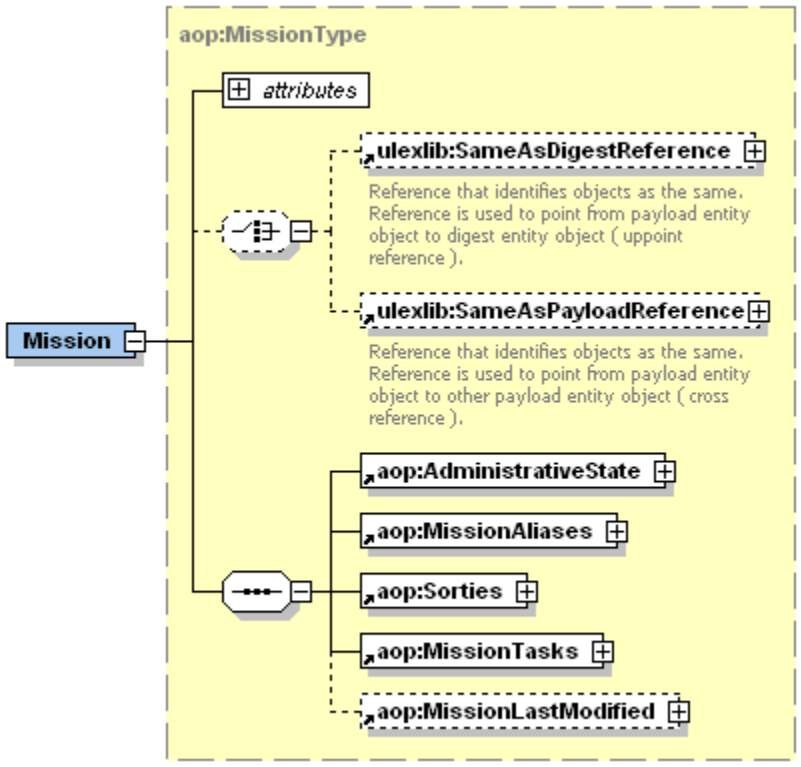

Figure 17: Air Operations COI Payload Schema

As with the C2 layer data components, AO COI data components may link back to the UCore Digest. Alternatively they may link to the C2 layer. The XML instance snippet in Figure 18 shows that the AO Aircraft Sortie links back to the C2 layer Sortie and the AO Aircraft links back to the UCore Digest layer Aircraft Entity.

```
<aop:Sorties ulexlib:id="Sorties_1_p2">
  <aop:AircraftSortie ulexlib:id="Sortie_1_p2">
    <ulexlib:SameAsPayloadReference ulexlib:nvref="Sortie_1_p" ulexlib:pnvref="C2CorePayload"/>
    <aop:Aircraft ulexlib:id="Aircraft_1_p2">
      <ulexlib:SameAsDigestReference ulexlib:nvref="Aircraft_1"/>
      <aop:PlannedPrimaryConfiguration>BEST</aop:PlannedPrimaryConfiguration>
    </aop:Aircraft>
```

Figure 18: AO COI Layer XML Snippet

There is additional information in the AO COI layer, for example, the air operations specific route information. Figure 19 shows parts of the AO route point data and the route point's link back to the C2 layer route point. Call signs and frequencies are considered air operations specific.


```
<!-- A0 COI Specific Route Point -->
<aop:RoutePoint ulexlib:id="RoutePoint_1_1_p2">
   <ulexlib:SameAsPayloadReference ulexlib:nvref="RoutePoint_1_p" ulexlib:pnvref="C2CorePayload"/>
   <aop:RoutePointTypeCode>TKF</aop:RoutePointTypeCode>
   <!--
      ===== Additional Location Information =====
   -->
   <aop:RoutePointLocation ulexlib:id="RoutePointLocation_1_1_p2">
   <!--
      ==== Additional Route Point Information ====
   -->
   <aop:Callsigns>
      <aop:Callsign>
         <aop:CallWord>COWBOY</aop:CallWord>
         <aop:CallNumber>01</aop:CallNumber>
         <aop:AOR>DAVIS MONTHAN</aop:AOR>
         <aop:ABPId>ACM3</aop:ABPId>
         <aop:ATOMissionNumber>1501</aop:ATOMissionNumber>
         <aop:IFFSIFs>
      </aop:Callsign>
   </aop:Callsigns>
</aop:RoutePoint>
```

Figure 19: AO COI Specific Route XML

The following two subsections discuss message instance production and consumption issues.

8.2 Inter-Payload Pointers and Data Referencing

To minimize redundancy in messages, pointers must be used extensively. The nature and structure of these pointers is a substantial complicating factor in how multi-layered messages are produced and consumed. There is a straightforward trade-off at work here; to reap the benefits of component reusability and redundancy minimization, the complexity of these pointers must be addressed.

8.2.1 Chaining Pointers

With three layers in a message though, a second question comes up. Should all subsequent layers point back to the UCore Digest? Or should each layer point to the layer immediately above it? These chaining options – referred to as “Digest-only” and “chaining” pointers, respectively – are pictorially illustrated in Figure 20 and Figure 21.

The Capstone Team chose to use chaining pointers. For one thing, chaining pointers provide the ability to link the next level of detail upward: the COI Payload links to the C2 Core Payload, and the C2 Core Payload links to the UCore Digest. Another reason for choosing chaining is that it may not be possible to represent everything in a detailed message with a corresponding entry in the UCore Digest. If linkage to the Digest is required, and there were no Digest entry for a particular entity, then it would not be possible to represent linkage between the COI Payload and the C2 Core Payload to connect two objects describing the same thing.

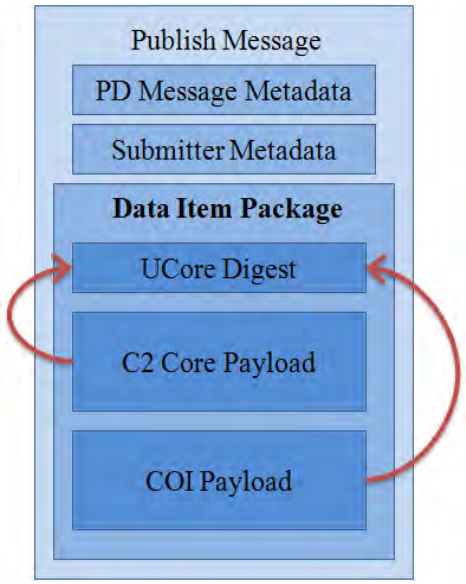

Figure 20: Digest-only pointers

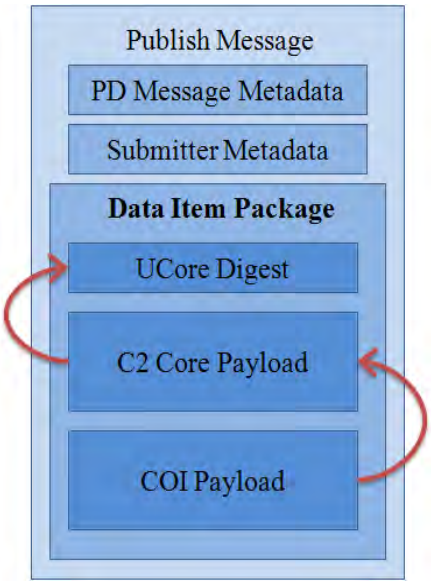

Figure 21: Chaining pointers

In the experimental instance messages there was only one exception to this rule: MissionTaskLocation elements in the COI layer pointed directly to the UCore Digest, and not to the intervening C2 Core Payload. The reason for this was that the UCore Digest had already represented most of what was needed, to the point that there were no additional value-added C2-specific data elements that the C2 Core Payload could represent. As a result, MissionTaskLocation elements did not occur in the C2 Core Payload, and the COI layer was forced to link back to the UCore Digest.

This raises an important point about pointer strategies. In many exchanges, it may not be possible to adopt one strategy or the other. Based on modeling requirements, multiple strategies may be needed concurrently and/or at different times, which complicate message production and consumption.

8.2.2 Pointer Validity

One of the important considerations when using pointers is that the referrers and referents be of the same type. For example, if a message uses a `ulexlib:SameAsDigestReference` pointer from a C2 Core `AttackMission` element, the referent must be a UCore "Event" of type `PlannedEvent`. Such constraints placed on referrers and referents help ensure the semantic consistency of the message. Should a message have a "same as" link from a type of organization in the C2 Core layer to a UCore Person, the interpretation of the message would be undefined. The testable baseline for C2 Core includes a discussion of the need for this kind of "binding validation." The testable baseline additionally includes a sample binding document to aid this validation, but leaves undefined exactly how or when this validation will be performed.

Although this is thought to be an important constraint on message design, due to the scope of the C2 Core Capstone such constraints were not implemented. Such constraints are likely only able to be expressed in terms of advanced validation languages, such as Schematron[16]. These

[16] Schematron (http://www.schematron.com/) is a language separate from XML schema that allows for assertions to be made against patterns in XML instance documents.

observations point to the fact that specifying sufficient rules to make C2 Core messages meaningful and useful will likely fall outside of the bounds of what XML schema is capable of expressing.

The primary advantage of this approach is that software processing the message can build a complete picture of the data object John Doe while avoiding redundancy in the message. The major disadvantage is that this requires refactoring of the COI schema to remove extra data (like name, sex, unit assignment) that would otherwise be redundant. As a result, this approach would appear more suitable for implementing new information exchanges with C2 Core, such as in situations where there is not an already existing schema, or where it is only in a developmental stage. Indeed, this is in line with the C2 Core value proposition, which focuses on the implementation of new exchanges; C2 Core does not make any specific claim with respect to the ease or value of refactoring/changing existing exchanges.

8.2.3 Pointer Processing

One other potential drawback may apply to this extensively interlinked message structure. In cases where ULEX messages grow quite large, it may be necessary to alter the message processing approach to resolve all "same as" pointers properly. Because the application may not know ahead of time where all the pointers resolve, doing such processing in a single pass may require a message that is small enough to fit into main memory. Cases in which a message is too large for main memory may require either multi-pass processing or intermediate storage (e.g., in a relational database), complicating message processing for consumers.

8.3 Message Implementation

8.3.1 Message Production

8.3.1.1 Modeling and Content Issues

Sample CMD messages were transformed into three-layer ULEX/C2 Core/COI messages through the use of an Extensible Stylesheet Language Transformations (XSLT) program. Message production proceeded by setting up a default ULEX wrapper for the target message, and by processing most elements in the source CMD message three times: the first pass generated the corresponding UCore Digest elements; the second pass generated corresponding C2 Core elements; and finally the third pass output the remaining message details into the COI layer.

Translating information from the perspective of the COI into that of the C2 Core is not always straightforward. "Widening conversions" such as needing to represent an F15 aircraft as a more general item of "equipment" are common and relatively straightforward. But frequently the COI representation of relationships differs from those that will be found in the C2 Core layer. This difference necessitates very idiosyncratic processing of some structures to output the corresponding C2 Core structures.

