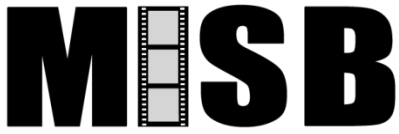
MOTION IMAGERY STANDARDS PROFILE

Motion Imagery Standards Board



MISP-2019.1: Motion Imagery Handbook

November 2018

Table of Contents

List of Figures

List of Tables

Change Log

2019.1	• Minor Editorial Change • Updated References [1], [14], [16], [23], [25]

Scope

The purpose of the Motion Imagery Handbook is to provide:

1. A definition of Motion Imagery.
2. Common terminology for all MISB documentation.
    a. There is no single authoritative source for technical definitions of terms within the community; therefore, the Motion Imagery Handbook serves as the authoritative source of definitions for the MISB community of practice.
3. Additional detail for topics identified in the Motion Imagery Standards Profile [1].
    a. The MISP succinctly states requirements, while the Motion Imagery Handbook discusses principles underlying requirements more thoroughly.

Many definitions and terms are used throughout the various commercial groups and vendors however many of these terms are either overloaded with conflicting meanings or there is disagreement about what a term means. The purpose of this document is to provide the MISB view of these definitions when these term definitions arise. The MISB has a "reference, clarify or define" philosophy when using term and definitions. When a term is well defined and accepted then we defer the definition to a formal external reference. When a term is not well defined due to overloaded use or disagreement then this document will clarify how the MISB will use the term within the MISP documents. When a term or definition is non-existent this document will provide the definition.

Although intended to be educational and informative the Motion Imagery Handbook is not a substitute for available material that addresses the theory of imaging, video/compression fundamentals, and transmission principles.

Organization

The Motion Imagery Handbook is composed of chapters, each emphasizing different topics that support the Motion Imagery Standards Profile (MISP) [1]. Each chapter is intended to be self-contained with references to other chapters where needed. Thus, a reader will be able to quickly locate information without reading preceding chapters. The Motion Imagery Handbook is expected to mature over time to include material considered essential in applying the requirements within the MISP as well as other MISB standards.

Chapter 1

Terminology and Definitions

1.1 Motion Imagery

Many different sensor technologies produce Motion Imagery. To support an imaging workflow in which different sensor data can be utilized by an Exploitation system, standards defining common formats and protocols are needed. The standards facilitate interoperable functionality, where different vendor products can readily be inserted within the workflow based on improvement and cost. Such standards need to be developed on a well-defined and integrated system of terminology. This section lays the foundation for this terminology.

1.1.1 Time

Time is fundamental in Motion Imagery. All events whether captured by a camera or artificially created are either formed over a period of time or are displayed over a period of time. In a camera, for instance, the light reflected or emitted from an object is exposed onto the camera's "sensor", which could be some type of imaging array or film. A **Start Time** and an **End Time** bounds the period of exposure. These are important qualifiers that play a significant role in image quality, particularly in capturing motion within the imagery. When exposure times are too long the motion is blurry; when exposure times are too short the motion may not be apparent.

A time-line is the continuous passage of time. An instant is an infinitesimal point on the time-line. Time is a measured offset between an instant and an absolute or relative reference on the time-line. Although these terms have various definitions in the literature, the MISB defines these terms here specific to their application in the MISB community. **Absolute Time** measures are an offset to a known universal source, such as International Atomic Time (TAI). **Relative Time** measures are an offset from some defined event. For example, Relative Time is a basis for overall system timing in the formatting of Motion Imagery, compression, and data transmission. Chapter 6 provides further discussion of time and Time Systems.

Start Time: Instant a process is initiated, measured in either Absolute Time or Relative Time.

End Time: Instant a process is completed, measured in either Absolute Time or Relative Time.

Absolute Time: An instant measured as an offset to a known universal source's (e.g. TAI) starting point, which is called an epoch.

Relative Time: An instant measured as an offset from a starting event.

1.1.2 Frame

The term Frame is commonly used in describing video and Motion Imagery, for instance, *image frame*, *frame size*, *frames per second*, etc. A **Frame** is defined as a two-dimensional array of regularly spaced values, called Pixels that represent some type of data – usually visual.

A **Pixel** is a *combination* of one or more individual numerical values, where each value is called a Sample. A **Sample** is data that represents a measured phenomenon such as light intensity.

In considering visual information, a Frame could be the data representing a monochrome (i.e. greyscale) or color picture. For example, with a monochrome picture, the Pixel values are the intensity values of light at each position in the picture. In a color picture, the Pixel data is composed of three different intensity values, i.e. red, green, and blue at each position in the picture.

An array of Samples where all phenomena are of the same type is called a **Band**. For example, a Band of Samples for the red component of color imagery contains only measurements of light sensitive to the "red" wavelength. For a monochrome picture, one Band is sufficient, whereas for color, three Bands are needed. A Frame can consist of Pixels combined from one or more Bands. Figure 1-1 illustrates the relationships of Samples and Bands to a Frame.

A **Pixel** is a *combination* of a number of Samples collectively taken from a number of Bands (see Figure 1-1). Where there is only one Band, a Pixel is equivalent to a Sample. A "color" Pixel is a combination of red, green and blue Samples from corresponding red, green and blue Bands. In Chapter 4 various color models where the relationship between Pixels and Samples are not one-for-one are discussed.

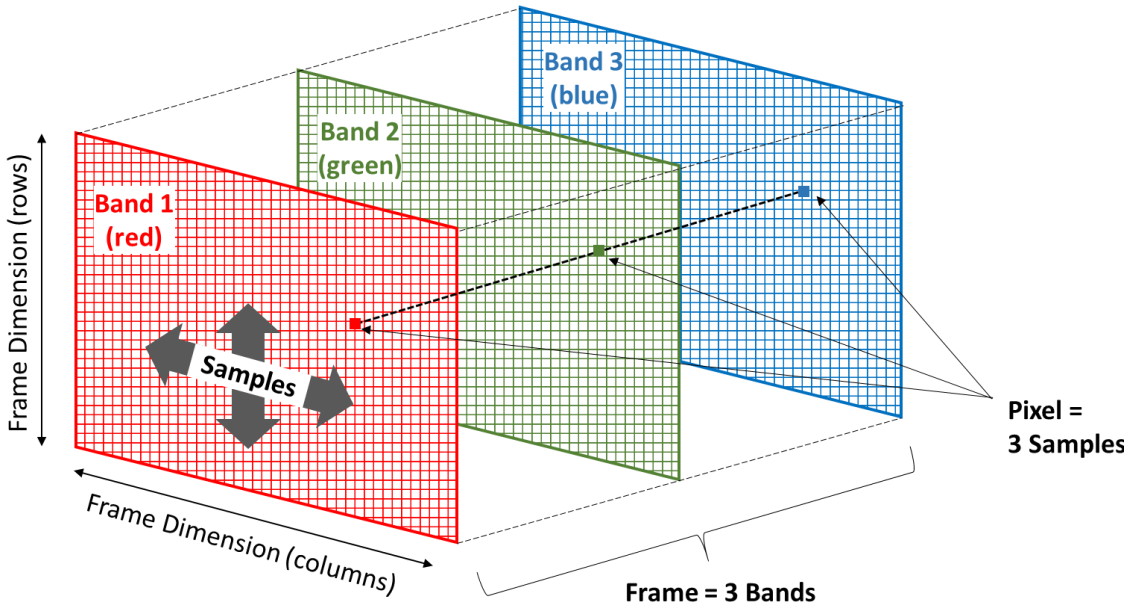


Figure 1-1: Samples, Pixels, Bands and Frame

The **Frame Dimension** is the height of the Frame measured in Pixels per column and the width measured in Pixels per row. Pixels within a Frame are bounded in their measurement over a period of time; that is, they have a **Start Time** and **End Time**. The Start Time and End Time may be based on Relative Time or Absolute Time. Although all Pixels within a Frame generally have the same Start Time and End Time, this is not always the case. A Frame is bounded by a **Frame Start Time** and **Frame End Time**, which accounts for the extremes of Start Time and End Time for the individual Pixels. Subtracting the Frame Start Time from the Frame End Time

results in the **Frame Period**. The inverse of the Frame Period (1/Frame Period) is the **Frame Rate**.

Sample: a numerical value that represents measured phenomena, such as light intensity along with its Start Time and End Time.

Band: collection of Samples where all measured phenomena are of the same type.

Pixel: A combination of a number of Samples collectively taken from a number of Bands.

Frame: A two-dimensional array of regularly spaced Pixels in the shape of a rectangle indexed by rows and columns along with a Start Time and an End Time of each Pixel.

Frame Dimension: The height and width of a Frame measured in Pixels per column and Pixels per row, respectively.

Frame Start Time: The minimum time value of all Pixel Start Times within a Frame.

Frame End Time: The maximum of all Pixel End Times within a Frame.

Frame Period: The time measured from the Frame Start Time to the Frame End Time.

Frame Rate: Inverse of Frame Period, measured in inverse seconds or Hertz (Hz).

A Frame is constructed by processing Source Data into a two-dimensional rectangular array of Pixels as illustrated in Figure 1-2.

