MISB MOTION IMAGERY STANDARDS BOARD	MISB RP 1204
Recommended Practice	**June 7th 2012**
Motion Imagery Identification System (MIIS)	

1 Scope

Motion imagery data is generated by many different sensors, distributed across many different networks and received by many different users and systems. With a large number of motion imagery sources it is important to define a consistent name or identity for each source.

This recommended practice (RP) defines the required identification elements that shall be inserted into motion imagery streams or files, where in the motion imagery data flow (i.e. from source to user) the insertions should happen and the insertion techniques, formats and locations within streams and files. This RP also defines how to construct UUID's for use as enterprise identifiers of Motion Imagery System.

In addition to the required identifiers there are many supplemental identifiers which can be included in the metadata for varying purposes. This recommended practice provides a list of valid identifiers along with an area to insert these identifiers.

The Motion Imagery Identification System (MIIS) works towards solutions to four related problems:

(1) Identifying whether two (or more) streams or motion imagery files are derived from the same source.
(2) Distinguishing when two (or more) streams or motion imagery files are distinct, even if they derive from the same source data.
(3) Establishing the "chain of custody" of a stream or motion imagery file.
(4) Adding useful identifying information about the stream or motion imagery file.

This main body of this document describes the recommended practices with a particular focus on problems (1) and (4).

2 References

2.1 Normative References

[1] ISO/IEC 8825-1:2008, Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)

[2] ITU-T X.667 - *Generation and registration of Universally Unique Identifiers (UUIDs)*
[3] ISO/IEC 14977:1996, Information Technology - - Syntactic meta-language – Extended BNF

2.2 Informative References

[4] MISB STD 0604.2 - Time stamping and Transport of Compressed Motion Imagery and Metadata , June 2011
[5] NSG Metadata Foundation (NMF) Part 1 Version 2.1
[6] ISO/IED 14496-10:2012 – Information Technology – coding of Audio-Visual Objects Part 10: Advanced Video Codec
[7] SMPTE 336M-2007 "*Data Encoding Protocol Using Key-Length-Value*"
[8] XML Schema - http://www.w3.org/XML/Schema

3 Modifications and Changes

- **RP 1204** – Initial Release - **June 4th 2012**

4 Definitions and Acronyms

FMV Full Motion Video

WAMI Wide Area Motion Imagery

5 Introduction/Background - Informative

With motion imagery systems, FMV and WAMI data is generated by many platforms and disseminated to many systems as illustrated in Figure 1 below.

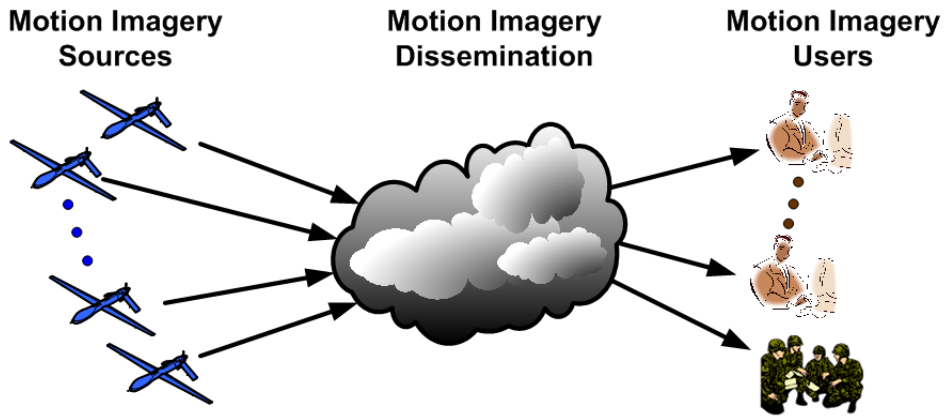

Figure 1 – Conceptual Data Flow

Figure 1 shows the typical data flow which is composed of three components, the data sources (e.g. UAV), the dissemination cloud (e.g. DISA) and the data receivers or users (e.g. NGA).

Users need to know the identity of the data source they are working with plus any supporting contextual information. This suggests that two types of identifiers are needed: **Core identifiers** and **Augmentation identifiers**. The purpose of a **Core identifier** is to provide a unique "name" for a motion imagery dataset; core identifiers are required in every motion imagery dataset (stream or file). The purpose of an **Augmentation identifier** is to provide the contextual information which is information about the data source and its usage (e.g. Mission IDs, ATO's, Feed color).Augmentation identifiers are not required but recommended.

5.1 Core Identifiers

There are two types of core identifiers (see Appendix A for rationale): **Absolute** and **Relative**. **Absolute** identifiers are generated when the motion imagery is first imaged and are accessible and usable by all users of the motion imagery data. **Relative** identifiers are generated further "downstream" from the motion imagery source and provide an identifier for a limited set of users – i.e. they are relative for a particular sub-group of users (e.g. a specific site or organization). Relative identifiers are only needed when absolute identifiers have not been defined for a given motion imagery dataset (stream/file); they are needed because not all systems are generating absolute identifiers (such as a legacy system).