For example, while it is generally true that each element in the document was processed three times (i.e., once for each layer), some elements in the source documents still will not contain sufficient information for their C2 Core counterparts owing to differing representations and modeling choices. This in turn requires "jumping around" in the source document to assemble enough information to output the correct C2 Core structure. It is unlikely that a straight event-

driven Simple API for XML (SAX) parsing of the document would be sufficient to translate a COI message into a three layer structure.

8.3.1.2 Implementation Issues

Generating the requisite pointer structure for three layer messages was very challenging. Much of the testing and debugging time that went into the message producer was spent validating pointers by hand and writing custom scripts to do the same automatically. As described in the section on pointing strategy (section 8.2), while the general approach was always for one layer to point to the layer immediately above it, there were exceptions and it was a challenge to validate that these were done properly.

Another difficultly dealt with message validation. The ULEX specification is written in such a way that payloads are not required to be valid. As a result, XML software did not check the validity of the C2 Core and COI layers even when schemas were independently specified for those layers. During testing, a very small change in the ULEX specification was required simply to force the XML software to validate all the layers in one pass, to confirm that message production was proceeding correctly. [17]

8.3.2 Message Consumption

8.3.2.1 Re-Assembling Message Content

The primary task of message consumption is to re-assemble the three layers of information into a single coherent structure with a complete account of a necessary data object. While the information is broken up into those three layers to provide for "layered interoperability" benefits in the form of selective layer consumption, any consumer who understands more than one layer will have to re-assemble the layers into a comprehensive picture.

Table 1 provides an example of how this looks for the "Mission" data object, which occurs in all three layers of the message. (A full technical example of a message can be found in the appendices of this document.) Each layer provides additional details about the same Mission object. The task of the consumer is to create a single Application Programming Interface (API) object that encapsulates all this detail.

In the interoperability literature, there is considerable discussion about "understanding." It is assumed that if a consumer is aware of a standardized schema, the consumer can "understand" data formatted according to that schema. When processing messages, however, a more detailed definition of "understanding" is needed to account for the gap between strategic discussions of data management, and the programming that is necessary to implement actual information systems.

To the programmer, "understanding" the message usually means either having a standardized and documented API which presents the information to the programmer in a convenient way, or a detailed enough set of specifications to write that API. In the case of multi-layered messages, having the schemas is insufficient to "understand" the message. The programmer needs additional detail, such as modeling decisions and the pointing strategy employed. While processing XML documents, the programmer must understand where the pointers will be, where

[17] The ULEX Payload has as its main property a wild card meant for any payload content. In UCore 2.0, the wild card element means "skip content validation even if a schema for the content is provided." The Capstone Team changed the wild card property to mean "if a schema for the content is provided, use it for content validation; otherwise skip content validation." This was a one word change in the ULEX schema.

they will point, and why; the programmer also needs to understand the subtle differences between modeling decisions made at the C2 Core and COI layers, so that the information can be reassembled in a meaningful way.

Table 1: Mission Information Distributed Across Layers

UCore Digest	• A `PlannedEvent` • Identified by `MissionKey`, `TBMCS ABP Mission Alias`, and `TBMCS-WWID Mission Alias` • Attack mission (external taxonomy reference) • Air mission (external taxonomy reference) • Live mission, i.e. not a test (external taxonomy reference)
C2 Core Payload	• Mission Owner contact • Execution status code • Authenticity code • Objective (destroy target)
COI Payload	• Administrative state

An API will likely be necessary due to the complexity of reconciling the links in the XML and assembling the overall picture. Additionally, since almost all consumers will need to do this task, it makes sense to do it once (correctly) in a standard API, rather than requiring that every consumer independently writes code to do the same assembly process.

Once the data is "understood" via this API, that overall model has to be filtered through the context of what the programmer is trying to accomplish. The API which he or she uses to consume the message must be capable of adapting to the information need the programmer is trying to satisfy.

To explore the complexity of consuming these messages, the Capstone Team constructed two distinct use cases: one case for a C2 Core only user who has no understanding of the COI Payload; and a second case for a COI participant.

8.3.2.2 Creating a Layered Message API

To process multi-layered messages, most of the work went into writing a Java API[18] which a consumer could use to access the content irrespective of layer. The consumption program used

[18] Programming code for this API is available through MITRE's internal source forge website, under the project name "C2 Core Capstone FY09" -- http://developer.mitre.org/scm/?group_id=1228. As discussed earlier, this code primarily focuses on re-assembly of the message, without providing many specifics on what to do with the data. Sample programs that use this API to implement the test cases are included in that project.

Java, Eclipse, and XMLBeans[19] to process raw XML messages and to provide the data in accordance with a Java API that mimicked the three source schemas (i.e., UCore Digest, C2 Core, and the Air Operations [AOP] schema which represented COI Payload contents).

XMLBeans allowed the source schemas to be "compiled" into native Java APIs, which greatly accelerated the processing of raw XML messages. Unfortunately, XMLBeans API objects are not capable of processing or resolving "pointing" structures, such as the ULEX "same as" pointers discussed earlier. What the XMLBeans API does do is to present the three-layer message as a set of object APIs that conform to the relevant schemas. This means that if the application programmer is familiar with the structure of the various schemas (ULEX, UCore, C2 Core, AOP, etc.) then the programmer will be able to write the next layer of API.

This additional layer of API was placed over the XMLBeans API objects to resolve the pointer structure, and to produce a single object which can be used to access necessary data. The result was that a consuming application could process an incoming XML message and receive a single "LayeredMessage" object, which could be interrogated for any aspect of the message (such as the latitude/longitude of a given mission route point).

This additional API layer was required to abstract the considerable complexity of message processing away from the code which implements the consumer's business logic. There is quite a bit of complication that goes into handling the technical artifacts of XML that has nothing to do with the high-level objective of the consumer. Such complication should be "hidden" from the consumers in an API which provides the necessary functionality without requiring detailed knowledge of the message.

This additional API layer indeed did make the creation of consumption use cases much easier, but it also implied a "joined meta-model" of the information in the message. By connecting all three layers into a single picture, the API made assumptions about how that single picture would be structured, i.e., what the elements would mean, and how relationships would be specified. While this joined model was suitable for air operations planning, it would not be suitable for other consumption use cases. Each message design using a different COI Payload schema will need its own joined meta-model for the whole message. Whether the meta-model does or does not exist explicitly, it has been conceived by the message designer to understand how the message content will be distributed across layers and which content remains for the COI Payload schema to represent.

The fact that different messages require different joined meta-models raises questions about whether it is possible to develop a single C2 Core consumption API that can be used across widely varying contexts. Even if it is possible to develop a single API, this "joined model" would need to be made explicit, and deemed "acceptable enough" by the C2 Core community for them to use it.

8.3.2.3 Message Consumption Use Case #1: C2 Core Consumer

To assess whether an unanticipated user could derive value from the UCore and C2 Core layers alone, the Capstone Team devised a use case for exactly such a user. This use case involves a notional user who needs to consume air operations data from the AO COI to coordinate missions in a given geographical area, and to prevent conflicts and friendly fire.

[19] http://xmlbeans.apache.org/

The task in this use case was to process a three-layered message and to determine which air missions occurred within a given space/time box. The test program was given the following inputs:

- A space/time box consisting of a bounded geographic box (i.e., two latitude and longitude points representing the upper left and bottom right of the box) and a time interval. An example of such a space/time box might be "upper left corner (10.07, -8.14), bottom right corner (4.993, 1.06)[20] between 8:00AM and 10:00PM on October 1, 2006"
- A series of six layered messages, each representing a direct attack air mission.

With this information, a notional, unanticipated user could determine whether there were planned direct attack air missions in a given area and time. Such information could help a commander avoid placing blue forces in that same space/time box, and/or help the commander coordinate with the units tasked with the direct attacks.

The test program was successful in producing the desired output, which was an indication of which of the sample layered messages represented missions that occurred within the space/time box of interest. Additionally, due to the program's ability to understand C2 Core `RoutePoint` objects, the program could output exactly which route points within which missions occurred in the desired range.

8.3.2.4 Message Consumption Use Case #2: Air Operations COI Consumer

COI participants typically use air mission documents for planning and execution purposes. The test messages produced for the experiment were comprised of data that came out of the high-level TBMCS planning system, destined for subordinate organizations who would do more detailed-level planning. For example, the test messages might indicate only the location of the target, and that the mission should fly from the source base, to the target, and subsequently return.

The organization tasked with executing the mission is required to do the next level of planning; for example, which specific aircraft (and tail number) will fly the mission? Which specific route will be taken to avoid en route obstacles, threats, and observed weather? Which configuration and armaments will the aircraft carry?

The COI consumer of these messages needs to understand all aspects of the entire message. For this use case, the Capstone Team focused on a set of data objects distributed across all three layers. The intent was to gather enough information from the message for the COI participant to determine whether they had the resources necessary to carry out the mission. This test case included consuming information about:

- The specified mission
- The tasked organization
- The aircraft assigned
- The routes involved, including all required route points
- The planned configuration of the aircraft tasked to fly the mission
- The target location, and desired effect (e.g., "destroy target")

[20] This geographic bounding box refers to most of the Ivory Coast, and portions of Ghana in western Africa

The test program was successful in producing the desired output, which was a complete account of all these data objects, suitable for input to a detailed mission planning system.

8.3.2.5 High-Level Outcome of Consumption Use Cases

The experiment determined that the three-layer messages produced had sufficient detail to be useful for a hypothetical C2 Core only consumer. Additionally, the messages preserved all necessary information from the source CMD instances, indicating that they are sufficiently detailed and correctly modeled to be useful to the original COI consumers.

While successful, message consumption involved a good deal of programming that was focused on overcoming the technical challenges of the layered format. Before code could be written that performed the use case, an API was needed to allow the programmer to access the three-layered message in a convenient way that abstracted away the complications of the underlying XML. Until a standardized API is available to assist this task, each consumer is likely to re-implement this type of software, tailored to his specific needs.

9 Technical Findings

Based on our experience using the proposed UCore - C2 Core message architecture for the Air Operations Direct Attack message, the Capstone Team discovered some issues needing further attention. First, using UCore for C2 Core messaging standardization is not a simple task. Second, UCore and C2 Core need to establish additional design specification details and support tools before their respective value propositions can be realized. The following paragraphs discuss the team's technical findings in more detail.

9.1 Data Modeling Rules for Layered Architecture is Missing

Differences in the modeling paradigm between the UCore Digest and the Payloads cause difficulty in declaring information consistently and cleanly across the message layers. There is no rule set yet for distributing information across layers. Additionally, the UCore Digest model by itself is intentionally flexible, leaving many decisions up to the UCore message producer. Without guidance and business rule enforcement for the C2 community, this flexibility will lead to inconsistent content representation.

9.1.1 Associating Objects

Structurally, the UCore Digest is flat and contains only explicit relationships; whereas the C2 Core Model is hierarchical with implicit relationships expressed via containment. In addition, the level of relationship abstraction is different between the Digest and the Payload: UCore Digest relationships are mostly vague – as in `AffiliatedWith` – whereas C2 Core relationships tend to be either specific – such as "`PerformsTask`" – or implied through containment. The composition or aggregation relationships implicitly declared by containment in C2 Core seem more intentionally meaningful than they would be without the domain context in which they are used.