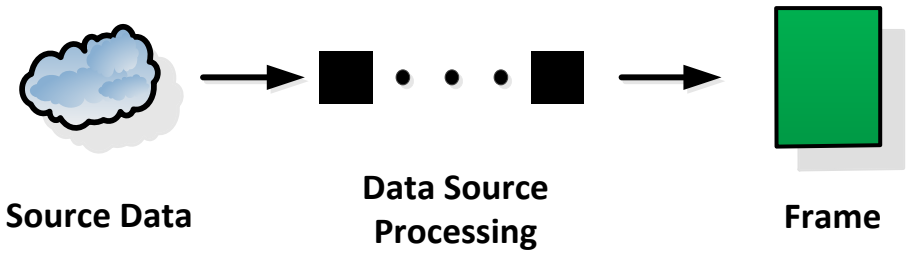


Figure 1-2: Generation of a Frame

There are two types of Source Data: Scene or Computer Generated. The space in the physical world imaged by a Sensor is called the **Scene**. **Scene Data** is data sensed by any device that detects **Scene Energy** from the Scene. Scene Energy includes Visible, Infrared, and Ultra-Violet light, plus RADAR and LIDAR returns, Acoustical or any type of data radiating from the Scene. Computer Generated Data is data that is not emanated from a Scene, but is rather manufactured to represent or simulate some type of scene or other information.

Scene: Space in the physical world that is sensed by a sensor and used to form an Image.

Scene Energy: Energy in any form that is radiated from the Scene.

Scene Data: A representation of the Scene Energy that is sensed and sampled by any device that detects energy from the Scene.

The Data Source Processing maps the Source Data to a two-dimensional rectangular output – the Frame. The Data Source Processing depends on the type of Source Data; it can be integral to the sensor, or exist as a separate system. Examples of Data Source Processing include: Visible Light

or Infrared (IR) cameras; the post processing of a 3D LIDAR cloud that supports a viewport into the 3D LIDAR scene; a simulation of flying over a city; simple text annotation. The Data Source Processing can provide contextual Metadata for the Frame and how it was formed.

The Data Source Processing may produce near-instantaneous Frames or Frames where the data is integrated over a period a time. Both types of Frames are bounded with a Frame Start Time and a Frame End Time (for the near-instantaneous case the Frame Start Time and Frame End Time are considered identical).

1.1.3 Image

Images are special cases of Frames. Frames represent content from either Scene or Computer-Generated Data; however, **Images** are Frames created only from Scene Data.

Image: A Frame with Pixels derived from Scene Data.

Image Dimension: The height of an Image measured in Pixels per column and the width of an Image measured in Pixels per row.

Image Start Time: The Start Time of an Image.

Image End Time: The End Time of an Image

Newscast graphics and computer animation are based on Frames, but because they are not produced from sensor data they are not Images. In contrast, the pictures from an air vehicle sensor, underwater sensor and sonar sensor are all Images, because they are formed from sensed data. Image is a subset of Frame (depicted as in Figure 1-3), therefore, Images retain all of the attributes of Frames (i.e. rectangular array structure and time information).

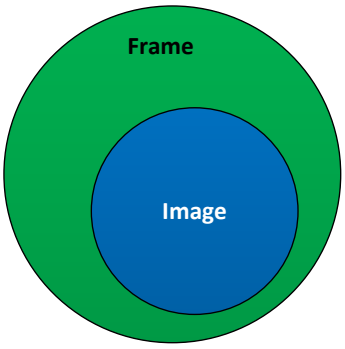


Figure 1-3: Image is a subset of Frame

1.1.4 Multiple Images

With two or more Images, relationships amongst Images can be formed both spatially and temporally. When two Images contain some portion of the same Scene, there is spatial overlap – these are called **Spatially Related Images**. Figure 1-4 illustrates spatial overlap, where the red square outlines similar content in each Image. Spatially Related Images do not necessarily need to occur within some given time period. For example, the two Images in Figure 1-4 may have been taken within milliseconds, minutes, or hours of one another. Spatially Related Images may be separated by a large difference in time, such as Images of a city taken years apart.

Figure 1-4: Example of Spatial Overlap

Images collected at some regular time interval, where the Images form a sequence, the Image Start Time for each is known, and each successive Image temporally follows the previous one, are called **Temporally Related Images**. There are no criteria that the content within Temporally Related Images be similar, only that they maintain some known time relationship.

Spatiotemporal data is information relating both space and time. For example, capturing a scene changing over time requires a sequence of Images to be captured at a periodic rate. While each Image portrays the spatial information of the scene, the sequence of these Images portrays the temporal or time-varying information. Images that are both spatially and temporally related are called **Spatio-Temporally Related Images**. These are the type of Images found in Motion Imagery.

Spatially Related Images: Images where recognizable content of a first Image is contained in a second Image.

Temporally Related Images: When the two Image Start Times of two Images are known relative to each other; the second Image is always temporally after the first.

Spatio-Temporal Related Images: When two Images are both Spatially Related Images and Temporally Related Images.

By collecting a series of Frames and/or Images, **Video** and **Motion Imagery** can be defined. The term "video" is not well defined in the literature. The word video is Latin for "I see." It has become synonymous with standards and technologies offered by the commercial broadcast industry. As such, the term serves a rather narrow segment of the application space served by Motion Imagery.

Video: An ordered series of Frames with each Frame assigned an increasing Presentation Time; where the Presentation Time is a Relative Time.

Presentation Time: A Relative Time associated with each Frame.

This definition of Video includes Presentation Time, which is an arbitrary relative timeline that is independent of a Frame's Start Time. For example, a Video of a glacier ice flow created by taking one picture per day has a Frame Start Time that is 24 hours apart from the next Frame; the Presentation Time, however, is set to play each Frame at a 1/30 second rate (i.e. Video at 30 frames per second).

Motion Imagery: A Video consisting of Spatio-Temporally Related Images where each Image in the Video is spatio-temporally related to the next Image.

Video is created from a sequence of Frames, whereas Motion Imagery is created from a sequence of Spatio-Temporal Related Images. Just as Image is a subset of Frame, Motion Imagery is a subset of Video. These relationships are depicted in Figure 1-5.

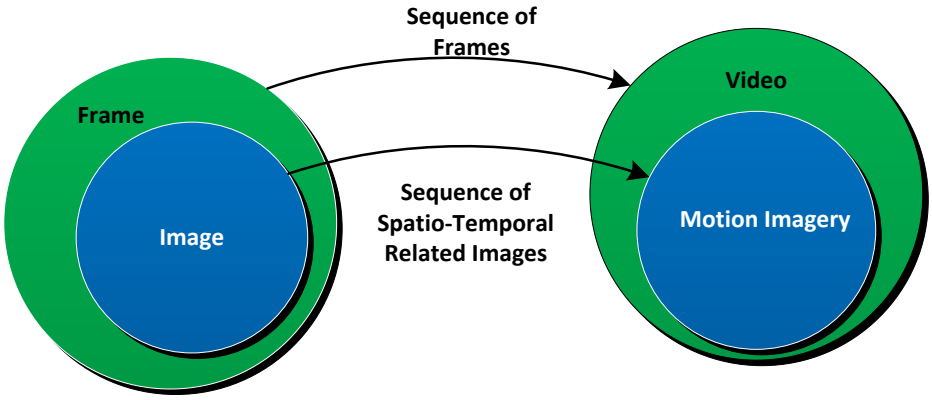


Figure 1-5: Relationships: Frame-to-Video and Image-to-Motion Imagery

1.1.5 Full Motion Video (FMV)

The term **Full Motion Video** (FMV) loosely characterizes Motion Imagery from Visible Light or Infrared sensors, playback rates typical of Video, and Frame Dimensions typical of those found in the commercial broadcast industry, defined by standards development organizations like SMPTE and ISO.

As with video, the term FMV characterizes a rather narrow subset of Motion Imagery. It is recommended the term FMV not be used, because of its ill-defined and limited applicability across the diverse application space served by Motion Imagery. Moreover, there is no clear definition for FMV available – it is sort of tribal knowledge and varies depending on who is asked. Historically, the term FMV was coined in the 90s by a vendor of video transponders to describe analog video that could be played back at its native frame rate showing all of the "motion" in the video.

The term FMV shall not be used in contractual language.

1.2 Metadata

Motion Imagery is the *visual* information that is exploited; however, in order to evaluate and understand the context of the Motion Imagery and its supporting system additional information called **Metadata** is needed. The types of Metadata include information about the sensor, the

platform, its position, the Image space, any transformations to the imagery, time, Image quality and archival information. Many MISB standards specifically address the definition of Metadata elements and the formatting of the Metadata associated with Motion Imagery. Chapter 7 discusses metadata in greater depth.

Chapter 2

Motion Imagery Functional Model

2.1 Introduction

A Motion Imagery Functional Model offers a common narrative across different audiences, such as Program Managers/Procurement officers, Technical Developers and End Users. The Functional Model describes the elements of systems that generate, manipulate and use Motion Imagery, and is based on the logical data flow from the Scene to the Analyst as shown in Figure 2-1. These elements include:

1) Scene - the data source for the Motion Imagery
2) Imager - a Sensor or Processor that converts Scene data into Images
3) Platform - static or movable system to which the Imager is attached
4) Control - a device that directs the Imager position, orientation or other attributes
5) Exploitation - the human/machine interaction with the Motion Imagery
6) Archive - stores Motion Imagery and additional exploitation data

In addition to these elements there are processing functions (denoted in the red block of Figure 2-1) used to format and manipulate the Motion Imagery; these are also included in the Functional Model.