Absolute Core identifiers have several important properties:

1) **Unique** – no two identifier values are the same amongst any sensor and/or platform
2) **Compact** – the identifiers are repeated in the stream frequently, so the identifier needs to be small to keep bandwidth low
3) **Consistent** – the same identifier is used after a system has been powered off and on
4) **Automatic** – the identifier is not dependent on human intervention for assignment
5) **Meaningful**– random numbers provide uniqueness but uniqueness can be achieved with other formats (discussed below) which provide additional value.

Relative Core Identifiers have a subset of these properties:

1) **Unique** – no two identifier values are the same amongst any sensor and/or platform
2) **Compact** – the identifiers are repeated in the stream frequently so the identifier needs to be small to keep bandwidth low
3) **Automatic -** the identifier value is not dependent on human intervention for assignment

5.2 Augmentation Identifiers

Augmentation identifiers provide further context (refinement) about the motion imagery dataset. Augmentation identifiers can be added by any system throughout the data flow. For example the systems can use third party databases to reference and cross correlate information for insertion into the data stream for downstream reference. The augmentation data is informational only – it is not required for basic identification.

5.3 Unique Identifier for Frames

To generate a unique identifier for a motion imagery frame the Core Identifier is combined with the MISP STD 0604 [4] time stamp. When an Absolute Core Identifier is used this technique

provides a unique identifier for every frame across all motion imagery data sources. Example for a motion imagery file where each frame's identifier is created from Core ID + 0604 Timestamp:

Table 1 - Unique Identifier for Motion Imagery Frames

Frame	Absolute Id	0604 Timestamp
Frame 1	A:050F-8148-0002-0301-01:14	2012-08-20T13:23:13.**134**000
Frame 2	A:050F-8148-0002-0301-01:14	2012-08-20T13:23:13.**167**000

6 Since the Absolute Identifier is unique across all platforms and the timestamp provides a unqiue id for each frame the combination of the two provides a universally unique frame id for any systems. Standard Description - Normative

This section provides the required and recommended implementation details for the Motion Imagery Identification System (MIIS). Section 6.1 describes the MIIS by using Extended Backus–Naur Form (EBNF) (defined in ISO/IEC 14977 [3]). By using EBNF the identification information is described independently from the transmission format. In section 6.1 the shaded text highlights the EBNF.

Section 6.2 supplies the details for how to represent the EBNF structure in different formats including KLV (Key-Length-Value) [7], XML (Extensible Markup Language) [8], SEI (Supplemental Enhancement Information [6]) message, and text version.

6.1 Identification Information Description

The MIIS has two parts, the first part is the required Core Identifier and the second part is a list of optional Augmentation Identifiers. The format for the whole MIIS is:

MIIS = Core Identifier, {Augmentation Identifier}

Where:
Core Identifier = defined below in section 6.1.1
Augmentation Identifier = defined below in section 6.1.2

6.1.1 Core Identifiers

There are two types of core identifiers: **Absolute Identifiers** (AID) and **Relative Identifiers** (RID). **Absolute identifiers** contain meaningful information about the source and are only constructed as part of the sensor collection process. Relative identifiers do not contain meaningful information and only provide a random unique ID for the motion imagery data. Relative identifiers are only **created** downstream from the source and only if an absolute identifier does not exist within the motion imagery. Thus, relative identifiers are only **used** in the absence of an AID in downstream processing.

Core Identifier = Core ID Type, AID | RID
Where:
Core ID Type = Indication of which type of Core identifier is being used: 'AID' or 'RID'.
AID = Absolute ID defined in section 6.1.1.1.
RID = Relative ID defined in section 6.1.1.2.

6.1.1.1 Absolute ID (AID)

The AID is created by combining device identifiers for the sensor and platform with an optional window identifier (explained below). The device identifier is a unique value that is created for a specific hardware device, i.e. the physical sensor or physical airframe, etc. The format of the device identifier is described in Appendix B.

AID = Sensor ID, [Platform ID], [Window ID]
Where:
Sensor ID = Device ID for the sensor as defined in Appendix B. <u>This id is required.</u>
Platform ID = Device ID for the platform as defined in Appendix B. This id is not required but recommended.
Window ID = A 16 byte Universal Unique ID (UUID). This ID is only needed when creating windows from motion imagery data.

The Sensor ID shall identify the device that is converting some optical (or other) phonomenon to a digital signal. For example the Sensor ID should be linked to a focal plane array and its electronics; however, the Sensor ID is **not** the identifier for the gimbal ball or lens system (identifiers for these devices could be added as augmentation identifiers).

The Platform ID shall identify the physical device that the sensor is connected to (i.e. aircraft, UAV, Humvee, etc). The Platform ID is not required but recommended if a platform is used; for example a handheld camera does not have a platform so the Platform ID is not required.

The Sensor ID and Platform ID can be added to the AID at different times: for example, the sensor may create the AID with just sensor information, and then the platform would add the Platform ID. There are many different configurations: the important requirement is that when the motion imagery data leaves the platform that the AID be fully constructed.