The C2 Core or COI layers will sometimes need to "extend" the vague Digest layer relationships, either by giving the relationship a more meaningful name, by adding properties, or by association via proximity in the parent-child hierarchy. To "extend" the Digest relationship in the Payload requires expressing it as a separate object; i.e., it cannot be expressed implicitly through containment.

In this experiment, the Capstone Team expressed many associations using containment. For example, `MissionTask` executes a `Mission`, which in the Digest is expressed as an `AffiliatedWith` object and in the C2 Payload as a parent-child containment association. There is no place to attach the `SameAsDigest` pointer property in the implied relationship to indicate it is the same as or an extension of the `AffiliatedWith` relation.

There was one situation in which the UCore Digest "`OccursAt`" relationship needed to be qualified in the C2 layer to indicate whether the time interval was planned, actual, or scheduled (see section 10.1.2). An explicit C2 relationship object was created with a same-as link back to the Digest relationship and a Time Status Code property.

This relationship modeling dichotomy puts the C2 Core schema designer in a dilemma. The schema designer could anticipate the need to link and extend UCore relationships by creating separate C2 relationship objects and eliminating the use of containment. It seems appropriate to

use the same modeling paradigm for the C2 layer as is used by the UCore layer on which it is built. Doing so means that relationships can be disambiguated with the same-as pointers.

On the other hand, use of explicit relationships will be awkward because XML is a hierarchical model and containment is an expedient way to represent composition-like relationships. XSD can easily declare whether the association is optional or mandatory and other cardinality constraints. Techniques such as XPath[21] for locating data in an XML document rely on containment to disambiguate data.

9.1.2 Classifying Objects

Class information in the UCore Digest is not declared using schema structure as is typically done with XSD. UCore schema components are generic and unclassified until a message instance is created, thereby giving UCore a lot of flexibility. The message instance contains taxonomy codes known as "what-codes;" these are assigned to the generic objects[22] providing classification at "run time." On the other hand, C2 Core and COIs rely on schemas (i.e., tag names declared as simple or complex data types) to ascribe meaning and structure to data. Although these are different approaches to classification, it is still necessary that an object in the Payload that is said to be the same as an object in the Digest agree semantically. This means that the taxonomy codes assigned in the Digest should agree with the element names used in the Payload for objects that are said to represent parts of the same real world object.

The C2 Core testable baseline uses a binding file to specify the correlation between the Digest object class and a Payload object class. For this to work, the binding specification must be able to disambiguate "like" Digest objects. For example, it must be able to distinguish one UCore Planned Event representing Mission from another UCore Planned Event representing Mission Task. This can be done only by assigning additional what-codes from a C2 taxonomy and including those as part of the binding specification. While this might take care of the one-to-one mapping disambiguation, it does not take care of the situation in which a Digest object is sufficient for a given C2 concept and therefore needs no additional representation in the C2 Payload. The requirement for the Digest object to exist cannot be expressed with a mapping. This problem is discussed below in section 9.1.5.

9.1.3 Populating the Digest

There is little guidance regarding which part of the message content should be declared in the UCore Digest to create a valuable Digest. The Capstone Team grappled with this as discussed previously. Should only "important" content be expressed? Or only content with time and space values? Or only content meeting some other criteria? When should relationships be expressed and when is doing so merely adding noise because of the abstract nature of many UCore relationships? Could putting some data in the Digest adversely affect comprehension, as in the use case where mission task time could not be qualified as planned versus actual?

9.1.4 Extending Taxonomies

UCore's "what code" design is underspecified for promoting consistent message production within a community. Although UCore comes with basic what-code taxonomies, C2 Core and

[21] XPath is a query language for selecting nodes from an XML document and that also may be used to compute values from XML content. For more information, see http://www.w3.org/TR/xpath

[22] Oddly, UCore does not have this capability for location or relationship objects.

COI groups need additional taxonomies to implement useful Digest instances. Creating and using community taxonomies in an interoperable fashion requires more specification. Here are some of the questions to consider:

- Are what-code classification terms only those that declare something intrinsic about the object or can they be any sort of keywords that help in filtering and discovering relevant data?
- Does the taxonomy need to be an "is-a" hierarchy or can it be any sort of category and sub-category hierarchy?
- What is the difference between a taxonomic term used as a Digest what-code and other category codes or status codes that can be declared as object property values in the Payload? For example, in this experiment should the Mission Category Code appear as a what-code in the C2 Core or COI codespace? Or should it appear as a property of the Mission in the C2 Core or COI Payload layers? Or both?
- How will changes in the UCore taxonomy affect both the C2 Core taxonomies and the C2 Core Schema?
- Can / should there be enforceable business rules that apply to the use of C2 Core codespace what-codes in the Digest? So for example, if the C2 message producer declares a mission object in the UCore Digest, then should there be a way to require and validate the use of a Mission Category Code in the what-code list?

9.1.5 Mapping Layered Data to One Joined Data Model

Once Digest content selection has been decided, those rules will need to be enforced by means external to those inherent in XSD schemas. As it is, the dependency between the C2 layer schema and the choices made by a message producer for a particular message instance is intangible. For example, the UCore "Controls" relationship in the example Digest is used for the relationship between a military unit and the country to which it belongs. An organization object is used to represent the country. No further specification for these two objects is needed in the Payload as long as the message producer populates the Digest with these objects. If one can depend on this information being present in the Digest then it need not be modeled in the C2 Core layer schema. Expressing this requirement is outside the capability of an XSD schema.

A method is needed for expressing the C2 logical model and how its representation is distributed across the UCore layer and C2 Core layer. Once expressed, a tool is needed to ensure instances have subscribed to that representation plan.

9.1.6 Translating System Level Data to Higher Levels of Abstraction

During this Capstone work, the team discussed the issue of abstraction quite a bit. C2 Core is likely to be an abstraction of many specific C2 COIs, i.e., providing reusable data components common to multiple COIs. Mapping from the specifics in CMD to the abstractions in C2 Core is an issue and subject to many subtle reasons and judgment calls on the part of the message producer.

For example, CMD declares a specific airplane type and its configuration, but never specifically declares a qualitative capability for that piece of equipment. In C2 Core, the model might declare something explicitly about those capability implications such as "long range aerial bombing

support," as opposed to declaring an F15 aircraft with extra large fuel tank and ordinance carriers.

Translating back and forth among abstractions and implications and COI specifics is an exercise left to the C2 Core user.

9.2 Uncertain Value in UCore Digest Layer

Early on in the Capstone effort, considerable effort was spent trying to create a UCore Digest of a CMD message that would provide a comprehensive, cohesive, stand-alone understanding of the air mission represented by CMD. For numerous reasons, the Capstone found this to be an elusive goal.

UCore relationships often are not sufficiently detailed to provide an accurate understanding of the real underlying relationship. For example, while it might be accurate to say that an aircraft is `AffiliatedWith` a sortie, it is vague to the point of being potentially misleading, and serves only to duplicate a relationship likely asserted at a different level of the same message. Likewise, C2 concepts such as Mission and Mission Task are merely Planned Events at the Digest layer. Adding C2 what-codes add semantics, but what-codes are not guaranteed to be understood by Digest-only consumers.

The Capstone Team concluded that UCore Digests do not provide any guarantee of a comprehensive summary understanding of the message. UCore Digests may consist of little more than the specification of a number of physical entities, along with their locations in space and time. The overarching relationships that tie all those entities together into a common mission or purpose may be missing, requiring unanticipated consumers to have enough sophistication to read other layers of the message for certain purposes.

9.3 XML Schema and Taxonomies are Insufficient to Specify UCore and C2 Core

Additional rules are needed to help define C2 Core and constrain messages to meet the rules of that definition consistently and accurately. UCore examples include specifying multiple conflicting taxonomy references, backwards time intervals, or open-ended time intervals. C2 Core examples include complex relationships, and referential integrity with respect to the linking strategy and missing data. Other examples have been discussed elsewhere in this paper.

9.4 Cross-Layer Linking via ulexlib:SameAsDigestReference

Creating proper ID references between layers was difficult. One of the issues encountered was the lack of guidance regarding how to do it properly. Message designers are faced with many options on how to design the linking structure, with no indication of which way is considered "best practice" or what is easiest for the consumer. It is safe to assume that different IES implementers will make different, potentially incompatible, choices.

It is desirable to choose a general principle to follow for a cross-linking strategy. This makes the message structure predictable and easy for the consumer. For example:

- Entities in the COI Payload should always link to the appropriate entity in the C2 Core Payload.

– OR –

- Entities in the COI Payload should always link to the appropriate entity in the UCore Digest.

The issue with these general principles is that due to modeling necessities, neither principle holds in all cases. Sometimes, the COI Payload mentions an entity that is not C2 related; and so it cannot be related to the C2 Core Payload, only to the Digest. In other situations, an entity exists in the C2 Core Payload and not the Digest, so that linking to the Digest is not an option. The team's experience indicates that the linking structure is in many ways designed on a case-by-case basis out of necessity driven by higher-level modeling decisions. This is very likely to complicate message consumption.

Finally, once a complex linking structure is chosen and implemented, there are currently no debugging tools available to help verify that instance messages use appropriate links and pointers. For message producers, it is cumbersome to verify manually that the pointer strategy in instance documents is correct.

9.4.1 Multi-Layered Messages Are Much More Difficult For Consumers

Through the experience of implementing a message consumer, the Capstone Team discovered a number of specific "pain points" that will make implementing C2 Core message consumers more difficult than implementing consumption software for a standard COI schema. Many of the specific issues have been discussed earlier in this paper so, rather than reiterating detailed explanations, the ones that bear on message consumption are simply enumerated here.

1. Developers have the fundamental task of re-assembling a multi-layered message.
    a. The "linking strategy" within multi-layered documents is not formally documented within a C2 Core IES.
    b. There is currently no standard API for performing this task.
2. Developers must understand and be conversant with many more standards and specifications. When consuming a COI Payload, the consumer need only understand the COI schema. When consuming a multi-layered message, the consumer must at a minimum understand UCore, ULEX, DDMS, IC-ISM, C2 Core, and the COI schema.
3. At present, C2 Core is insufficiently specified regarding how to validate instance messages. Consumers may not be able to sort out invalid/meaningless messages from valid messages before processing even begins.

In many cases, there are remedies available to address these challenges. For example, much complexity could be hidden through the use of a standardized message consumption API, and with the issuance of additional best practices guidance for message consumption. But without such products, it is likely that implementing C2 Core message consumption software will be slower, more expensive, and more difficult than traditional development options.

10 Conclusions

10.1 Summary of Findings

In both use cases, the test programs were successful in correctly producing the expected outputs, indicating that the three-layer messages were sufficiently detailed to be useful for both C2 Core and COI consumers. At the same time, the Capstone Team documented how using messages compliant with an IES based on the layered message design demonstrated in this Capstone is not a simple task. Quite a bit of infrastructure needed to be built to support even simple scenarios. Part of this is attributable to the experimental, "leading edge" nature of the investigation. However, the Team includes in this report many well-substantiated reasons why C2 Core and UCore proponents need to provide additional design specification details and support tools before value propositions can be realized from layered messages.