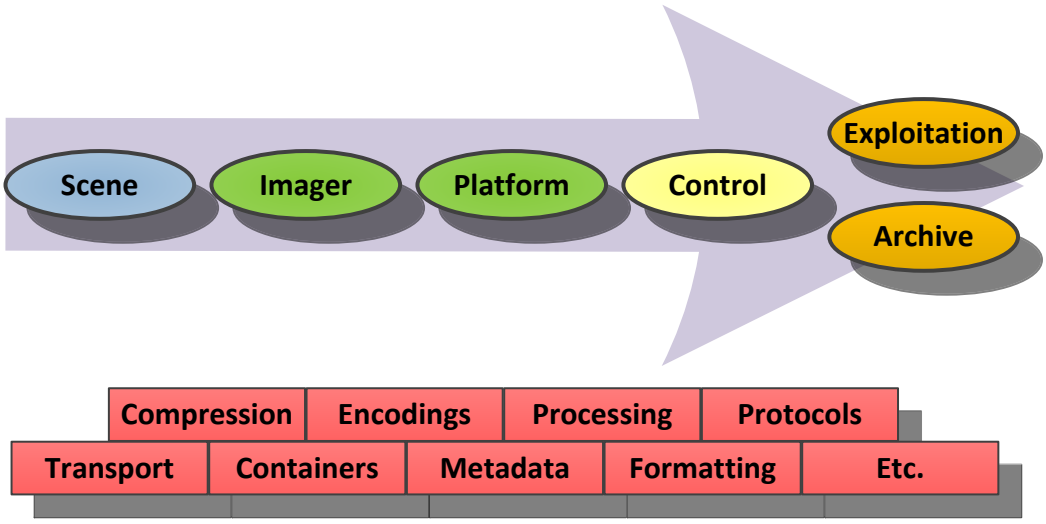


Figure 2-1: Elements of the Motion Imagery Functional Model

Using the Motion Imagery Functional Model, MISB Standards (ST) and Recommended Practices (RP) that address particular stages in the model are related. This facilitates ready association to those standards that are mandatory when specifying system requirements.

The Building Block Functions are core to MISB ST and RP documents; these may be cast in standalone documents, or as guidance provided within the MISP, where the function is defined generically and then referenced within MISB STs and RPs.

2.2 Scene

The Scene is what is being viewed by an Imager. The Scene propagates many different types of energy (Scene Energy) so different Imager types may be used to construct Images of a Scene. Each Scene may produce multiple types of Scene Data at the same time if multiple Imagers are used at the same time. Typical Imager types include:

- Electro-Optical - Emitted or reflected energy across the Ultra-Violet/Visible/Infrared portion of the electromagnetic spectrum (Ultraviolet, Visible, near IR, and IR).
    - Ultraviolet – Pictorial representation of Ultraviolet Energy.
    - Visible Light - Color or Monochrome
    - Infrared Light - Pictorial representation of thermal Infrared Energy.
    - Spectral Imagery - Image data captured in discrete frequency spectral bands across the electromagnetic spectrum.
        - MSI - Multispectral Imagery - 10's of individual spectral bands
        - HSI - Hyperspectral Imagery - 100's of individual spectral bands
- RADAR – Energy from the radio frequency portion of the electromagnetic spectrum transmitted towards the scene then reflected back and converted into an image representation.
- LIDAR - Laser pulses transmitted towards the scene then reflected back providing range information (i.e. point cloud) that is converted into an image.

2.3 Imager

The Imager converts the Scene Energy into an Image and, when possible, provides supporting information, such as the Imager characteristics or time about when the Samples or Pixels were created. Information that supports the Imager is called Metadata. The MISP specifies requirements on the format of imagery produced by an Imager, such as horizontal and vertical Sample/Pixel density, temporal rates, and Sample/Pixel bit depth. These requirements assure that common formats and structures are used, thereby facilitating interoperability.

Figure 2-2 illustrates the varieties of Scene Energy used to create Motion Imagery. Scene Energy is processed into Images along with associated Metadata. While the methods used to sense and measure the different types of Scene Energy are unique, a resulting Image, along with its Metadata is the common result.

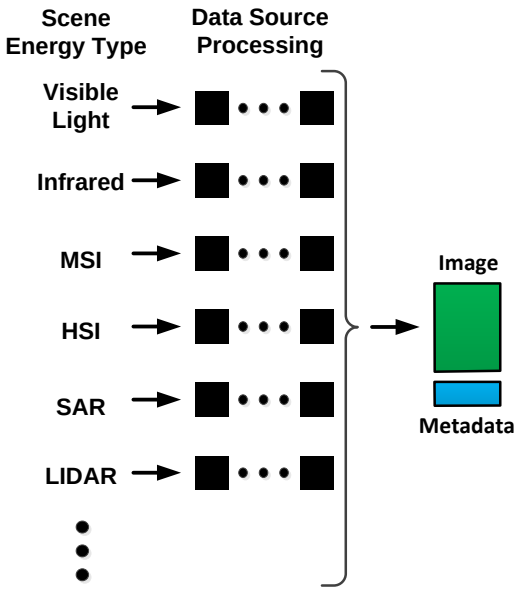


Figure 2-2: Motion Imagery from Varieties of Modalities

There are many types of Imagers depending on the type of Scene Energy and the phenomena being sensed. An important aspect of the Imager is the metadata that is gathered during the image formation process. The temporal information, orientation and position of the collected Image are needed for accurate geographical or relativistic positioning. When this information is not available during the image formation process then it is estimated in later stages of the functional model which may reduce accuracy.

Chapter 3 provides in-depth information about Imagers.

2.4 Platform

Any system to which the Imager is attached may be considered its platform. A platform may provide information regarding its environment, such as time, place, orientation, condition of the platform, etc. that may be quantified and provided in the form of Metadata along with Imager essence. The MISP provides numerous Metadata elements that serve specific purposes within its suite of Standards documents.

2.5 Control

Motion Imagery systems generally allow for control over the Imager, whether orienting its direction dynamically, or modifying its parameters, such as contrast, brightness, Image format, etc. The MISB does not issue guidance for control of a platform; it does, however, prescribe Metadata to indicate the state of control variables and actions and to enable Image transformations whether at the platform or in later phases of processing.

2.6 Exploitation

Exploitation of Motion Imagery may range from simple situational awareness – the when and where – to in-depth extraction of detected features, measurement, and coordination with other

intelligence data. Because the tools used in exploitation operate on the data structures of Motion Imagery, revisions to the MISP are backward compatible as much as possible, so all operational tools may continue to function as new capabilities are made available. While this is a goal, the advance and adoption of new technologies may impact compatibility in some cases.

2.7 Archive

Motion Imagery content is stored for later phases of exploitation, generating reports and historical reference for comparison. An important aspect of storage is file format. Choosing a standardized file format and a means to database/search the Motion Imagery is critical to reuse. The MISP provides guidance on several file containers and continues to evaluate new technologies that may offer greater value in the community.

2.8 Building Block Functions

A Building Block Function is itself a MISB standard that defines a reusable function that supports other higher-level MISB standards.

2.8.1 Compression

Motion Imagery typically is output by an Imager as a number of continuous sequential Images, where each Image contains a defined number of Samples/Pixels in the horizontal direction (columns) and a defined number of Samples/Pixels in the vertical direction (rows). The Images are spaced at a fixed time period.

Compression is an algorithmic sequence of operations designed to reduce the redundancy in a Motion Imagery sequence, so the data may be transported within a prescribed bandwidth transmission channel. The tradeoffs in compressing Motion Imagery are transmission data rate, Image quality and stream latency. These must be optimized on a per-application basis. The MISB governs the type of compression and provides guidelines for its proper use.

Audio is another “essence” type that may be provided by the platform. It also is typically compressed, and the MISB allows a choice among several industry compression standards.

2.8.2 Encodings

An encoding is the process of putting a sequence of characters (letters, numbers, and certain symbols) into a specialized format for efficient transmission or storage. Encodings such as KLV (Key Length Value) format are designed for low-overhead representations of Metadata. While many MISB Standards assume KLV encodings for Metadata, the MISB is investigating other encodings for use in web-enabled environments.

2.8.3 Protocols

Protocols provide the linkage for systems to communicate; they are key to interoperability. Protocols include the interface specifications for data transfer between functions along the Motion Imagery Functional Model. MISB chooses protocols specified by the commercial and international standards development organizations. When appropriate, these protocols are further profiled for specific use in this community, which aids interoperability and conformance.

2.8.4 Processing

Many points along the data flow within the Motion Imagery Functional Model are acted on for conversion, formatting and improvement to the signals passed. Examples include image transformations, data type conversion, and clipping of streams into smaller files. While the MISB does not dictate specific implementations of processing, the Standards are designed to provide flexibility and consistency across implementations.

Chapter 3

Imagers

There are two types of Imagers: Direct and Indirect. A Direct Imager transforms the raw Source Data information into an Image. Examples of direct Imagers include Visible Light cameras, Infrared Cameras, and Hyperspectral Sensors that gather data, perform some processing and generate the Image from the same perspective as the sensor. An Indirect Imager transforms the raw Source Data into an intermediate form, which is then processed into an Image via projective methods. A LIDAR sensor is an example of an Indirect Imager that produces Images by first building a point cloud; Images are then built from "flying" around the point cloud. Figure 3-1 illustrates the difference between the two types of Imagers.