6.1.1.2 The Window ID identifies a "virtual" window that may be extracted from a motion imagery source. This ID is only needed when sub-images of a motion imagery data set are being created and/or transmitted. For example a WAMI data source may have multiple users extracting virtual windows from the large frame data – each frame would have identical Sensor ID and Platform ID so a Window ID is required to enforce the uniqueness requirement. Relative ID (RID)

The RID is created by generating a 16 byte Universal Unique ID (UUID) [2]. Relative identifiers are only **created** downstream from the source if an absolute identifier (AID) does not exist within the motion imagery. Therefore, relative identifiers are only **used** in the absence of an AID in downstream processing.

RID = UUID

6.1.2 Augmentation Identifiers

Augmentation Identifiers are name and value pairs of information. The name identifies the contextual information value that is being specified. The value can be of any type string, float, integer, etc. The format of an augmentation identifier is:

Augmentation Identifier =Name, Value
Where:
Name = an identifier for the value (e.g. Air Tasking Order)
Value= value for the identifier (e.g. ATO12345)

6.1.3 MIIS Summary

The following shows the complete EBNF for MIIS

MIIS = 'AID' | 'RID', (Sensor Id, [Platform Id], [Window Id]) | (UUID) {(Name, Value)}
Where:
Core Identifier Type = 'AID' or 'RID'
Core Identifier = (Sensor Id, [Platform Id], [Window Id]) or (UUID)
Augmentation Id List = {(Name, Value)}

6.2 MIIS Formats

There are several different formats used to represent MIIS information:

Table 2 - MIIS Formats

Format Name	Section	Description
KLV	6.2.1	Used within the metadata section of a motion imagery stream or file.
XML	6.2.2	Used when communicating identity information outside of motion

		imagery data stream or file (i.e. system to system).
SEI Message	6.2.3	Used to embed the Core Identifier into a H.264 Video Elementary stream/file
Textual Format	6.2.4	Used when specifying a Core Identifier for human use.

6.2.1 Formatting MIIS information as KLV

When formatting the MIIS data as KLV all of the data is formatted in a local data set with key 06.0E.2B.34.02.0B.01.01.0E.01.03.05.03.00.00.00 (CRC 47531, symbol of "miis_lds").

The following table describes the mapping between the MIIS EBNF to MIIS KLV:

Table 3 - MIIS EBNF to MIIS KLV Mapping

EBNF	KLV Construct
Core Identifier Type	Defined by the tag used for core identifier (tag 1 or 2)
Core Identifier	Core ID Tag Value – binary format for either AID or RID value
Augmentation ID List	List of Tags and values listed after the Core ID (note freeform values based on identifier)
Other notes	This LDS also contains a version number

The following table lists the Core Identifier tags for the LDS, along with the version key and some reserved tag space; the remaining augmentation identifiers are listed in Appendix C.

Table 4 - MIIS LDS Tags/Keys

Tag	Name / Symbol	KLV Key	Description
1	AID absolute_id	06.0E.2B.34.01.01.01.01 0E.01.04.05.01.00.00.00 (CRC 39712)	Absolute Identifier - combination of Sensor ID, Platform ID and optional Window ID.
2	RID relative_id	06.0E.2B.34.01.01.01.01 0E.01.04.05.02.00.00.00 (CRC 00252)	Relative Identifier - UUID. Only used if AID is not generated.
3	Version document_version	06.0E.2B.34.01.01.01.01 0E.01.02.05.05.00.00.00 (CRC 56368)	Version of this document – BER OID integer. Only used if version increments beyond version 0.
4	Reserved	Reserved	Reserved
5	Reserved	Reserved	Reserved
6	Reserved	Reserved	Reserved
7	Reserved	Reserved	Reserved

Figure 2, shows an example of a MIIS Local Data Set (LDS). This illustration shows the LDS with an absolute identifier, plus the version and an augmentation list.