A layered architecture for information exchange, such as that advocated by C2 Core (and UCore), introduces special developer challenges and requires specificity to avoid introducing interoperability break points. The Capstone Team grouped specific challenges to the way ahead into four areas:

- Guidance for how and what information to distribute across all three layers is needed. Layered messaging presents a complex set of issues. There are differences in the modeling paradigms among the three layers[23], plus few clear requirements to help anticipate the content needs of the broader community of consumers. Without more guidance the risk of inconsistent content representation increases thereby reducing the potential for interoperability. Also needed are standardized interfaces to hide the underlying complexity of representation choices to reduce effort and increase consistency for message producers and consumers.

- Additional formalisms are needed to express domain core models which need to be distributed across the UCore Digest and Domain Core Payload layers. The formalisms must take into account that the UCore Digest layer by itself has no means to enforce adherence to the domain core model. Not only should these formalisms express the conceptual intent of the distributed domain core model, but they are also the basis for validating the content of layered message instances. Traditional XML data models are not burdened with these extra requirements.

- Additional design constraints for domain IES developers are needed regarding how to link information across layers[24]. Not having these design constraints may result in inconsistent implementations across the domain. This problem is exacerbated by a lack of debugging tools to verify that instance messages use links and pointers per any design constraints selected.

- A decision across the domain is needed on how to utilize the UCore Digest. The UCore Digest without extensions does not guarantee provision of sufficient content to support

[23] UCore Digest model is a flat list of objects, while typical XML such as that which would appear in a payload, is a hierarchical model.

[24] There are several different ULEX linking options using "same-as" pointers. In addition, to date, there are not sufficient ways to express the linking business rules needed to ensure coherent layered messages.

stand-alone understanding of messages by unanticipated consumers. Adding extensions in the Digest[25] provides more content but not necessarily content interpretable by the unanticipated UCore consumer.

Without responses to these findings, it is unlikely that the current technical framework is sufficient to achieve the multi-layered, data interoperability promise it seeks.

10.2 Recommendations

The C2 Core Specification can provide formal guidance to help mitigate the risk to interoperability by facilitating consistent information distribution across the message layers and determining the optimal linking strategy. Additionally, standardized message APIs for production, validation, and consumption could be developed.

Should the goal of C2 Core be to provide multi-community, non-redundant layered interoperability, then the following recommendations already suggested in the findings above should be considered:

- Decide how the UCore Digest shall be utilized by the domain participants and to what value and for whom.
- Develop formalisms for expressing and enforcing data models that span digest and payload layers.
- Provide implementation rules to IES developers regarding how to link information across layers.
- Develop extensions to standard XML tools for supporting consistent production, consumption and validation of layered messages.

Problems uncovered by the Capstone Team can be addressed specifically for the C2 domain rather than waiting for the UCore Specification to provide all the answers. In turn, the C2 domain can influence best practices and tools across the whole UCore community and help to mold future releases of the UCore Specification.

[25] Digest extension mechanisms include Simple Properties and community specific code spaces.

Appendix A Content Review

A.1 Sample C2 Core Content

The appendix describes how the Capstone Team arrived at sample C2 Core content for the experiments described in this paper.

While the intent of the Capstone was not to produce high-quality content for later inclusion in the official C2 Core, it was considered important to create highly plausible content. This meant meeting two primary criteria:

- The content should represent a real, joint C2 need.
- The concepts represented by the content should be in active use by more than one COI.

A.2 MITRE COI Member Review

The Capstone Team sent out a message to roughly twelve MITRE COI participants soliciting COI exchange schemas that are actively being used. Some of the products that came back included:

- Common Route Definition (CRD) version 2.0.2
- Aircraft Collection Tasking Message (ACTM) schema
- Mission Task Request (MTR) schemas
- Air Force (AF) Global Cyberspace Integration Center (GCIC) Analysis presented to the Joint Command and Control (JC2) CPM (18-20 March 2008) containing a crosswalk of vocabularies and schemas as part of a survey for candidate concepts.
- Publish and Subscribe Services (PASS) overlay
- Shared Situational Awareness Tracks Framework (SSATF) – Net Enabled Command Capability (NECC) schemas.
- XML-Message Text Format (MTF) schemas (XML representations of standard US-MTF messages)

A.3 DoD Metadata Registry Review

The Capstone Team conducted a search of the DoD Metadata Registry (see metadata.dod.mil) for C2 and related topics in an attempt to discover schemas that could be useful for building defensible sample content. The results of that search are summarized below. For the most part, this survey did not yield usable schemas for consideration, but they are documented here to record due diligence. The schemas that were discovered showed a reasonable amount of overlap between the concepts in the experimental C2 Core data model and those found in operational C2 schemas. Thus the team concluded its C2 Core data model sufficiently represents real and cross-C2 data sharing needs.

- **Command and Control Namespace** (https://metadata.dod.mil/mdr/viewByNamespace.htm?selectedNamespace=C2)
    - Joint C2 Core Version 1.0

- **Air Defense Namespace**
  (https://metadata.dod.mil/mdr/viewByNamespace.htm?selectedNamespace=AD)
  70 schema documents; most from 2004, and all marked "deprecated" or "retired."
- **Combat Support Namespace**
  (https://metadata.dod.mil/mdr/viewByNamespace.htm?selectedNamespace=CSS)
  3 DTD documents; all date to 2001 with no users identified.
- **Coalition Namespace**
  (https://metadata.dod.mil/mdr/viewByNamespace.htm?selectedNamespace=COAL)
  103 Schemas; few from 2002 relating to GHD data, and a few that might be applicable:
  - (JC3IEDM-3.1d-WSOO-EntityElements-20081211.xsd) XML Schema Definition for all entity elements contained in the JC3IEDM 3.1d specification
- **Airspace Operations**
  (https://metadata.dod.mil/mdr/viewByNamespace.htm?selectedNamespace=AOP)
  68 schema documents, including a few things possibly useful. Most were not sufficiently described to discern usefulness or intent. Several were earmarked as potentially useful:
  - (CoT_track.xsd) CoT track subschema (2005)
  - (CBOVer2.0.xsd) Common Battle Object v2.0
  - TBMCS Target Management Service Schemas
- **C2 Shared Situational Awareness (SSA) COI**
  (https://metadata.dod.mil/mdr/viewByNamespace.htm?selectedNamespace=C2_SSA)
  314 schema documents
  - IMOM 4.2.0.1 schemas
- **Chemical, Biological, Radiological, Nuclear (CBRN) COI**
  (https://metadata.dod.mil/mdr/viewByNamespace.htm?selectedNamespace=CBRN)
  13 schemas available; all that appeared relevant were marked "retired."

Appendix B Existing UCore Consumer Software

Paul Franklin wrote a package that consumes UCore and National Information Exchange Model (NIEM)[26] Suspicious Activity Reporting (SAR) messages as part of UCore experimentation in 2008[27]. Franklin's software used UCore 2.0 alpha, and while his consumer software was geared towards SAR and based on an older version of the UCore schema and taxonomy, it was nonetheless very helpful in generating ideas for implementing UCore consumer software, most notably his use of the XMLBeans API.

The XMLBeans API allows compilation of an XML schema into a set of Java classes which can automatically read and validate document instances conforming to that schema. It also provides easy programmatic access to the data within those messages without the need to use Document Object Model (DOM)[28] or SAX[29] parsers to access the raw XML[30].

[26] See http://www.niem.gov.
[27] His software and report are available through MITRE.
[28] DOM is a way of representing an XML document internally as a large tree. DOM is typically used to keep a model of the entire document in memory at a given time. See also http://www.w3schools.com/dom/default.asp
[29] SAX is the "Simple API for XML", a method of processing XML documents that generates a stream of events that the programmer can handle. SAX is very helpful when the entire document is too large to be handled in memory as a single DOM tree. For more information see http://en.wikipedia.org/wiki/Simple_API_for_XML.
[30] Apache's XMLBeans API: http://xmlbeans.apache.org/

Appendix C Source Testing Environment

C.1 Source of CMD Messages

CMD messages were taken from a MITRE instance of the TBMCS development test bed. Messages were downloaded via an RSS feed, which when given proper authentication credentials and a set of search terms, would return a Rich Site Summary (RSS) formatted list of URLs containing CMD messages matching the desired criteria. The TBMCS development test bed additionally provided a filtering tool (the "Mission Filter Builder") that permitted the user to construct an RSS filter of only CMD messages matching sets of criteria. Those criteria in turn were searches against certain required fields within the source CMD messages.

The project used the RSS feed and the filtering tools to obtain a list of DirectAttack CMD messages, which were then used as the basis for the three-layer messages described in this report.

Figure 22: Mission Filter Builder is a screenshot of the mission filter builder.