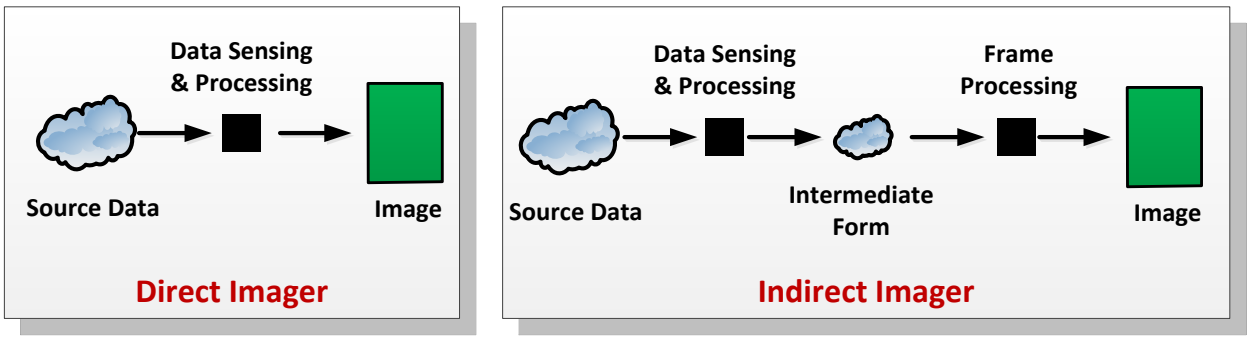


Figure 3-1: Types of Imagers

The remainder of this chapter focuses on Direct Imagers. All Direct Imagers convert Scene Energy to a series of Images. The process for producing an Image is unique for the given type of Scene Energy that is imaged; however, at a high level there are a common set of steps that define the Imager Processing Model.

Producing an Image from Scene Energy is a multi-step process with each step using the output of one or more previous steps spatially, temporally, or both. When performing precise exploitation of the Motion Imagery, it is important to understand what data manipulations have been performed on the original Scene Energy. The Image Processing Model shown in Figure 3-2 provides a consistent method for recording the process.

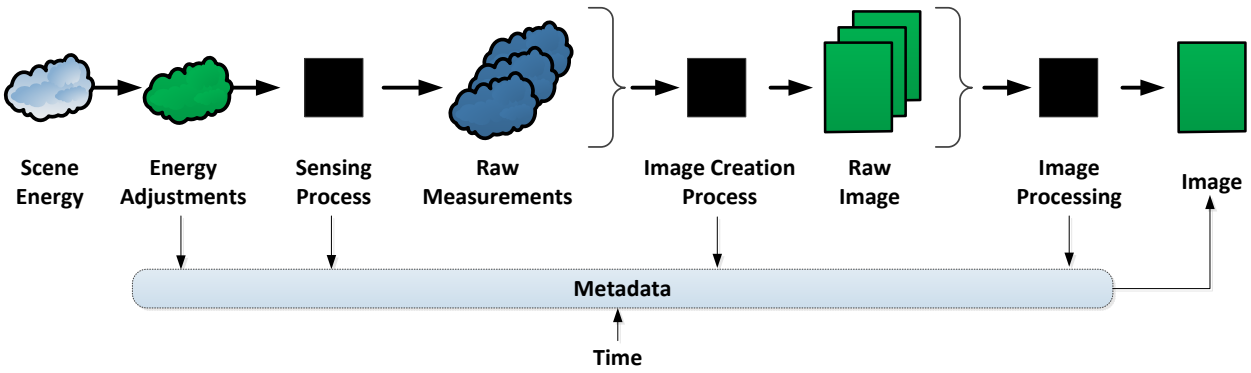


Figure 3-2: Imager Processing Model

The Image Processing Model defines a number of steps; however, depending on the Scene Energy Type and sensor type not all of the steps are needed (and can be skipped) to produce an Image. The only required step is the Sensing Process.

The Imager Processing Model shows the information flow from left to right:

- **Scene Energy**: the energy that emanates from a Scene (Section 3.1).
- **Energy Adjustments**: adjustments to the Scene Energy that occurs before the Sensing Process; for example, atmospheric distortions, optical filters, lens focusing, and distortions (Section 3.2).
- **Sensing Process**: measures the Scene Energy into a set of digital Raw Measurements (Section 3.3). Examples include: CCD camera, CMOS camera, Infrared camera and LIDAR receiver.
- **Raw Measurements**: a two-dimensional array of Samples (in any configuration, i.e. shape or spacing) that are measurements from the Scene. Each Sample has a numeric value, a location relative to the other Samples, a Start Time and an End Time (Section 3.4).
- **Image Creation Process**: maps the Raw Measurements to a set of regularly-spaced homogenous Samples in the shape of a rectangle. The mapping is dependent on the type of Scene Energy and Raw Measurements. An Image can be created from one or more sets of Raw Measurements either temporally or spatially (Section 3.5).
- **Raw Image**: same as definition of Image in Section 1.1.3.
- **Image Processing**: either augmentation or manipulation of one or more Images (spatially or temporally) for the purpose of formatting or enhancing an Image (Section 3.6).
- **Image**: as defined in Section 1.1.3.

3.1 Scene Energy

Scene Energy is any energy propagated from Scene to the Sensing Process. The energy comes from one or more of the following:

- Generated by the objects in the scene,

- Transmitted through objects in the scene,
- Reflections from objects in the scene from a natural light source,
- Reflections from objects in the scene from a controlled source

Scene Energy generated by objects or reflected from natural external sources is **Passive** energy. Scene Energy that is a reflection of a controlled source is **Active** energy. There are two forms of Scene Energy: mechanical and electromagnetic energy. Mechanical energy is energy that propagates through a physical medium such as water or air; examples of systems that measure this energy are sonar and acoustical sensors. Electromagnetic energy is energy that does not require a physical medium and can travel through empty space; examples of systems that measure this energy are visible light cameras, infrared cameras, Synthetic Aperture Radar (SAR) sensors and LIDAR sensors.

Mechanical and electromagnetic energy both propagate the energy using travelling waves. A travelling wave of energy from a scene is represented using Equation 1.

	$\chi(x,t) = A\,Sin(kx - \omega t + \phi)$	Equation 1

Where:

$\chi(x,t)$	Energy at a given distance (x) and time (t) from the scene,
A	Amplitude of the energy from the scene,
k	Wave number. The wave number is related to the energy's wave-length, λ by $k = 2\pi/\lambda$,
ω	Angular frequency. The angular frequency is related to the energy's frequency, v, by $\omega = 2\pi v$,
ϕ	Initial phase of the radiator energy.

Equation 1 is similar to Equation 7.5 from [2] but with the addition of the initial phase. This is added to support active energy Imagers, which can compute phase changes.

In most cases the Scene Energy measured for Motion Imagery is electromagnetic, which is discussed in Section 3.1.1. Mechanical energy systems, such as acoustical or sonar, are not documented in the MISP at this time.

3.1.1 Electromagnetic Scene Energy

Electromagnetic energy, or Electro-Magnetic Radiation (EMR), consists of photons travelling at the speed of light (c). Each photon carries an amount of energy (Q_e), that is related to the frequency of the radiation by $Q_e = hv$, where h is Planck's constant. Since EMR waves travel at the speed of light, the frequency (v) and wavelength (λ) are directly related by $c = v\lambda$. EMR waves have an orientation orthogonal to direction of propagation, called polarization, which can also be measured. Table 3-1 identifies the characteristics of EMR waves (from Equation 1) that can be measured by various sensing devices.

Table 3-1: Measureable EMR Properties

Name	Symbol	Description
Frequency	v	Number of oscillations per second. With EMR when the frequency is known the wavelength can be computed.
Wavelength	λ	Distance from the start to the end of one wave cycle. With EMR when the wavelength is known the frequency can be computed.
Photon Energy	Q_e	Energy which relates to the number of photons for a given frequency range.
Phase	ϕ	The angular difference in a wave's starting point from a given reference wave.
Polarization	θ	The orientation of the EMR wave orthogonal to the direction of propagation.

The range of EMR frequencies and wavelengths is described using the Electromagnetic Spectrum as illustrated in Figure 3-3.

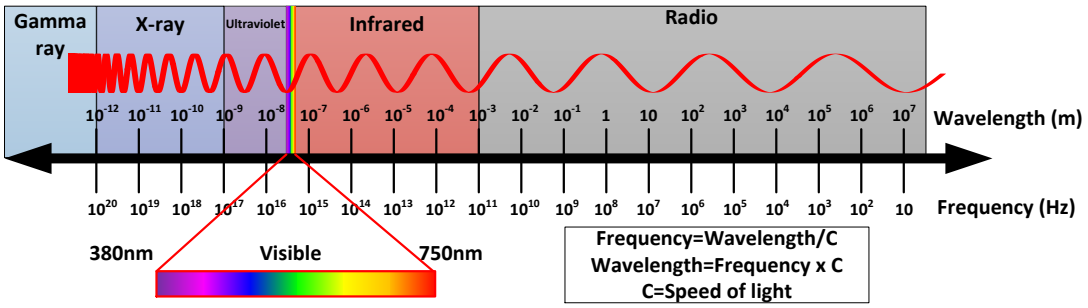


Figure 3-3: Electromagnetic Spectrum

Various ranges of wavelength with the electromagnet spectrum are grouped into Spectrum Bands. A Spectrum Band is loosely bound by a lower and upper wavelength and given a name; for example, the Visible Band range extends from 380 nanometers (violet) to 750 nanometers (red). Table 3-2 lists popular Spectral Bands along with their wavelength and frequency values. The Ranges for each band are not well defined and the ranges can overlap between bands

Table 3-2: Electromagnetic Bands. The exact wavelength/frequency ranges are notional.