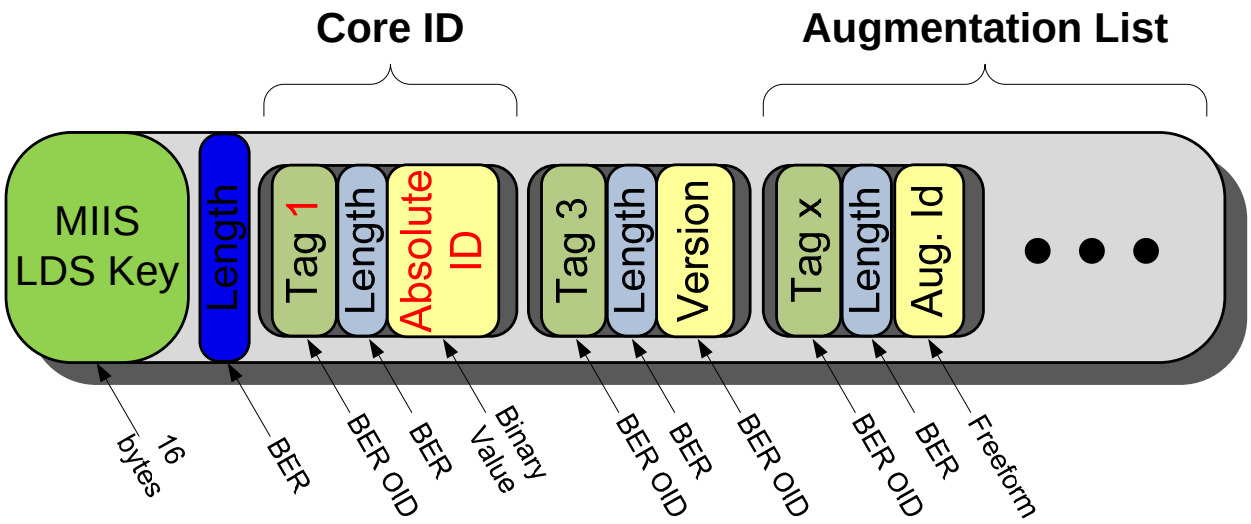

Figure 2 - MIIS LDS

6.2.2 Formatting MIIS information as XML

Table 5 describes the mapping between the MIIS EBNF to MIIS XML

Table 5 - MIIS EBNF to MIIS XML

EBNF	XML Construct
Core Identifier Type	Defined by the XML tag used when specifying the core identifier (<AID> or <RID>)
Core Identifier	Core ID Value as a hex value
Augmentation Id List	List of XML tags with names defined along with values
Other notes	This XML data also contains a version number

When formatting the MIIS information as XML the following schema is used:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org/NewXMLSchema"
xmlns:tns="http://www.example.org/NewXMLSchema" elementFormDefault="qualified">

<xs:complexType name="miis">
  <xs:sequence>
    <xs:element name="core_id" type="tns:core_id"/>
    <!-- version should be only >=0 -->
    <xs:element name="version" type="xs:integer"/>
    <xs:element name="augmentation_list" type="tns:augmentation_item" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>


<xs:complexType name="core_id">
```

```
  <xs:choice>
    <xs:element name="aid" type="tns:aid"/>
    <xs:element name="rid" type="tns:rid"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="aid">
  <xs:sequence>
    <xs:element name="sensor_id" type="tns:device_id"/>
    <xs:element name="platform_id" minOccurs="0" type="tns:device_id"/>
    <!-- window id is added only if a platform id is added -->
    <xs:element name="window_id" minOccurs="0" type="tns:uuid"/>
  </xs:sequence>

</xs:complexType>

<xs:complexType name="rid">
  <xs:sequence>
  <xs:element name="window_id" type="tns:uuid"/>
  </xs:sequence>
</xs:complexType>


<xs:complexType name="augmentation_item">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="value" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="uuid"  >
  <xs:restriction base="xs:string">
    <xs:pattern
      value="[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}">
    </xs:pattern>
  </xs:restriction>
</xs:simpleType>


<xs:complexType name="device_id">
  <xs:sequence>
    <xs:element name="manufacture_num" type="tns:serial_num"/>
    <xs:element name="model_num" type="tns:serial_num"/>
    <xs:element name="serial_num" type="tns:serial_num"/>
    <xs:element name="version_num" type="tns:serial_num"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="serial_num">
  <xs:restriction base="xs:integer">
```