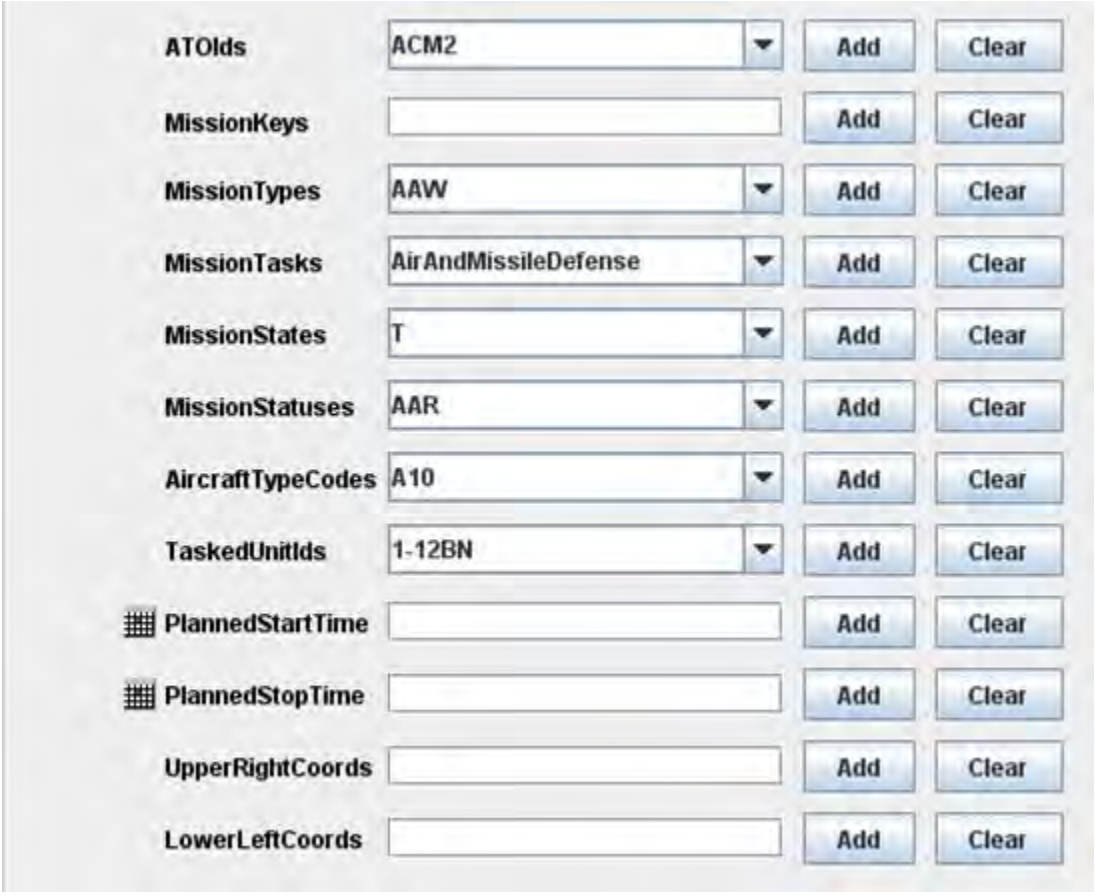

Figure 22: Mission Filter Builder

Appendix D Sample 3-Layer Document

```
<?xml version="1.0" encoding="UTF-8"?>
<ulexpd:doPublish xmlns:aop="http://xml.dod.mil/aop/cmdpayload/0.1"
xmlns:c2c="http://us.jfcom.mil/C2 Core/0.4" xmlns:ddms="http://metadata.dod.mil/mdr/ns/DDMS/2.0/"
xmlns:fn="http://www.w3.org/2005/xpath-functions" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:icism="urn:us:gov:ic:ism:v2" xmlns:lxslt="http://xml.apache.org/xslt"
xmlns:ucore="http://ucore.gov/ucore/2.0" xmlns:ulex="ulex:message:structure:1.0"
xmlns:ulexcodes="ulex:message:codes:1.0" xmlns:ulexlib="ulex:message:library:1.0"
xmlns:ulexpd="ulex:message:pd:1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="ulex:message:pd:1.0
../Schemas/UCore/2.0/ucore-message.xsd">
  <ulex:PublishMessageContainer>
    <ulex:PublishMessage>
      <ulex:PDMessageMetadata>
        <ulex:ULEXFramework>1.0</ulex:ULEXFramework>
        <ulex:ULEXImplementation>
          <ulex:ULEXImplementationVersion>2.0</ulex:ULEXImplementationVersion>
          <ulex:ULEXImplementationName>ucore-message</ulex:ULEXImplementationName>
        </ulex:ULEXImplementation>
        <ulex:MessageDateTime>2009-05-14T12:20:57-04:00</ulex:MessageDateTime>
        <ulex:MessageSequenceNumber>1</ulex:MessageSequenceNumber>
        <ucore:MessageClassification icism:classification="U" icism:ownerProducer="USA"/>
      </ulex:PDMessageMetadata>
      <ulex:DataSubmitterMetadata>
        <ucore:SystemIdentifier>Some meaningless default</ucore:SystemIdentifier>
        <ucore:SystemContact>
          <ddms:Organization>
            <ddms:name>MITRE Corporation</ddms:name>
          </ddms:Organization>
        </ucore:SystemContact>
      </ulex:DataSubmitterMetadata>
      <ulex:DataItemPackage>
        <ulex:PackageMetadata>
          <ulex:DataItemID>MzYyaTAxMDM5MHAwMDh2NDE4MDFnaGlq</ulex:DataItemID>
          <ulex:DataItemReferenceID>MzYyaTAxMDM5MHAwMDh2NDE4MDFnaGlq</ulex:DataItemReferenceID>
          <ucore:DataItemStatus ucore:label="Test"/>
          <ulex:DataOwnerMetadata>
            <ucore:DataOwnerIdentifier>
              <ddms:Organization>
                <ddms:name>MITRE Corporation</ddms:name>
              </ddms:Organization>
            </ucore:DataOwnerIdentifier>
            <ucore:DataOwnerContact>
              <ddms:Organization>
                <ddms:name>MITRE Corporation</ddms:name>
              </ddms:Organization>
            </ucore:DataOwnerContact>
          </ulex:DataOwnerMetadata>
          <ucore:DisseminationCriteria icism:classification="U" icism:classifiedBy="USA"/>
        </ulex:PackageMetadata>
        <ucore:Digest>
          <ucore:Event id="Mission_idroot_2_0">
            <ucore:Identifier ucore:code="MissionKey" ucore:label="Mission Key"
ucore:codespace="http://xml.dod.mil/aop/cmd/">MzYyaTAxMDM5MHAwMDh2NDE4MDFnaGlq</ucore:Identifier>
            <ucore:Identifier ucore:label="TBMCS-ABP Mission Alias">ACM2/1801</ucore:Identifier>
            <ucore:Identifier ucore:label="TBMCS-WWID Mission
Alias">362i010390u009V</ucore:Identifier>
            <ucore:What ucore:code="PlannedEvent"
ucore:codespace="http://ucore.gov/ucore/2.0/codespace/"/>
            <ucore:What ucore:code="Mission" ucore:codespace="http://xml.dod.mil/aop/cmd/"/>
            <ucore:What ucore:code="ATK"
ucore:codespace="http://xml.dod.mil/aop/cmd#ATOMissionTypeCodes/PrimaryMissionTypeCode"/>
            <ucore:What ucore:code="AIR"
ucore:codespace="http://xml.dod.mil/aop/cmd/#MissionCategoryCode"/>
            <ucore:What ucore:code="LIVE"
ucore:codespace="http://xml.dod.mil/aop/cmd/#Authenticity"/>
          </ucore:Event>
          <ucore:Organization id="Country_US">
```



```
            <ucore:Identifier ucore:code="ISO 3166-1-alpha-2" ucore:label="Country Code"
ucore:codespace="http://www.iso.org/iso/english_country_names_and_code_elements/">US
</ucore:Identifier>
            <ucore:What ucore:code="Organization"
ucore:codespace="http://ucore.gov/ucore/2.0/codespace/"/>
            <ucore:What ucore:code="Country" ucore:codespace="http://us.jfcom.mil/C2
Core/codespace/"/>
            <ucore:Name>
              <ucore:Value>US</ucore:Value>
            </ucore:Name>
          </ucore:Organization>
          <ucore:Organization id="Unit idTaskedUnit 2 40">
            <ucore:Identifier ucore:code="UnitId" ucore:label="Unit Id"
ucore:codespace="http://xml.dod.mil/aop/cmd/"></ucore:Identifier>
            <ucore:Identifier ucore:code="UnitName" ucore:label="Unit Name"
ucore:codespace="http://xml.dod.mil/aop/cmd/">213SFS</ucore:Identifier>
            <ucore:What ucore:code="Organization"
ucore:codespace="http://ucore.gov/ucore/2.0/codespace/"/>
            <ucore:What ucore:code="TaskedUnit" ucore:codespace="http://xml.dod.mil/aop/cmd/"/>
            <ucore:Name>
              <ucore:Value>213SFS</ucore:Value>
            </ucore:Name>
          </ucore:Organization>
          <ucore:Location id="Tasked_Unit_LocationidTaskedUnit_2_40">
            <ucore:Identifier ucore:code="LocationId" ucore:label="Operating Location Id"
ucore:codespace="http://xml.dod.mil/aop/cmd/">NENWNjVlZmdoaWprbG1ub3BxcnN0dXZ3</ucore:Identifier>
            <ucore:Identifier ucore:code="LocationName" ucore:label="Operating Location Name"
ucore:codespace="http://xml.dod.mil/aop/cmd/">CV65</ucore:Identifier>
          </ucore:Location>
          <ucore:LocatedAt id="Unit_336FS_LocatedAt">
            <ucore:EntityRef ref="Unit_idTaskedUnit_2_40"/>
            <ucore:LocationRef ref="Tasked_Unit_LocationidTaskedUnit_2_40"/>
          </ucore:LocatedAt>
          <ucore:Controls id="Country_Unit_Control">
            <ucore:AgentRef ref="Country US"/>
            <ucore:EntityRef ref="Unit_idTaskedUnit_2_40"/>
          </ucore:Controls>
          <ucore:Event id="MissionTask_idDirectAttack_2_493">
            <ucore:Identifier ucore:code="TaskId" ucore:label="Task Id"
ucore:codespace="http://xml.dod.mil/aop/cmd/codespace/">1</ucore:Identifier>
            <ucore:What ucore:code="PlannedEvent"
ucore:codespace="http://ucore.gov/ucore/2.0/codespace/"/>
            <ucore:What ucore:code="DirectAttack" ucore:codespace="http://us.jfcom.mil/C2
Core/codespace/"/>
            <ucore:What ucore:code="ATK"
ucore:codespace="http://xml.dod.mil/aop/cmd#MissionTypeCode"/>
          </ucore:Event>
          <ucore:InvolvedIn id="MissionTask_idDirectAttack_2_493_Unit_idTaskedUnit_2_40">
            <ucore:AgentRef ref="Unit_idTaskedUnit_2_40"/>
            <ucore:EventRef ref="MissionTask_idDirectAttack_2_493"/>
          </ucore:InvolvedIn>
          <ucore:Location id="MissionTaskLocation idDirectAttack 2 493">
            <ucore:Identifier ucore:code="LocationId" ucore:label="ID"
ucore:codespace="http://xml.dod.mil/aop/cmd/">TC1</ucore:Identifier>
            <ucore:Identifier ucore:code="LocationName" ucore:label="Name"
ucore:codespace="http://xml.dod.mil/aop/cmd/">TRAINING CAMP</ucore:Identifier>
            <ucore:GeoLocation>
              <ucore:Point>
                <gml:Point gml:id="gmlMissionTaskLocation idDirectAttack_2_493"
srsName="http://metadata.dod.mil/mdr/ns/GSIP/crs/WGS84E_2D">
                  <gml:pos>7.561527777777776 -10.87086111111111</gml:pos>
                </gml:Point>
              </ucore:Point>
            </ucore:GeoLocation>
          </ucore:Location>
          <ucore:OccursAt id="MissionTask_OccursAt_idDirectAttack_2_493">
            <ucore:Time>
              <ucore:TimeInterval>
                <ucore:StartTime>
                  <ucore:Value>2009-02-26T10:20:00.000Z</ucore:Value>
                </ucore:StartTime>
                <ucore:EndTime>
                  <ucore:Value>2009-02-26T10:20:00.000Z</ucore:Value>
```

```
                </ucore:EndTime>
              </ucore:TimeInterval>
            </ucore:Time>
            <ucore:EventRef ref="MissionTask idDirectAttack 2 493"/>
            <ucore:LocationRef ref="MissionTaskLocation_idDirectAttack_2_493"/>
          </ucore:OccursAt>
          <ucore:AffiliatedWith id="Mission_MissionTask_idDirectAttack_2_493_Affiliation">
            <ucore:ThingRef ref="Mission_idroot_2_0"/>
            <ucore:ThingRef ref="MissionTask_idDirectAttack_2_493"/>
          </ucore:AffiliatedWith>
          <ucore:Entity id="Target_idTarget_2_573">
            <ucore:Identifier ucore:code="TargetId" ucore:label="ID"
ucore:codespace="http://xml.dod.mil/aop/cmd/">TC1</ucore:Identifier>
            <ucore:Identifier ucore:code="TargetName" ucore:label="Name"
ucore:codespace="http://xml.dod.mil/aop/cmd/">TRAINING CAMP</ucore:Identifier>
            <ucore:Identifier ucore:code="ComponentTargetId" ucore:label="System Component Target
Id" ucore:codespace="http://xml.dod.mil/aop/cmd/">TC1</ucore:Identifier>
            <ucore:Identifier ucore:code="Facility" ucore:label="Facility"
ucore:codespace="http://xml.dod.mil/aop/cmd/">TRAINING CAMP</ucore:Identifier>
            <ucore:What ucore:code="Infrastructure"
ucore:codespace="http://ucore.gov/ucore/2.0/codespace/"/>
            <ucore:What ucore:code="Target" ucore:codespace="http://xml.dod.mil/aop/cmd/"/>
          </ucore:Entity>
          <ucore:Location id="Target_idTarget_2_573_Location">
            <ucore:Identifier ucore:code="LocationId" ucore:label="ID"
ucore:codespace="http://xml.dod.mil/aop/cmd/">TC1</ucore:Identifier>
            <ucore:Identifier ucore:code="LocationName" ucore:label="Name"
ucore:codespace="http://xml.dod.mil/aop/cmd/">TRAINING CAMP</ucore:Identifier>
            <ucore:GeoLocation>
              <ucore:Point>
                <gml:Point gml:id="gmlTarget_idTarget_2_573_Location"
srsName="http://metadata.dod.mil/mdr/ns/GSIP/crs/WGS84E_2D">
                  <gml:pos>7.561527777777776 -10.87086111111111</gml:pos>
                </gml:Point>
              </ucore:Point>
            </ucore:GeoLocation>
          </ucore:Location>
          <ucore:LocatedAt id="Target_idTarget_2_573_LocatedAt">
            <ucore:EntityRef ref="Target idTarget 2 573"/>
            <ucore:LocationRef ref="Target_idTarget_2_573_Location"/>
          </ucore:LocatedAt>
          <ucore:AffiliatedWith id="MissionTask_1_Target_SH3_Affiliation">
            <ucore:ThingRef ref="MissionTask_idDirectAttack_2_493"/>
            <ucore:ThingRef ref="Target_idTarget_2_573"/>
          </ucore:AffiliatedWith>
          <ucore:Entity id="Aircraft_idAircraftSortie_2_54">
            <ucore:What ucore:code="Aircraft"
ucore:codespace="http://ucore.gov/ucore/2.0/codespace/"/>
            <ucore:What ucore:code="FA18A"
ucore:codespace="http://xml.dod.mil/aop/cmd/#AircraftModelCode"/>
          </ucore:Entity>
          <ucore:Entity id="Aircraft idAircraftSortie_2_273">
            <ucore:What ucore:code="Aircraft"