Spectrum Band	Wavelength				Frequency			
	Start		End		Start		End	
Gamma Ray			10	pm			30	EHz
X-Ray	10	pm	10	nm	30	EHz	30	PHz
Ultraviolet	10	nm	380	nm	30	PHz	789	THz
Visible	380	nm	750	nm	789	THz	400	THz
Violet	380	nm	450	nm	789	THz	666	THz

Spectrum Band	Wavelength				Frequency			
	Start		End		Start		End	
Blue	450	nm	495	nm	666	THz	606	THz
Green	495	nm	570	nm	606	THz	526	THz
Yellow	570	nm	590	nm	526	THz	508	THz
Orange	590	nm	620	nm	508	THz	484	THz
Red	620	nm	750	nm	484	THz	400	THz
Infrared	750	nm	1	mm	400	THz	300	GHz
Near/Short Wave	750	nm	3	µm	300	GHz	100	THz
Mid-Wave	3	µm	8	µm	100	THz	37	THz
Long-Wave	8	µm	14	µm	37	THz	21	THz
Far-Infrared	14	µm	1	mm	21	THz	300	GHz
Radio	1	mm	100	Mm	300	GHz	3	Hz
Microwave	1	mm	1	m	300	GHz	300	MHz
mm	1	mm	7	mm	300	GHz	40	GHz
W	3	mm	4	mm	110	GHz	75	GHz
V	4	mm	7	mm	75	GHz	40	GHz
Ka	7	mm	12	mm	40	GHz	24	GHz
K	12	mm	17	mm	24	GHz	18	GHz
Ku	17	mm	25	mm	18	GHz	12	GHz
X	25	mm	37	mm	12	GHz	8	GHz
C	37	mm	75	mm	8	GHz	4	GHz
S	75	mm	150	mm	4	GHz	2	GHz
L	150	mm	300	mm	2	GHz	1	GHz
UHF	300	mm	999	mm	1	GHz	300	MHz
VHF	999	mm	10	m	300	MHz	30	MHz
HF	10	m	100	Mm	30	MHz	3	Hz

For a detailed overview of EMR see Chapter 7.1 of [2].

3.2 Energy Adjustments

Energy Adjustments are changes to the Scene Energy before it is measured. Energy Adjustments can affect any of the EMR properties listed in Table 3-1.

Waves travel through different mediums as they propagate from the Scene to the Imager. A Medium is a substance with a specific density. Energy propagates through different mediums such as water, air or glass at different rates. In classical EMR theory, electromagnetic waves can travel through mediums or through a vacuum, which is the standard reference medium for EMR.

There are two types of Energy Adjustments: Absorption and Transitional. Absorption adjustments occur when Scene Energy is consumed by the medium that the energy is

propagating through; for example, color filters absorb certain wavelengths of EMR. Transitional adjustments occur when Scene Energy in one medium (source) interacts with another medium (destination). As shown in Figure 3-4, three changes can occur to wave energy when interacting with the destination medium: Reflection, Refraction and Diffraction.

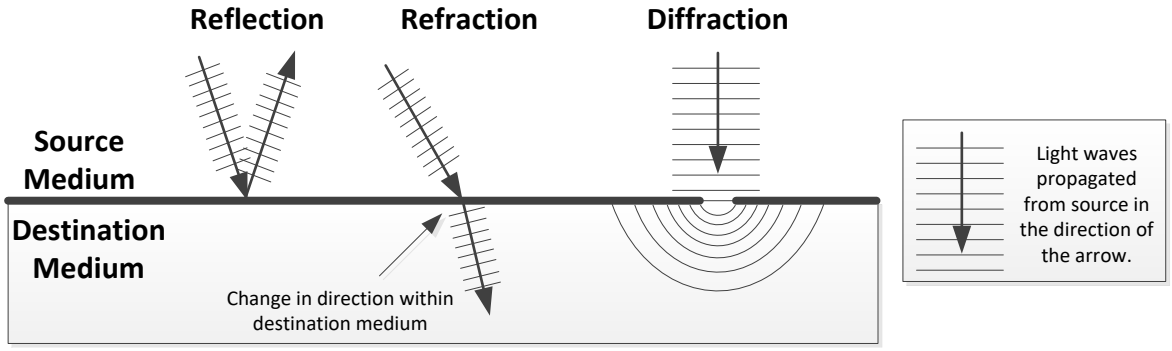


Figure 3-4: Illustration of Reflection, Refraction and Diffraction

Reflection occurs when a wave's propagation reverses direction away from the destination medium (i.e. bounces backward) and the wave's energy does not enter the destination medium. Refraction occurs when a wave's direction is altered as it enters the destination medium. Diffraction occurs when a wave "bends around" an object or interacts with a slit that is comparable in size to the wavelength. One or more of the three Energy Adjustments can occur simultaneously depending on the medium, the strength of the energy, and the wavelength of the energy.

In the MISP, two classes of Energy Adjustments are considered: Uncontrolled and Controlled. Uncontrolled energy adjustments are caused by environmental factors and are not directly measurable; however, some uncontrolled energy adjustments can be modelled, estimated and corrected in downstream processing. Controlled energy adjustments are deliberately imposed on the Scene Energy to enable or improve the energy measurements, for example a lens or a filter.

3.2.1 Uncontrolled Energy Adjustments

As Scene Energy travels to an Imager it passes through one or more mediums, all of which can distort the energy arriving at the Imager in unknown ways. Energy arriving at the Imager can be added to, reduced or changed in comparison to the original energy leaving the scene, as shown in Figure 3-5.

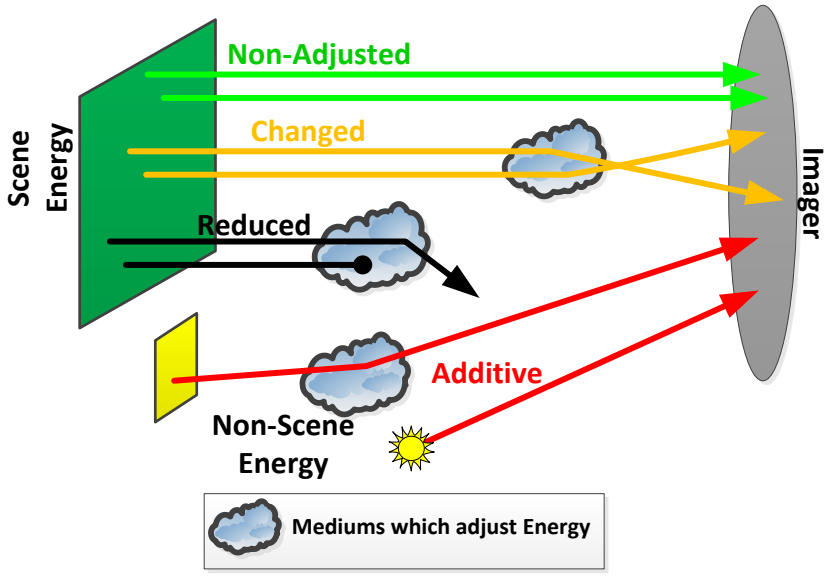


Figure 3-5: Uncontrolled Adjustments

Additive energy comes from any non-scene source such as refractions and reflections of energy not contributed by the scene. Additive energy is considered noise, which can be random, with a pattern or both. An example of added noise is the backscatter of light from particles in the air (e.g. fog). Additive energy is illustrated as red in Figure 3-5.

Energy is reduced when the scene energy is either directed away from the sensor (refracted or reflected), or the energy is absorbed by the medium it's passing through. For example, absorption of energy occurs in the atmosphere depending on the different wavelengths of light. Figure 3-6 shows the transmittance of energy through the atmosphere compared to the wavelength of the energy. Transmittance is the opposite of reduction, so the figure shows that wavelengths of 5.5 through 6.5 microns are completely absorbed or reduced by the atmosphere. Reduced energy is represented as black lines in Figure 3-5.