```
  <xs:minExclusive value="0"/>
 </xs:restriction>
</xs:simpleType>

</xs:schema>
```

The list of augmentation identifier names and types are listed in Appendix C.

6.2.3 Formatting Core Identifier as SEI message

As discussed in the policy section the Core Identifier shall be inserted into an H.264 elementary stream by using an SEI message. (The MISB is discouraging MPEG-2 in favor of H.264. Please contact the MISB for guidance with respect to implementation in MPEG-2)

The H.264 standard [6] allows user defined data that is associated with a particular compressed frame of imagery to be inserted into the elementary stream using the "user data unregistered" SEI Message field (*user_data_unregistered*). The Core Identifier shall be inserted into the H.264 elementary stream *user_data_unregistered* SEI Message field, so that it relates to a specific frame of imagery.

The *user_data_unregistered* SEI message shall consist of two sub-fields: 1) an identification string and 2) a Core Identifier. This composite field will be comprised of a variable number of total bytes based on the length of the core identifier.

- Identification String: The uuid_iso_iec_11578 is a 16-byte field (value set as a UUID) that represents the ASCII string "MISPMIISCOREIDNT" and shall be set to the values as specified in Table 6.

Table 6 - SEI Message Identification String

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
M	I	S	P	M	I	I	S
0x4D	0x49	0x53	0x50	0x4D	0x49	0x49	0x53
Byte 9	**Byte 10**	**Byte 11**	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**
C	O	R	E	I	D	N	T
0x43	0x4F	0x52	0x45	0x49	0x44	0x4E	0x54

- Core Identifier: The *user_data_payload_byte* is a variable length field with the same format and is set to the values as specified in :

Table 7 - SEI Message Core Identifier

Byte	Value
17	Use 0x01 for Absolute identifier or 0x02 for Relative identifier
18,19	Byte 1 and 2 (most significant bytes) of Absolute or Relative ID Value

20	Start Code Emulation Prevention byte (always 0xFF)
21,22	Byte 3 and 4 of Absolute or Relative ID Value
23	Start Code Emulation Prevention byte (always 0xFF)
…	<Repeat pattern, two bytes of value followed by 1 byte of start code emulation prevention byte>
Last	Start Code Emulation Prevention byte (always 0xFF)

6.2.4 Text Format (Human Readable)

When communicating Core Identifiers via text three values will be supplied: Core Id Type, Core ID Hex value and Check byte. The check byte is included so that values can be quickly validated when being hand entered or transcribed. The following format shall be used:

Core Id Type	Core Id Hex Value	Check Byte
Single character A for absolute or R for relative, followed by a colon, i.e. A: or R:	AID hex value or RID hex Value formatted in groups of four characters separated by a dash, i.e. 0102-0304-0506-0708-090A-0B0C	Check sum value computed as described below. Formatted as a single hex value prefixed by a colon, i.e. :3F

Examples		
Absolute ID	A:050F-8148-0002-0301-01:14	
Relative ID	R:550e-8400-e29b-41d4-a716-4466-5544-0000:79	

The check sum byte is computed by adding each byte value together and only keeping the final byte of the sum (Least Significant Byte). For Absolute Id example above (all hex values): 05+0F+81+48+00+02+03+01+31 = 0x0114 so 0x14 would be the check sum.

7 Generating Enterprise UUID's from Core Identifiers

Core Identifiers will be used for many different purposes including as the identifier within enterprise systems however, to be consistent with enterprise standards [5] the identifier must be converted into a UUID. As discussed in section 6 the Core Identifier can be either an AID or RID each of which has their own technique for generating an Enterprise UUID (EUUID). The EUUID for an AID based Core Identifier shall be constructed using the "name based UUID" technique from X.667 [2] (section 14), specifically the SHA-1 subclause (section 14.3). The EUUID for an RID based Core Identifier shall constructed by using the UUID from the RID (i.e. the RID = EUUID).

In addition to generating a EUUID for the Core Identifier, EUUIDs can be constructed for the sensor and/or platform identifiers independently of the Core Identifier. The EUUID for the **Sensor ID** shall be constructed using the "name based UUID" technique from X.667 [2] (section 14), specifically the SHA-1 subclause (section 14.3). The EUUID for the **Platform ID** shall be constructed using the "name based UUID" technique from X.667 [2] (section 14), specifically the SHA-1 subclause (section 14.3).

8 Implementation Policy

The implementation policy is governed by the following set of rules:

1. All Sensors shall insert the Absolute Core Identifier into the motion imagery data. This includes FMV sensors and WAMI sensors.