ucore:codespace="http://ucore.gov/ucore/2.0/codespace/"/>
            <ucore:What ucore:code="FA18A"
ucore:codespace="http://xml.dod.mil/aop/cmd/#AircraftModelCode"/>
          </ucore:Entity>
          <ucore:Collection id="Equipment Collection_1">
            <ucore:What ucore:code="Equipment"
ucore:codespace="http://ucore.gov/ucore/2.0/codespace/"/>
            <ucore:ThingRef ref="Aircraft_idAircraftSortie_2_54"/>
            <ucore:ThingRef ref="Aircraft_idAircraftSortie_2_273"/>
          </ucore:Collection>
          <ucore:AffiliatedWith id="MissionTask idMission_2_2_Equipment">
            <ucore:ThingRef ref="Mission_idroot_2_0"/>
            <ucore:ThingRef ref="Equipment_Collection_1"/>
          </ucore:AffiliatedWith>
        </ucore:Digest>
        <ulex:StructuredPayload ulexlib:id="C2 CorePayload">
          <ulex:StructuredPayloadMetadata>
            <ulex:CommunityURI>https://us.jfcom.mil/C2 Core</ulex:CommunityURI>
            <ulex:CommunityDescription>C2 Core</ulex:CommunityDescription>
            <ulex:CommunityVersion>1.0</ulex:CommunityVersion>
```

```
          </ulex:StructuredPayloadMetadata>
          <c2c:C2 CoreMessage xsi:schemaLocation="http://us.jfcom.mil/C2 Core/0.4 ../Schemas/C2
Core/C2 Core_Candidate_Schema.xsd">
            <c2c:Missions ulexlib:id="idroot 2 0-missions C2 Core">
              <c2c:Mission ulexlib:id="Mission_idroot_2_0_C2 Core">
                <ulexlib:SameAsDigestReference ulexlib:nvref="Mission_idroot_2_0"/>
                <c2c:MissionOwnerContact/>
                <c2c:MissionExecutionStatusCode>CHG</c2c:MissionExecutionStatusCode>
                <c2c:AuthenticityCode>LIVE</c2c:AuthenticityCode>
                <c2c:MissionObjective ulexlib:id="MissionObjective_idroot_2_0_C2 Core">

<c2c:MissionObjectiveCategoryCode>DestroyTarget</c2c:MissionObjectiveCategoryCode>
                  <c2c:MissionOjectiveDescription>
                      Increase security in the region by distroying suspected enemy control
center.
                  </c2c:MissionOjectiveDescription>
                </c2c:MissionObjective>
                <c2c:Sorties>
                  <c2c:Sortie ulexlib:id="idSortie_2_53">
                    <c2c:Equipment ulexlib:id="Equipment_idAircraft_2_66_C2 Core">
                      <ulexlib:SameAsDigestReference
ulexlib:nvref="Aircraft idAircraftSortie_2_54"/>
                    </c2c:Equipment>
                    <c2c:Route ulexlib:id="Route_idRoute_2_72_C2 Core">
                      <c2c:RoutePoints ulexlib:id="RoutePoints_idRoute_2_72_C2 Core">
                        <c2c:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_74_C2 Core">
                          <c2c:RoutePointLocation
ulexlib:id="RoutePointLocation_idRoutePoint_2_74_C2 Core">
                            <c2c:LocationName>USS ENTERPRISE</c2c:LocationName>
                            <ucore:GeoLocation>
                              <ucore:Point>
                                <gml:Point gml:id="RoutePointLocation_gml_idRoutePoint_2_74_C2
Core" srsName="http://metadata.dod.mil/mdr/ns/GSIP/crs/WGS84E 2D">
                                  <gml:pos>5.285555555555556 -17.50611111111111</gml:pos>
                                </gml:Point>
                              </ucore:Point>
                            </ucore:GeoLocation>
                          </c2c:RoutePointLocation>
                          <c2c:TimeAtPoint>
                            <c2c:TimeStatusCode>Planned</c2c:TimeStatusCode>
                            <ucore:TimeInstant>
                              <ucore:Value>2009-02-26T09:15:00.000Z</ucore:Value>
                            </ucore:TimeInstant>
                          </c2c:TimeAtPoint>
                        </c2c:RoutePoint>
                        <c2c:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_135_C2 Core">
                          <c2c:RoutePointLocation
ulexlib:id="RoutePointLocation_idRoutePoint_2_135_C2 Core">
                            <c2c:LocationName>TC1</c2c:LocationName>
                            <ucore:GeoLocation>
                              <ucore:Point>
                                <gml:Point gml:id="RoutePointLocation_gml_idRoutePoint_2_135_C2
Core" srsName="http://metadata.dod.mil/mdr/ns/GSIP/crs/WGS84E_2D">
                                  <gml:pos>7.561527777777776 -10.87086111111111</gml:pos>
                                </gml:Point>
                              </ucore:Point>
                            </ucore:GeoLocation>
                          </c2c:RoutePointLocation>
                          <c2c:TimeAtPoint>
                            <c2c:TimeStatusCode>Planned</c2c:TimeStatusCode>
                            <ucore:TimeInstant>
                              <ucore:Value>2009-02-26T10:20:00.000Z</ucore:Value>
                            </ucore:TimeInstant>
                          </c2c:TimeAtPoint>
                        </c2c:RoutePoint>
                        <c2c:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_204_C2 Core">
                          <c2c:RoutePointLocation
ulexlib:id="RoutePointLocation idRoutePoint 2 204 C2 Core">
                            <c2c:LocationName>USS ENTERPRISE</c2c:LocationName>
                            <ucore:GeoLocation>
                              <ucore:Point>
                                <gml:Point gml:id="RoutePointLocation_gml_idRoutePoint_2_204_C2
Core" srsName="http://metadata.dod.mil/mdr/ns/GSIP/crs/WGS84E_2D">
```

```
                              <gml:pos>5.285555555555556 -17.50611111111111</gml:pos>
                            </gml:Point>
                          </ucore:Point>
                        </ucore:GeoLocation>
                      </c2c:RoutePointLocation>
                      <c2c:TimeAtPoint>
                        <c2c:TimeStatusCode>Planned</c2c:TimeStatusCode>
                        <ucore:TimeInstant>
                          <ucore:Value>2009-02-26T11:18:00.000Z</ucore:Value>
                        </ucore:TimeInstant>
                      </c2c:TimeAtPoint>
                    </c2c:RoutePoint>
                  </c2c:RoutePoints>
                </c2c:Route>
                <c2c:TasksPerformed>
                  <c2c:MissionTaskRef ulexlib:vref="MissionTask_idDirectAttack_2_493_C2
Core"/>
                </c2c:TasksPerformed>
              </c2c:Sortie>
              <c2c:Sortie ulexlib:id="idSortie_2_272">
                <c2c:Equipment ulexlib:id="Equipment_idAircraft_2_285_C2 Core">
                  <ulexlib:SameAsDigestReference
ulexlib:nvref="Aircraft idAircraftSortie_2_273"/>
                </c2c:Equipment>
                <c2c:Route ulexlib:id="Route_idRoute_2_291_C2 Core">
                  <c2c:RoutePoints ulexlib:id="RoutePoints_idRoute_2_291_C2 Core">
                    <c2c:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_293_C2 Core">
                      <c2c:RoutePointLocation
ulexlib:id="RoutePointLocation idRoutePoint 2 293 C2 Core">
                        <c2c:LocationName>USS ENTERPRISE</c2c:LocationName>
                        <ucore:GeoLocation>
                          <ucore:Point>
                            <gml:Point gml:id="RoutePointLocation_gml_idRoutePoint_2_293_C2
Core" srsName="http://metadata.dod.mil/mdr/ns/GSIP/crs/WGS84E_2D">
                              <gml:pos>5.285555555555556 -17.50611111111111</gml:pos>
                            </gml:Point>
                          </ucore:Point>
                        </ucore:GeoLocation>
                      </c2c:RoutePointLocation>
                      <c2c:TimeAtPoint>
                        <c2c:TimeStatusCode>Planned</c2c:TimeStatusCode>
                        <ucore:TimeInstant>
                          <ucore:Value>2009-02-26T09:15:00.000Z</ucore:Value>
                        </ucore:TimeInstant>
                      </c2c:TimeAtPoint>
                    </c2c:RoutePoint>
                    <c2c:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_354_C2 Core">
                      <c2c:RoutePointLocation
ulexlib:id="RoutePointLocation_idRoutePoint_2_354_C2 Core">
                        <c2c:LocationName>TC1</c2c:LocationName>
                        <ucore:GeoLocation>
                          <ucore:Point>
                            <gml:Point gml:id="RoutePointLocation_gml_idRoutePoint_2_354_C2
Core" srsName="http://metadata.dod.mil/mdr/ns/GSIP/crs/WGS84E_2D">
                              <gml:pos>7.561527777777776 -10.87086111111111</gml:pos>
                            </gml:Point>
                          </ucore:Point>
                        </ucore:GeoLocation>
                      </c2c:RoutePointLocation>
                      <c2c:TimeAtPoint>
                        <c2c:TimeStatusCode>Planned</c2c:TimeStatusCode>
                        <ucore:TimeInstant>
                          <ucore:Value>2009-02-26T10:20:00.000Z</ucore:Value>
                        </ucore:TimeInstant>
                      </c2c:TimeAtPoint>
                    </c2c:RoutePoint>
                    <c2c:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_423_C2 Core">
                      <c2c:RoutePointLocation
ulexlib:id="RoutePointLocation idRoutePoint 2 423 C2 Core">
                        <c2c:LocationName>USS ENTERPRISE</c2c:LocationName>
                        <ucore:GeoLocation>
                          <ucore:Point>
```

```
                              <gml:Point gml:id="RoutePointLocation_gml_idRoutePoint_2_423_C2
Core" srsName="http://metadata.dod.mil/mdr/ns/GSIP/crs/WGS84E_2D">
                                <gml:pos>5.285555555555556 -17.50611111111111</gml:pos>
                              </gml:Point>
                            </ucore:Point>
                          </ucore:GeoLocation>
                        </c2c:RoutePointLocation>
                        <c2c:TimeAtPoint>
                          <c2c:TimeStatusCode>Planned</c2c:TimeStatusCode>
                          <ucore:TimeInstant>
                            <ucore:Value>2009-02-26T11:18:00.000Z</ucore:Value>
                          </ucore:TimeInstant>
                        </c2c:TimeAtPoint>
                      </c2c:RoutePoint>
                    </c2c:RoutePoints>
                  </c2c:Route>
                  <c2c:TasksPerformed>
                    <c2c:MissionTaskRef ulexlib:vref="MissionTask_idDirectAttack_2_493_C2
Core"/>
                  </c2c:TasksPerformed>
                </c2c:Sortie>
              </c2c:Sorties>
              <c2c:MissionTasks ulexlib:id="MissionTasks idMissionTasks 2 491 C2 Core">
                <c2c:MissionTask ulexlib:id="MissionTask_idDirectAttack_2_493_C2 Core">