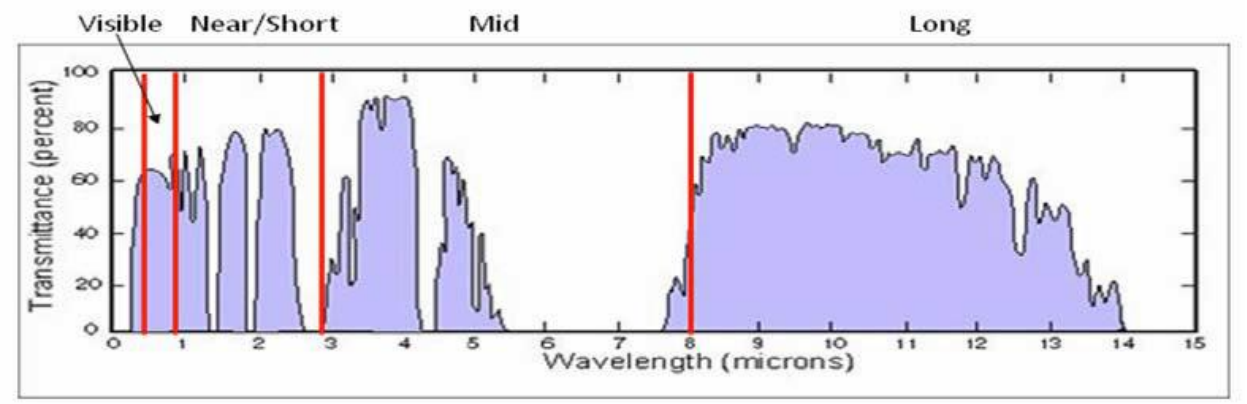


Figure 3-6: Transmittance of Energy through the Atmosphere

Changed energy results from refractions and diffraction of the Scene Energy. For example, energy passing through the atmosphere can be refracted and diffracted, resulting in what is called atmospheric distortions. Atmospheric distortions are caused by changes in the density of air,

which refracts the energy in various directions. Atmospheric distortions cause a set of parallel waves of energy from the scene to become divergent or convergent. Atmospheric distortions result in defocused (blurry) imagery. In Motion Imagery, atmospheric distortions will cause apparent movement or distortion in each successive image of a static scene. Changed energy is illustrated as orange lines in Figure 3-5.

Uncontrolled energy adjustments do not include distortions from Imager technology such as lenses, mirrors and glass; these are Controlled Energy Adjustments and are discussed in Section 3.2.2.

Corrections for uncontrolled energy can be made given sufficient information about the environment, and knowledge of the imaging system. Such information about the environment and imaging system can be included within the Motion Imagery metadata. Currently the MISB does not define this data.

3.2.2 Controlled Energy Adjustments

Controlled energy adjustments are deliberately-imposed energy adjustments on the Scene Energy to enable various types of energy measurements. Controlled energy adjustments include the application of reflection, refraction, diffraction and absorption of energy using devices such as mirrors, lenses, diffraction gratings and filters, respectively. Since the devices' behavior is known, metadata can be collected to assist in characterizing the collected data, and, correcting any adverse effects from such energy adjustments.

Typically, a multitude of controlled energy adjustments are combined either serially or in parallel, providing a wide range of functionality which includes focusing, zoom, color imagery, hyper-spectral imagery and others.

Chapter 4 of [2] provides a detailed description of Controlled Energy Adjustments (e.g. lenses, mirrors and filters) used by Imager systems.

3.3 Sensing Process

The Sensing Process converts Adjusted Energy into a set of digital Raw Measurements over a specified time period. In the MISB Image Processing Model, the Sensing Process requires a device that accepts Scene Energy and produces digital Raw Measurements of some aspect of the energy. These aspects include Intensity, Frequency, Phase and Propagation Time.

Intensity is the amount of Energy incident on a given area for a given time period. Intensity measurements range from counting a single photon (e.g. Geiger Mode LIDAR) to determining the total amount of energy captured over a detector's area during a defined period of time. Filters and prisms provide the means to measure the Intensity of different wavelengths at the same time, which can be further processed into color, multi- and hyperspectral imagery.

Energy Frequency, Phase and Propagation Time Raw Measurements measure energy information to/from the Scene; this is used by LIDAR and RADAR systems. Imagers that measure Frequency, Phase and Propagation Time are not discussed in the MISP at this time.

Imagers are constructed from one or more individual sensing elements that collect energy collaboratively. There are many configurations and types of sensing elements but they all measure energy, which can be indirectly equated to counting photons.

3.3.1 Single Element Detection

Sensing intensity is achieved by using electro-mechanical devices, called *detectors*, which collect energy (photons) over a period of time and report the results as a digital value called a Sample (see Section 1.1.2). As discussed in [3], detectors are divided into two classes: Thermal and Photon (or Quantum). Thermal detectors absorb EMR producing a change in the temperature of the detector relating to the intensity of the source EMR. At this time, Thermal detectors are not a subject within the MISP.

Quantum detectors interact with the incoming EMR resulting in changes to four possible electrical characteristics of the given detector material: EMR can be converted directly into an electrical charge, EMR can produce a photocurrent, EMR can cause a change in resistance (photoconductivity), or EMR can generate a voltage across a junction (photovoltaic).

Generally, all detectors include two primary stages: **Exposure** and **Readout**. **Exposure** accounts for the detector interacting with the EMR, where the device is either thermally or quantumly changed. *Exposure Duration* is the period of time spent in this stage. After the Exposure Duration, additional EMR is prevented from being included in the measurement until the start of the next Exposure Duration. The detector is "shuttered" during the non-exposure time. Historically, a shutter was a physical device that blocked light from exposing film. With digital sensors, a shutter can be either a physical device or an electronic means of preventing EMR from affecting a detector's measurement.

Exposure: When EMR is interacting with a detector for the purposes of measuring changes in the characteristics of the detector.

Exposure Start Time: The Start Time of Exposure for a detector.

Exposure End Time: The End Time of Exposure for a detector.

Exposure Duration: The time period when the detector is exposed to EMR measured as the difference between Exposure End Time and Exposure Start Time.

Readout is the process where the changes in the detector are converted into electrical signals and transferred out of the detector as a data value. The period of time for this process is called the *Readout Duration*.

Readout: When the changes of a detector (after Exposure) are converted to either analog or digital values.

Readout Start Time: The Start Time of Readout for a detector.

Readout End Time: The End Time of Readout for a detector.

Readout Duration: The time period when Readout is occurring measured as the difference between Readout End Time and Readout Start Time.

In addition to Exposure and Readout other processes for clearing charges, shutter drains and other detector "maintenance" are performed. Between the Exposure End Time and the Readout Start Time other operations may be performed that clear the detector and other electronic components. Figure 3-7 illustrates an example of timing for a single detector. First, there is a Clear operation, which is followed by an Exposure (with Exposure Start Time and Exposure End Time noted), then a Clear-Register operation, which is followed by a Readout operation (with

Readout Start Time and Readout End Time noted). These four steps complete one operating cycle of the detector. This cycle then repeats. The number of cycles, or Exposures, per second is called the Exposure Rate. Image frame rate is equal to the Exposure Rate when temporal processing is not performed in later steps (i.e. Image Creation and Image Processing) of the Imager Processing Model.

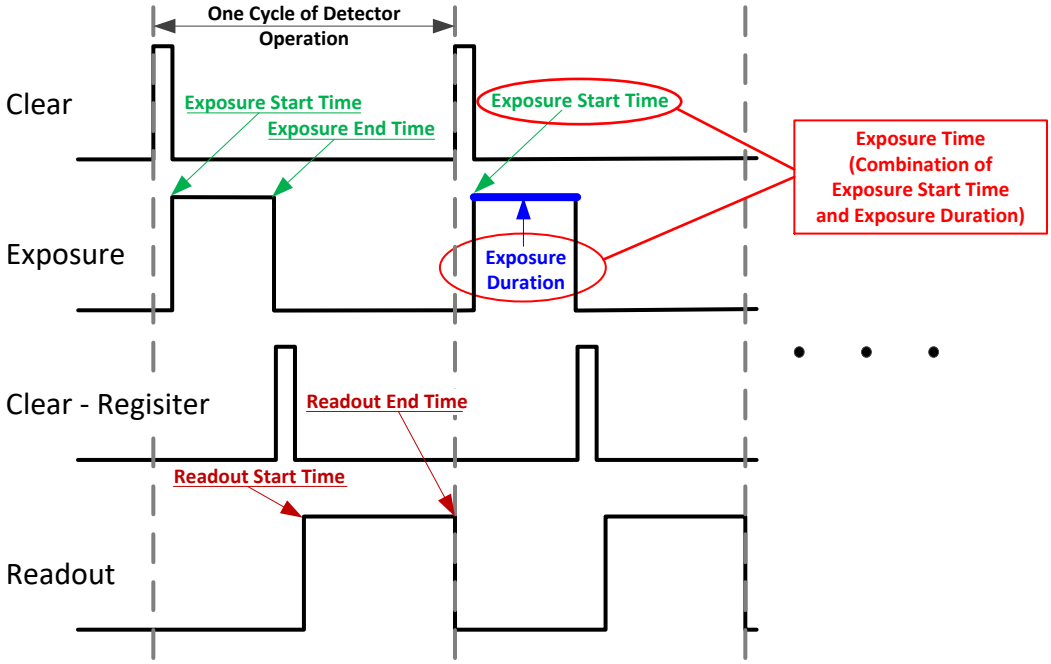


Figure 3-7: Illustration of Timing for a Single Detector.

Exposure Rate: The number of Exposures processed by a Detector in one second.

In this handbook, Exposure Time is comprised of two time measurements: Exposure Start Time and Exposure Duration, which is an offset to the Exposure Start Time. Equivalently the Exposure Time can be defined as the Exposure Start Time and Exposure End Time. By providing both the Exposure Start Time and the Exposure Duration (or Exposure End Time), any time within the Exposure Duration can be computed as a reference time for the image.