2. All Platforms shall add the platform identifier to the Absolute Core Identifier if a platform exists for the sensor (e.g. hand held camera does not have a platform so no platform id is required).
3. Sensors and/or Platforms are the only devices allowed to generate Absolute Identifiers.
4. Sensors and/or Platforms shall NOT generate Relative Identifiers.
5. Relative Identifiers shall only be created and inserted by devices downstream of the sensor/ platform.
6. Relative Identifiers shall only be created and used if Absolute Identifiers are not defined in the stream.
7. Absolute Identifiers and Relative Identifiers shall NOT be allowed in the same motion imagery data file or stream.
8. For streaming formats (e.g. H.264 compression over MPEG-2 Transport Streams)
    1. The MIIS Identifier shall be inserted into the stream's metadata at least once every 30 seconds,
    2. The MIIS Identifier shall be inserted into the stream's metadata whenever its Core Identifier value changes (e.g., if a sensor changes from EO to IR, then a different Core Identifier is used),
    3. The MIIS Core Identifier shall be inserted into the stream's video elementary stream once every frame (as with the 0604 time stamp). Please note that this document only provides instructions for inserting the Core Identifier into H.264 Elementary streams because the MISB is discouraging the use of MPEG-2 Elementary Streams. Please contact the MISB for guidance with respect to implementation in MPEG-2 elementary streams.
9. For non-streaming formats (e.g. NITF wrapped JPEG2000 or other WAMI formats)
    1. The MIIS Identifier shall be inserted into or along with each frame of the motion imagery data.

Appendix A Core Identifier Background

Figure 3 shows a single motion imagery data source being distributed to a group of users.

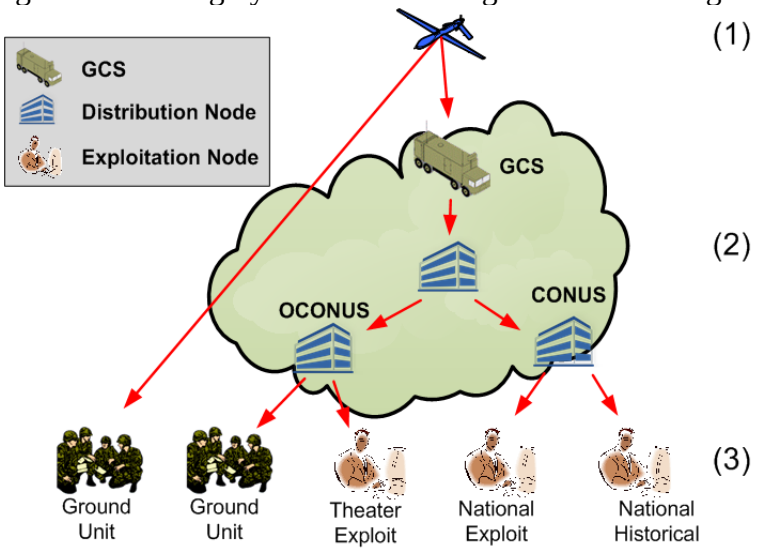

Figure 3 - Motion Imagery Data Flow Scenario

Identification information can be inserted into the motion imagery data at several points: (1) the source, (2) the distribution cloud or at (3) the end user sites. Ideally, the optimal point to insert identification information is at the source (1), because the identifier(s) will be available (i.e. flow down) to all the following components of the overall system (i.e., distribution cloud and end users sites). If identifiers have not been inserted at the source then the next best place to insert identification is within the distribution cloud (2); however, the usefulness of this varies depending on which "part" of the cloud the identifiers are inserted. If the identifiers are inserted right after receiving the video data (i.e., at the GCS(2)) then this provides a consistent identifier for all downstream dissemination. If the identifiers are inserted just before leaving the dissemination cloud, then there is no guarantee that all copies of the motion imagery data leaving the cloud will have the same identifiers, which, does not provide the consistency desired. If neither the source (1) nor dissemination cloud (2) inserts an identifier, then the user sites (3) can insert their own identifiers; however, each site would have different identifiers unless there is some extra backchannel coordination performed with between sites.

Identification **coverage** is defined as the percentage of users that are able to access or use a core identifier. The goal is to provide an identifier that has 100% or *absolute* coverage so that it is available to all users of the motion imagery data.

> *As an example, Figure 3 shows that if identifiers are inserted at the "top" within the source then all users at the bottom of the diagram would be receiving the same identification information, providing 100%, or absolute, coverage of the end users. If the identifiers were inserted at the GCS, then a majority of users would be receiving the identification information providing 80% coverage. If the*

identifiers were inserted at the CONUS or OCONUS distribution sites then there is only 40% coverage. If an identifier is inserted at any of the user sites then only 20% coverage is obtained. If there is some form of coordination between sites then higher coverage can be obtained, however there is no guarantee that all data would ever be labeled with an identifier.