                  <ulexlib:SameAsDigestReference
ulexlib:nvref="MissionTask_idDirectAttack_2_493"/>
                  <c2c:AreaOfResponsibilityCode>DAVIS MONTHAN</c2c:AreaOfResponsibilityCode>
                  <c2c:TaskExecutionStatusCode>Z</c2c:TaskExecutionStatusCode>
                  <c2c:Occurance ulexlib:id="Occurance_idDirectAttack_2_493_C2 Core">
                    <ulexlib:SameAsDigestReference
ulexlib:nvref="MissionTask_OccursAt_idDirectAttack_2_493"/>
                    <c2c:TimeStatusCode>Planned</c2c:TimeStatusCode>
                  </c2c:Occurance>
                  <c2c:TaskedUnit ulexlib:id="Unit_idTaskedUnit_2_40_C2 Core">
                    <ulexlib:SameAsDigestReference ulexlib:nvref="Unit_idTaskedUnit_2_40"/>
                  </c2c:TaskedUnit>
                  <c2c:Target ulexlib:id="Target_idTarget_2_573_C2 Core">
                    <ulexlib:SameAsDigestReference ulexlib:nvref="Target_idTarget_2_573"/>
                    <c2c:TargetPriorityCode>2</c2c:TargetPriorityCode>
                    <c2c:TargetEffectCode></c2c:TargetEffectCode>
                  </c2c:Target>
                </c2c:MissionTask>
              </c2c:MissionTasks>
            </c2c:Mission>
          </c2c:Missions>
        </c2c:C2 CoreMessage>
      </ulex:StructuredPayload>
      <ulex:StructuredPayload ulexlib:id="AO-COI-Payload">
        <ulex:StructuredPayloadMetadata>
          <ulex:CommunityURI>http://xml.dod.mil/aop/cmd</ulex:CommunityURI>
          <ulex:CommunityDescription>Air Operations Common Mission
Definition</ulex:CommunityDescription>
          <ulex:CommunityVersion>1.0</ulex:CommunityVersion>
        </ulex:StructuredPayloadMetadata>
        <aop:AO_COI_UCore_C2 Core_Payload
xsi:schemaLocation="http://xml.dod.mil/aop/cmdpayload/0.1
../Schemas/AOP/AOP_CMD_Payload_Candidate_Schema.xsd">
          <aop:Missions ulexlib:id="Missions 1 p2">
            <aop:Mission ulexlib:id="Mission 1 p2">
              <ulexlib:SameAsPayloadReference ulexlib:nvref="Mission_idroot_2_0_C2 Core"
ulexlib:pnvref="C2 CorePayload"/>
              <aop:AdministrativeState>
                <aop:AdministrativeStateCode>T</aop:AdministrativeStateCode>
                <aop:ATOIncludeState>
                  <aop:ATOKey>
                    <aop:MsgId>ACM2</aop:MsgId>
                    <aop:MsgChangeNumber>1</aop:MsgChangeNumber>
                  </aop:ATOKey>
                </aop:ATOIncludeState>
              </aop:AdministrativeState>
              <aop:MissionAliases>
                <aop:MissionAlias>
                  <aop:MissionNumber>ACM2/1801</aop:MissionNumber>
```

```
                    <aop:SourceSystem>TBMCS-ABP</aop:SourceSystem>
                    <aop:SourceNode>DAVIS MONTHAN</aop:SourceNode>
                  </aop:MissionAlias>
                  <aop:MissionAlias>
                    <aop:MissionNumber>362i010390u009V</aop:MissionNumber>
                    <aop:SourceSystem>TBMCS-WWID</aop:SourceSystem>
                    <aop:SourceNode>DAVIS MONTHAN</aop:SourceNode>
                  </aop:MissionAlias>
                </aop:MissionAliases>
                <aop:Sorties ulexlib:id="Sorties idroot 2 0">
                  <aop:AircraftSortie ulexlib:id="AircraftSortie_idAircraftSortie_2_54_aop">
                    <aop:Aircraft ulexlib:id="Aircraft idAircraft 2 66 aop">
                      <ulexlib:SameAsPayloadReference ulexlib:nvref="Equipment_idAircraft_2_66_C2
Core" ulexlib:pnvref="C2 CorePayload"/>
                      <aop:PlannedPrimaryConfiguration>BEST</aop:PlannedPrimaryConfiguration>
                    </aop:Aircraft>
                    <aop:Routes ulexlib:id="Routes_idAircraftSortie_2_54_aop">
                      <aop:Route ulexlib:id="idRoute 2 72">
                        <ulexlib:SameAsPayloadReference ulexlib:nvref="Route_idRoute_2_72_C2
Core" ulexlib:pnvref="C2 CorePayload"/>
                        <aop:RoutePoints ulexlib:id="RoutePoints_idRoutePoints_2_73_aop">
                          <aop:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_74_aop">
                            <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePoint_idRoutePoint_2_74_C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                            <aop:RoutePointTypeCode>TKF</aop:RoutePointTypeCode>
                            <aop:RoutePointLocation
ulexlib:id="RoutePointLocation idRoutePoint 2 74 aop">
                              <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePointLocation idRoutePoint 2 74 C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                              <aop:LocationTypeCode>BASE</aop:LocationTypeCode>
                              <aop:LocationId>NENWNjVlZmdoaWprbG1ub3BxcnN0dXZ3</aop:LocationId>
                              <aop:SourceSystem>TBMCS</aop:SourceSystem>
                              <aop:SourceNode>DAVIS MONTHAN</aop:SourceNode>
                              <aop:ICAO>CV65</aop:ICAO>
                            </aop:RoutePointLocation>
                            <aop:Callsigns>
                              <aop:Callsign>
                                <aop:CallWord>LION</aop:CallWord>
                                <aop:CallNumber>01</aop:CallNumber>
                                <aop:AOR>DAVIS MONTHAN</aop:AOR>
                                <aop:ABPId>ACM2</aop:ABPId>
                                <aop:ATOMissionNumber>1801</aop:ATOMissionNumber>
                                <aop:IFFSIFs>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>1</aop:Mode>
                                    <aop:Code>00</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>2</aop:Mode>
                                    <aop:Code>2121</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>3</aop:Mode>
                                    <aop:Code>2121</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                </aop:IFFSIFs>
                              </aop:Callsign>
                            </aop:Callsigns>
                          </aop:RoutePoint>
                          <aop:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_135_aop">
                            <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePoint idRoutePoint 2 135 C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                            <aop:RoutePointTypeCode>TGT</aop:RoutePointTypeCode>
                            <aop:RoutePointLocation
ulexlib:id="RoutePointLocation_idRoutePoint_2_135_aop">
                              <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePointLocation_idRoutePoint_2_135_C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                              <aop:LocationTypeCode>GEOGRAPHICAREA</aop:LocationTypeCode>
                              <aop:LocationId>TRAINING CAMP</aop:LocationId>
                              <aop:SourceSystem>TBMCS</aop:SourceSystem>
                              <aop:SourceNode>DAVIS MONTHAN</aop:SourceNode>
                              <aop:ICAO></aop:ICAO>
                            </aop:RoutePointLocation>
```

```
                            <aop:Callsigns>
                              <aop:Callsign>
                                <aop:CallWord>LION</aop:CallWord>
                                <aop:CallNumber>01</aop:CallNumber>
                                <aop:AOR>DAVIS MONTHAN</aop:AOR>
                                <aop:ABPId>ACM2</aop:ABPId>
                                <aop:ATOMissionNumber>1801</aop:ATOMissionNumber>
                                <aop:IFFSIFs>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>1</aop:Mode>
                                    <aop:Code>00</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>2</aop:Mode>
                                    <aop:Code>2121</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>3</aop:Mode>
                                    <aop:Code>2121</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                </aop:IFFSIFs>
                              </aop:Callsign>
                            </aop:Callsigns>
                          </aop:RoutePoint>
                          <aop:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_204_aop">
                            <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePoint idRoutePoint 2 204 C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                            <aop:RoutePointTypeCode>LND</aop:RoutePointTypeCode>
                            <aop:RoutePointLocation
ulexlib:id="RoutePointLocation_idRoutePoint_2_204_aop">
                              <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePointLocation_idRoutePoint_2_204_C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                              <aop:LocationTypeCode>BASE</aop:LocationTypeCode>
                              <aop:LocationId>NENWNjVlZmdoaWprbG1ub3BxcnN0dXZ3</aop:LocationId>
                              <aop:SourceSystem>TBMCS</aop:SourceSystem>
                              <aop:SourceNode>DAVIS MONTHAN</aop:SourceNode>
                              <aop:ICAO>CV65</aop:ICAO>
                            </aop:RoutePointLocation>
                            <aop:Callsigns>
                              <aop:Callsign>
                                <aop:CallWord>LION</aop:CallWord>
                                <aop:CallNumber>01</aop:CallNumber>
                                <aop:AOR>DAVIS MONTHAN</aop:AOR>
                                <aop:ABPId>ACM2</aop:ABPId>
                                <aop:ATOMissionNumber>1801</aop:ATOMissionNumber>
                                <aop:IFFSIFs>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>1</aop:Mode>
                                    <aop:Code>00</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>2</aop:Mode>
                                    <aop:Code>2121</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>3</aop:Mode>
                                    <aop:Code>2121</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                </aop:IFFSIFs>
                              </aop:Callsign>
                            </aop:Callsigns>
                          </aop:RoutePoint>
                        </aop:RoutePoints>
                        <aop:RouteLastModified>
                          <aop:SourceSystem>TBMCS</aop:SourceSystem>
                          <aop:SourceNode>DAVIS MONTHAN</aop:SourceNode>
                          <ucore:TimeInstant>
                            <ucore:Value>2009-02-25T16:06:04.000Z</ucore:Value>
                          </ucore:TimeInstant>
                        </aop:RouteLastModified>
                      </aop:Route>
                    </aop:Routes>