Exposure Time: The pairwise combination of the Exposure Start Time and the Exposure Duration.

Exposure Time (alternate): The pairwise combination of the Exposure Start Time and the Exposure End Time.

Detectors have a physical shape or surface where they gather EMR. The detector shape is typically rectangular or square, but other shapes, such as hexagons, are sometimes used for compact detector spacing (see Detector Groups in Section 3.3.2).

Along with a detector's measurements, related information – if determined and recorded – provides great value; these include: Exposure Time, the surface area of the detector, and the physical location of the detector. In the MISP and MISB-related standards, the Exposure Time is

specified by the Precision Time Stamp or the Nano Precision Time Stamp, and the physical location of the detector is included in the photogrammetric metadata.

3.3.1.1 Physical Description of Single Detector and Supporting Infrastructure

Single detectors are devices that collect photons over some surface area of the device. To support photon collection, the surface area may be partially covered with non-photon collecting supporting material; this is shown in Figure 3-8, where the photon sensitive area covers a portion of the total detector surface.

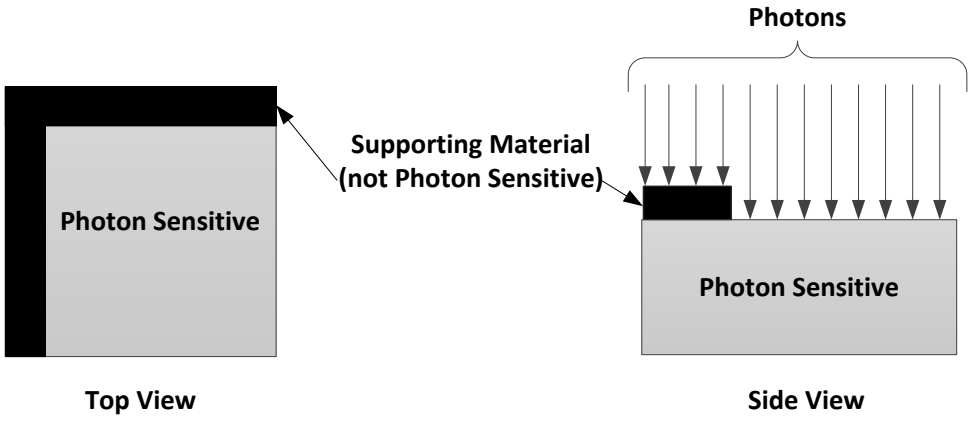


Figure 3-8: Illustration of a detector showing photon sensitive and non-sensitive areas.

A side view of the detector shows some incoming photons blocked by the devices supporting material. There are several methods for dealing with this issue. One method is to ignore the loss of photons and attempt to keep the ratio of photosensitive-to-support area as high as possible during design and manufacturing. Other methods include using lenses (see Section 3.2.2) or other devices to direct the light above the detector onto the photon-sensitive area, as shown in Figure 3-9. The lens focuses the light only onto the photon sensitive portion of the detector. An array of these lenses, used with a group of detectors, is called *lenslets* or *microlens* array.

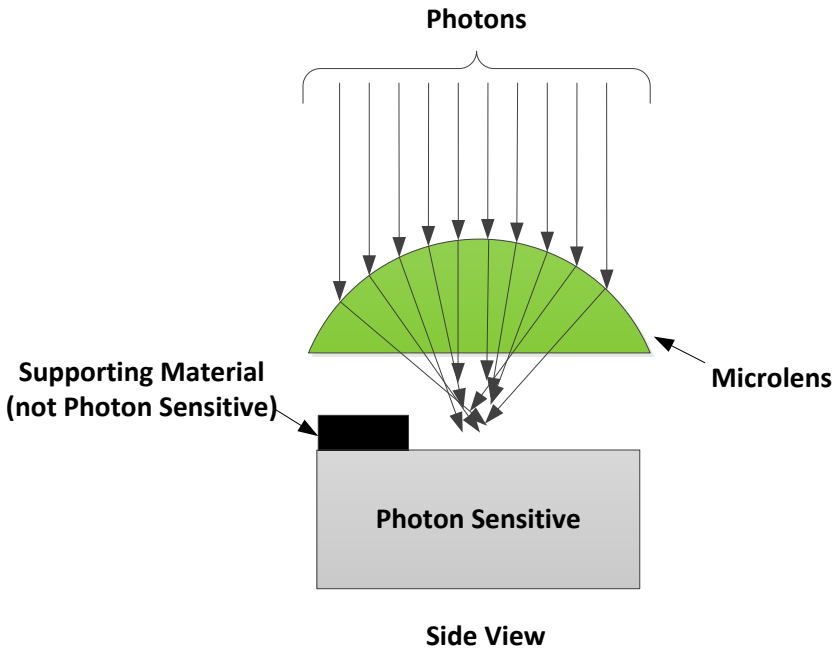


Figure 3-9: Illustration of a Detector with a lens to focus most of the incoming photons into the Photon Sensitive area.

Some detector systems include a filter intended to pass only photons meeting a specific criterion. For color Imagers, filters allow photons with certain wavelengths to pass through to the detector. For Polarimetric Imagers, filters allow only photons with certain wave orientations to pass through to the detector. When detectors are used in a Detector Group (Section 3.3.2), the filter types may not be the same for each detector. For example, two adjacent detectors may have two different color filters, such as a blue and green color filter. There are many different filter patterns used within Detector Groups (Section 3.3.2) based on manufacturer and purpose. Two examples of filter patterns are the Bayer pattern, used to detect red, green and blue EMR, and Polarimetric filters, used to detect different polarization orientations of EMR. Figure 3-10 illustrates a blue filter, which prevents all colors except blue from reaching the detector. The filter can be above or below the micro lens.

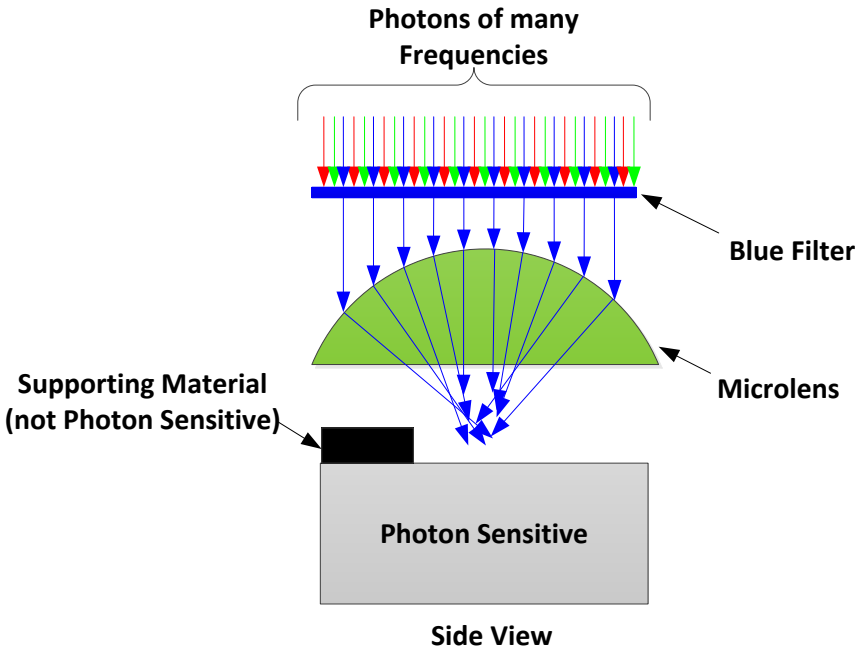


Figure 3-10: Illustration of blue filter over a single detector.

3.3.1.2 Noise

Several types of noise may affect the output of a detector, including shot noise, dark current noise and circuit noise. Shot noise is attributed to the statistical nature of gathering EMR within a detector. Dark current noise (aka thermal noise) is caused by temperature variations within the detector over time. Circuit noise comes from supporting electronics used to amplify and measure the photons within the detector. For more information on noise sources see Chapter 7 of [2].

3.3.2 Detector Groups

Detectors are configured and operated together in Detector Groups. A **Detector Group** combines multiple individual detectors spatially, temporally or both, producing a set of Raw Measurements (i.e. Samples) used to create an Image. A Detector Group can have any physical configuration or geometric shape. The most primitive configuration of a Detector Group is a single line (linear) of two or more detectors. A more common Detector Group configuration is a rectangular array of detectors, where the rows and columns of detectors may lie in a grid pattern, offset grid pattern or other patterns. Figure 3-11 illustrates linear, regular grid, offset grid and hexagonal detector patterns.