Figure 3 also can be used to illustrate a problem with the value to insert as the core identifier. As discussed previously, if the identifier is not created at the source then the identifier cannot provide *absolute* coverage. When a non-source component inserts a core identifier it is not an *absolute* identifier, but a *relative* identifier which will have less coverage. Other non-source components could insert different *relative* identifiers. For example, in Figure 3, if the OCONUS component inserts a *relative* identifier independently from the CONUS component, each would be inserting a different *relative* identifier and then theater exploiters (or systems) and national exploiters (or systems) would not be able to recognize that they may be working on the same stream (without using other information for correlation). This means that there will need to be two classes of identifiers (a) *absolute* (source) identifiers and (b) *relative* (non-source) identifiers. With the two types of identifiers downstream users or systems can determine if they know exactly which stream/file they are working with (*absolute*), or if further coordination is needed (*relative*).

At the motion imagery source there are several options for providing a unique identifier. Appendix D illustrates a couple of source architectures. It is clear that there is no one single piece of information that can be used as an identifier, so a collection of identifiers is used as the source identifier. Ideally, every device that motion imagery passes through should add identifying information to the stream creating an “ID history”; however this is impractical, since it would imply that every device would have to de-mux, update and re-mux the motion imagery stream.

This recommended practice defines a minimum set of elements such that the identifiers still provide uniqueness and a basis for augmentation data. The resulting minimum identifier is a sensor identifier, combined with a platform identifier; these two pieces of information uniquely identify a source. However, when a sensor system, such as a WAMI sensor, is produces one or more “virtual” windows from a motion imagery source, an additional identifier must be added called a window identifier. These two or three pieces of identifying information form the Absolute Core ID.

The Absolute Core ID can change over time within a mission. For example when the sensor source is changed (i.e. EO to IR) the sensor component of the ID will change; however, the platform component of the identifier remains the same. It is up to the receivers of the data to interpret these values and determine how to process and use them. This identifier along with the MISP STD 0604 timestamp will enable frame by frame unique identifiers that can be used as a basis for augmented identifiers.

Relative identifiers cannot be generated from system data so a unique random number is created insted. Specifically, the Universal Unique Identifier is used (UUID).

In summary, the only condition that *absolute* identification coverage can be guaranteed is when the identifiers are inserted at the source. Unfortunately, currently deployed systems do not have this capability so downstream or *relative* identification methods have to be used until such time that the platforms/sensors provide *absolute* identification information. Furthermore, there must be a distinction between *absolute* (source) identifiers versus *relative* (non-source) identifiers so that users can properly understand the source of the identifier.

Appendix B Device Identifiers - Normative

The purpose of the device identifier is to provide a well-defined, well-formatted, unique identifier for a manufactured hardware device. This identifier is constructed with a unique number to be assigned by the MISB and all remaining numbers to be defined by a manufacturer with the manufacturer ensuring uniqueness for every device. Additionally this identifier is required to be "power cycle" safe – meaning that if the power is shut off and turned back on the exact same identifier is always produced for that device. The one exception is the version # can be incremented if an update (software or hardware) to the device is made; updating the version number will still provide an overall unique device identifier. A device identifier is defined as:

Device ID = <Manufacturer Number>, <Model Number>, <Serial Number>, <Version Number>

Where each"Number" is a BER OID (positive integer) [1] value defined in Table 8:

Table 8 - Device Identifier Definitions

Name	Description	Example
Manufacturer #	A number that represents the manufacturer; and issued by the MISB. Note: there are reserved values in this list – see Table 10.	ACME = 5
Model #	A number representing the model number of the device – this must be unique for each device model and is assigned by the manufacturer.	IR Sensor Model IRIS Model number = 15
Serial #	A number representing the serial number of the device – this must be unique for each device and is assigned by the manufacturer.	Serial Number = 200
Version #	A number representing the version of the device – this is assigned and maintained by the manufacturer.	Version Number = 0

These numbers are combined into one value to represent a unique identifier for a single device.

Table 9 - Example of Device Identifier