                  </aop:AircraftSortie>
```

```
                  <aop:AircraftSortie ulexlib:id="AircraftSortie_idAircraftSortie_2_273_aop">
                    <aop:Aircraft ulexlib:id="Aircraft_idAircraft_2_285_aop">
                      <ulexlib:SameAsPayloadReference
ulexlib:nvref="Equipment idAircraft 2 285 C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                      <aop:PlannedPrimaryConfiguration>BEST</aop:PlannedPrimaryConfiguration>
                    </aop:Aircraft>
                    <aop:Routes ulexlib:id="Routes_idAircraftSortie_2_273_aop">
                      <aop:Route ulexlib:id="idRoute_2_291">
                        <ulexlib:SameAsPayloadReference ulexlib:nvref="Route_idRoute_2_291_C2
Core" ulexlib:pnvref="C2 CorePayload"/>
                        <aop:RoutePoints ulexlib:id="RoutePoints_idRoutePoints_2_292_aop">
                          <aop:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_293_aop">
                            <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePoint_idRoutePoint_2_293_C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                            <aop:RoutePointTypeCode>TKF</aop:RoutePointTypeCode>
                            <aop:RoutePointLocation
ulexlib:id="RoutePointLocation_idRoutePoint_2_293_aop">
                              <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePointLocation_idRoutePoint_2_293_C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                              <aop:LocationTypeCode>BASE</aop:LocationTypeCode>
                              <aop:LocationId>NENWNjVlZmdoaWprbG1ub3BxcnN0dXZ3</aop:LocationId>
                              <aop:SourceSystem>TBMCS</aop:SourceSystem>
                              <aop:SourceNode>DAVIS MONTHAN</aop:SourceNode>
                              <aop:ICAO>CV65</aop:ICAO>
                            </aop:RoutePointLocation>
                            <aop:Callsigns>
                              <aop:Callsign>
                                <aop:CallWord>LION</aop:CallWord>
                                <aop:CallNumber>02</aop:CallNumber>
                                <aop:AOR>DAVIS MONTHAN</aop:AOR>
                                <aop:ABPId>ACM2</aop:ABPId>
                                <aop:ATOMissionNumber>1801</aop:ATOMissionNumber>
                                <aop:IFFSIFs>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>1</aop:Mode>
                                    <aop:Code>00</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>2</aop:Mode>
                                    <aop:Code>2122</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>3</aop:Mode>
                                    <aop:Code>2122</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                </aop:IFFSIFs>
                              </aop:Callsign>
                            </aop:Callsigns>
                          </aop:RoutePoint>
                          <aop:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_354_aop">
                            <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePoint idRoutePoint 2 354 C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                            <aop:RoutePointTypeCode>TGT</aop:RoutePointTypeCode>
                            <aop:RoutePointLocation
ulexlib:id="RoutePointLocation_idRoutePoint_2_354_aop">
                              <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePointLocation_idRoutePoint_2_354_C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                              <aop:LocationTypeCode>GEOGRAPHICAREA</aop:LocationTypeCode>
                              <aop:LocationId>TRAINING CAMP</aop:LocationId>
                              <aop:SourceSystem>TBMCS</aop:SourceSystem>
                              <aop:SourceNode>DAVIS MONTHAN</aop:SourceNode>
                              <aop:ICAO></aop:ICAO>
                            </aop:RoutePointLocation>
                            <aop:Callsigns>
                              <aop:Callsign>
                                <aop:CallWord>LION</aop:CallWord>
                                <aop:CallNumber>02</aop:CallNumber>
                                <aop:AOR>DAVIS MONTHAN</aop:AOR>
                                <aop:ABPId>ACM2</aop:ABPId>
                                <aop:ATOMissionNumber>1801</aop:ATOMissionNumber>
                                <aop:IFFSIFs>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>1</aop:Mode>
```

```
                                    <aop:Code>00</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>2</aop:Mode>
                                    <aop:Code>2122</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>3</aop:Mode>
                                    <aop:Code>2122</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                </aop:IFFSIFs>
                              </aop:Callsign>
                            </aop:Callsigns>
                          </aop:RoutePoint>
                          <aop:RoutePoint ulexlib:id="RoutePoint_idRoutePoint_2_423_aop">
                            <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePoint_idRoutePoint_2_423_C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                            <aop:RoutePointTypeCode>LND</aop:RoutePointTypeCode>
                            <aop:RoutePointLocation
ulexlib:id="RoutePointLocation_idRoutePoint_2_423_aop">
                              <ulexlib:SameAsPayloadReference
ulexlib:nvref="RoutePointLocation idRoutePoint 2 423 C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                              <aop:LocationTypeCode>BASE</aop:LocationTypeCode>
                              <aop:LocationId>NENWNjVlZmdoaWprbG1ub3BxcnN0dXZ3</aop:LocationId>
                              <aop:SourceSystem>TBMCS</aop:SourceSystem>
                              <aop:SourceNode>DAVIS MONTHAN</aop:SourceNode>
                              <aop:ICAO>CV65</aop:ICAO>
                            </aop:RoutePointLocation>
                            <aop:Callsigns>
                              <aop:Callsign>
                                <aop:CallWord>LION</aop:CallWord>
                                <aop:CallNumber>02</aop:CallNumber>
                                <aop:AOR>DAVIS MONTHAN</aop:AOR>
                                <aop:ABPId>ACM2</aop:ABPId>
                                <aop:ATOMissionNumber>1801</aop:ATOMissionNumber>
                                <aop:IFFSIFs>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>1</aop:Mode>
                                    <aop:Code>00</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>2</aop:Mode>
                                    <aop:Code>2122</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                  <aop:IFFSIFModeAndCode>
                                    <aop:Mode>3</aop:Mode>
                                    <aop:Code>2122</aop:Code>
                                  </aop:IFFSIFModeAndCode>
                                </aop:IFFSIFs>
                              </aop:Callsign>
                            </aop:Callsigns>
                          </aop:RoutePoint>
                        </aop:RoutePoints>
                        <aop:RouteLastModified>
                          <aop:SourceSystem>TBMCS</aop:SourceSystem>
                          <aop:SourceNode>DAVIS MONTHAN</aop:SourceNode>
                          <ucore:TimeInstant>
                            <ucore:Value>2009-02-25T16:06:04.000Z</ucore:Value>
                          </ucore:TimeInstant>
                        </aop:RouteLastModified>
                      </aop:Route>
                    </aop:Routes>
                  </aop:AircraftSortie>
                </aop:Sorties>
                <aop:MissionTasks ulexlib:id="idMissionTasks_2_491">
                  <aop:MissionTask ulexlib:id="idMissionTask_2_492">
                    <ulexlib:SameAsPayloadReference
ulexlib:nvref="MissionTask idDirectAttack 2 493 C2 Core" ulexlib:pnvref="C2 CorePayload"/>
                    <aop:AssignedPriority>2</aop:AssignedPriority>
                    <aop:RequestIds>
                      <aop:RequestId>REQ4</aop:RequestId>
                    </aop:RequestIds>
                    <aop:MissionTaskLocation ulexlib:id="idLocation_2_516">
```

```
                      <ulexlib:SameAsDigestReference
ulexlib:nvref="MissionTaskLocation_idDirectAttack_2_493"/>
                      <aop:LocationSourceSystem>TBMCS</aop:LocationSourceSystem>
                      <aop:LocationSourceNode>DAVIS MONTHAN</aop:LocationSourceNode>
                    </aop:MissionTaskLocation>
                    <aop:Target ulexlib:id="idTarget 2 573">
                      <ulexlib:SameAsPayloadReference ulexlib:nvref="Target_idTarget_2_573_C2
Core" ulexlib:pnvref="C2 CorePayload"/>
                      <aop:JDPI>TC1</aop:JDPI>
                      <aop:JDPIName>TRAINING CAMP</aop:JDPIName>
                      <aop:ComponentTargetId>TC1</aop:ComponentTargetId>
                      <aop:Facility>TRAINING CAMP</aop:Facility>
                    </aop:Target>
                  </aop:MissionTask>
                </aop:MissionTasks>
              </aop:Mission>
            </aop:Missions>
          </aop:AO COI UCore C2 Core_Payload>
        </ulex:StructuredPayload>
        <ucore:Narrative>A message translated from CMD to C2 Core</ucore:Narrative>
      </ulex:DataItemPackage>
    </ulex:PublishMessage>
  </ulex:PublishMessageContainer>
</ulexpd:doPublish>
```

Appendix E Acronyms

ACTM	Aircraft Collection Tasking Message
AF	Air Force
AO	Air Operations
AOP	Air Operations
API	Application Programming Interface
ATO	Air Tasking Order
C2	Command and Control
CBRN	Chemical. Biological, Radiological, Nuclear
CMD	Common Mission Definition
COI	Community of Interest
CPM	C2 Portfolio Manager
CRD	Common Route Definition
DDMS	DoD Discovery Metadata Specification
DoD	Department of Defense
DOM	Document Object Model
GCIC	Global Cyberspace Integration Center
IES	information exchange specification
JC2	Joint Command and Control
MTF	Message Text Format
MTR	Mission Task Request
NECC	Net Enabled Command Capability
NIEM	National Information Exchange Model
PASS	Publish and Subscribe Services
RSS	Rich Source Summary
SAR	Suspicious Activity Reporting

SAX	Simple API for XML
SSA	Shared Situational Awareness
SSATF	Shared Situational Awareness Tracks Framework
TBMCS	Theatre Battle Management Command System
TFTSWG	Technical Framework Architecture and Tools Sub-Working Group
ULEX	Universal Lexical Exchange
UML	Unified Modeling Language
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