	Manufacturer #	Model #	Serial #	Version #
Value	5	15	200	0
Hex Value	0x05	0x0F	0xC8	0x00

BER OID Value	0x05	0x0F	0x8148	0x00
Final Value	0x050F 8148 00 (total of 5 bytes)			

This value would be generated by the device when requested, during device startup or during normal operation.

The minimum size for the Device Identifier is 4 bytes; there is no maximum size but it is not expected to exceed 2 bytes for the manufacturer (16,384 manufacturers), 2 bytes for the model number (16,384 device models for that manufacturer), 3 bytes for the serial number (268 Million devices of one model), and one byte for the version number (128 versions of one model) – for a total expected maximum of 8 bytes.

The MISB is reserves the following manufacture numbers shown in Table 10 for future needs.

Table 10 - Reserved Manufacture Numbers

Manufacturer #	Owner	Purpose
0x00	MISB	Reserved
0x01	MISB	Reserved
0x02	MISB	Reserved
0x03	MISB	Reserved
0x04	MISB	Reserved

Appendix C Augmentation Identifiers

Table 11 lists the valid augmentation identifiers for this RP. LDS keys 1 through 7 are used by core identifiers and are defined in section 6.2.1.

Note: At this time there are no Augmentation Identifiers officially defined. The table below shows only examples.

Table 11 - Example Augmentation Identifiers

LDS Tag	KLV Key	Name (XML Name)	Type	Description
8	TBD	Tail_ Number	String	Tail number of aircraft
9	TBD	ATO	String	Air tasking order

Appendix D Architecture Overview of Platform Sensor

This appendix describes a general architecture for motion imagery systems along with several examples of the general architecture.

Figure 4 shows a generic architecture to support a single sensor and the data flow through the architecture. This figure shows four different types of items: data producers (blue), data processors/movers (yellow), data storage (orange) and users (green). Each item is described in Table 12.

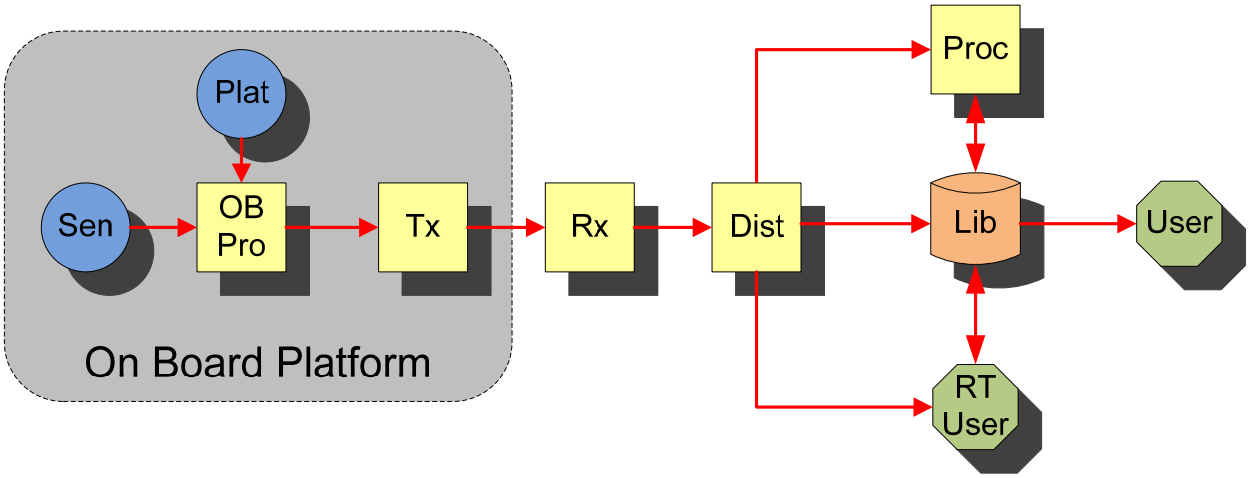

Figure 4 - Generic Motion Imagery Architecture

Table 12 – Architecture Components

Item	Name	Description
Sen	Sensor	Collects Motion Imagery (including metadata) from scene
Plat	Platform	Physically supports the sensor, produces metadata (e.g. orientation, etc.)
OBPro	On Board Processing	Processes Motion Imagery and Metadata onboard the platform. Processing examples: Encoder, Multiplexor, Switching, on board storage, etc.
Tx	Transmitter	Transmits Motion Imagery to one or more Receivers – can include encoding and/or transrating of motion imagery.
Rx	Receiver	Receives transmitted motion imagery – can include multiplexing additional metadata into motion imagery
Dist	Distribution	Dissemination of Motion Imagery through one or more networks. Can include switches, guards, SATCOMs, etc.
Lib	Library	Library storage of Motion Imagery data
Proc	Processing	Real-time and Post-processing of Motion Imagery data
RTUser	Real Time User	User exploiting Motion Imagery during mission. Operations include annotation, clipping, etc.
User	User	Non-Real time exploiting motion imagery mission

Motion Imagery systems contain at least one sensor but other items in this list may not be used. For example a hand held camera has a sensor and on board storage, the user could view the video directly on the back of the camera system; in this example the Tx, Rx, Dist, Lib, Proc are not needed. Another example is an airborne system which uses all items in the list/diagram. This generic architecture can be applied to all Motion Imagery systems use in Air, Ground, Water and Space.

To understand the complexities of the identification problem a couple of detailed examples are described below. Figure 5 illustrates a UAS architecture/data flow which uses two sensors and an A/B switch to determine which sensors data is transmitted to the receivers. Additionally, the selected stream is encoded with two different encoders, one for SATCOM transmission and one for line–of-site (LOS) transmission; each encoder may be operating using different data rates and qualities.

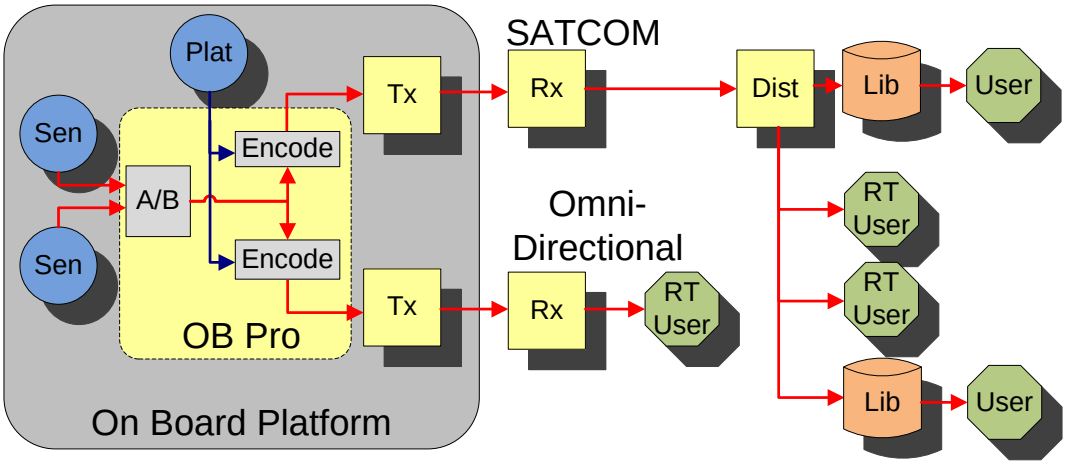

Figure 5 – UAS Dual Sensor with Dual Encoder

With this architecture inserting an identifier before the encoders is the only way to insure that the SATCOM users and LOS users both receive the same identifier. In this example a platform unique identifier would solve the identification problem since the system is focused on one mission; however, other examples show that this is not always the case.

Figure 6 illustrates an example of many sensors' data being switched (as needed) to a transmitter. Each sensor could be used for different missions or user needs.

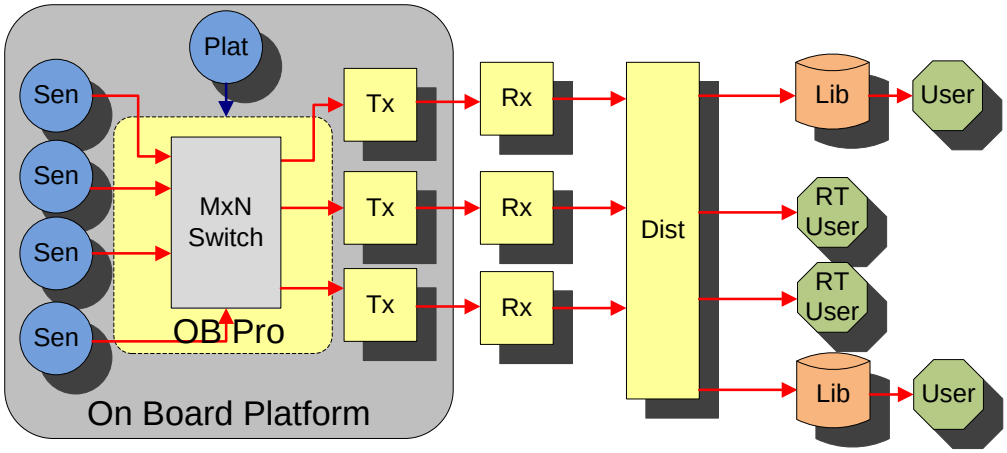

Figure 6 – Multi-Sensor Mulit-Tranmission

In this example the platform id would be insufficient and a more appropriate identifier would be to use a sensor id.