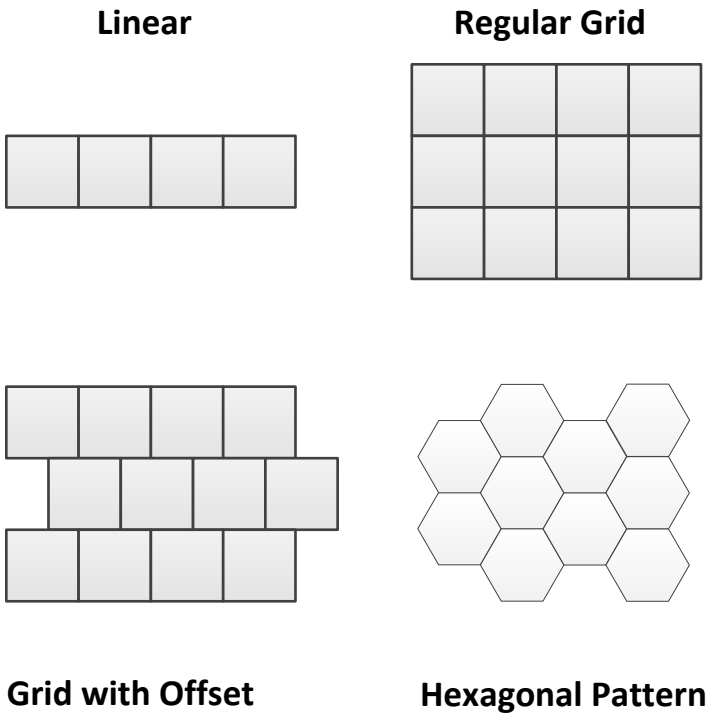


Figure 3-11: Detector Group Patterns

Both the physical layout of the Detector Group and the area of EMR being imaged by each detector are important metadata for photogrammetry. With most layout patterns, the center location of each detector can be computed if the pattern, spacing and detector sizes are known.

Detector Group: A collection of detectors operating together with a physical layout (Detector count and dimension, Detector pattern, and pitch (distance) between Detectors).

The Exposure and Readout of each detector (Section 3.3.1) in a Detector Group may occur at different times for a single set of raw measurements from the Detector Group. Examples include: (1) Exposure of all detectors simultaneously within the Detector Group (i.e. Global Shutter), or (2) Exposure of one line of a Detector Group at a time sequentially through the image (i.e. Rolling Shutter), or (3) random Exposure of each detector in the Detector Group. In all cases, the Exposure Time(s) are important metadata for Motion Imagery Sensors to report; this information is called the **Exposure Configuration**. An Exposure Configuration is a list of all Exposure Times for a single set of Samples collected from all detectors in a Detector Group. The Exposure Configuration may change for each set of Samples measured depending on adjustments needed for proper exposure (i.e. auto exposure adjustments).

Exposure Configuration: A list of all Exposure Times for a single set of raw measurements from a detector group.

Specifying the Exposure Configuration for every detector produces a large quantity of metadata. For some Exposure Configurations, the quantity of metadata can be reduced. Identifying the constant values and repetitions within an Exposure Configuration can help to minimize this metadata.

To organize the Exposure and Readout of a Detector Group, each Detector Group is divided into Regions, where each Region is composed of Sub-Regions. Figure 3-12 illustrates a Detector Group (the composite rectangle) with two Regions (A and B), where each Region consists of a number of Sub-Regions (Detectors common to a Sub-Region are shaded similarly).

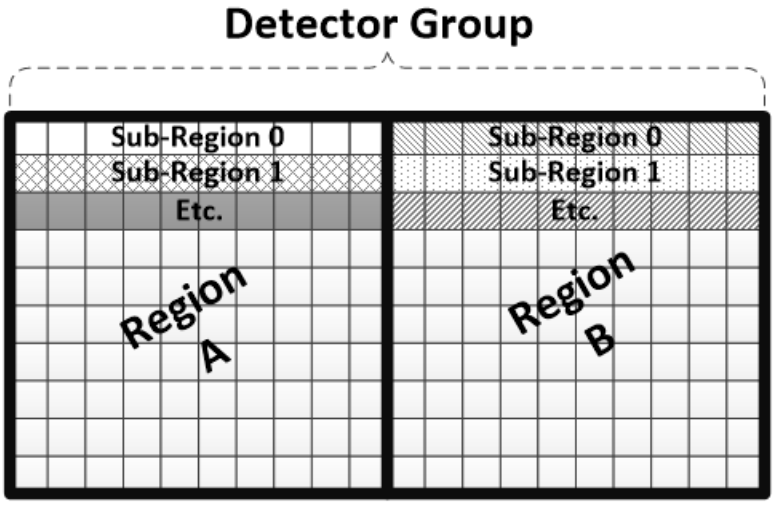


Figure 3-12: Illustration of Detector Group, Region and Sub-Region

3.3.2.1 Regions

A **Region** is created by dividing a Detector Group into rectangular Sub-Regions that share the same Exposure *orientation*. Orientation is the direction across the Region that two or more Sub-Regions are exposed.

Region: A collection of Sub-Regions with the same Exposure orientation.

Possible Region readout orientations are shown in Figure 3-13; these include Top-Down, Bottom-Up, Left-Right and Right-Left. In this illustration each Region consists of five Sub-Regions, where each row or column is a Sub-Region. In the Top-Down case, for example, each Sub-Region beginning with the top row then downward as numbered is exposed and read out sequentially.

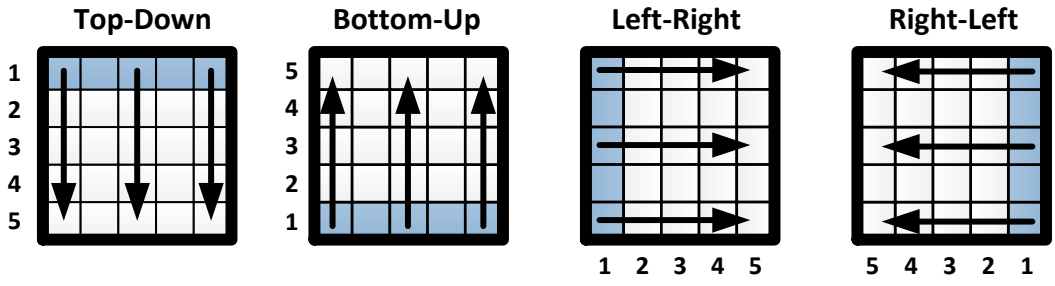


Figure 3-13: Region Readout Orientations across Sub-Regions

The different orientations of Figure 3-13 can be found within the same Detector Group. For example, Figure 3-14 (left figure) shows a Left-Right Region (Region 1) together with a Right-Left Region (Region 2) organized as an Outward-In configuration, and a Right-Left Region

(Region 1) together with a Left-Right Region (Region 2) in an Inward-Out configuration (right figure).

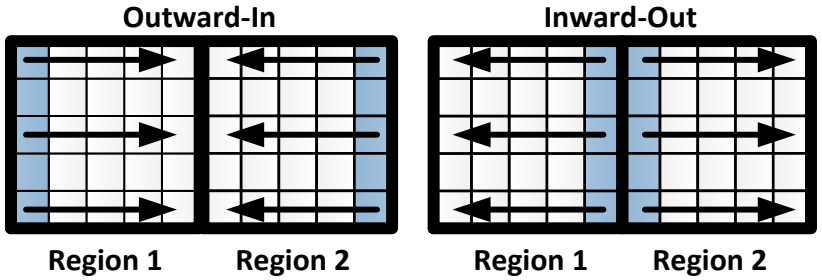


Figure 3-14: Different Region Orientations in the same Detector Group

A Sub-Region area is the spatial size – in rows and columns of detectors – that each Sub-Region occupies within a Region. In Figure 3-13 the area of the Sub-Region in the Top-Down example is a one by five (i.e. one row by five columns of detectors).

3.3.2.2 Sub-Regions

A **Sub-Region** is a line (vertical or horizontal) of detectors that operate with a similar Exposure Start Time and an identical Exposure Duration. The Exposure Start Time for all Detectors in a Sub-Region has a linear relationship with the Exposure Start Time of the first Detector.

Let $\tilde{s}_i$ denote the Exposure Start Time for the i[th] detector in a Sub-Region, where $\tilde{s}_0$ denotes the Exposure Start Time for the first detector. Also, let Δ indicate a linear offset from the Exposure Start Time of the first detector in the Sub-Region. Then, the Exposure Start Time for successive detectors in the Sub-Region is computed as $\tilde{s}_i = (i * \Delta) + \tilde{s}_0$. When all detectors in a Sub-Region have the same Exposure Start Time (i.e. the offset Δ=0) the Sub-Region is called *Homogeneous*.

Now, let δ_e denote the Exposure Duration, and E_i the Exposure End Time. If the Exposure Duration is constant for all detectors in a Sub-Region, the Exposure End Time for a given detector is $E_i = \tilde{s}_i + \delta_e$ (see Figure 3-15 for an example).

Sub-Region: A collection of detectors in which each detector has similar Exposure Start Time and the same Exposure Duration. The Exposure Start Time of a given detector in the Sub-Region can be computed as a linear offset from the first Detector of the Sub-Region.

3.3.2.3 Exposure Configuration Examples

The following two examples illustrate how Regions and Sub-Regions are used to define the timing of a Detector-Group.

3.3.2.3.1 Exposure Configuration Example 1

In Figure 3-15 a basic rolling shutter Exposure Configuration is shown. The Detector Group is represented as one Region with Sub-Region's 0 through N. Each Sub-Region contains M detectors within a row. The Exposure Start Time for the j[th] Sub-Region is denoted as S_j. Within each Sub-Region the Detector Start Time is the same; thus, the Sub-Regions are homogeneous (i.e. $\Delta = 0$) and $S_j = \tilde{s}_{j,0} = \tilde{s}_{j,1} = \tilde{s}_{j,2}$ ... etc., where $\tilde{s}_{j,i}$ is the i[th] detector in the j[th] Sub-Region.

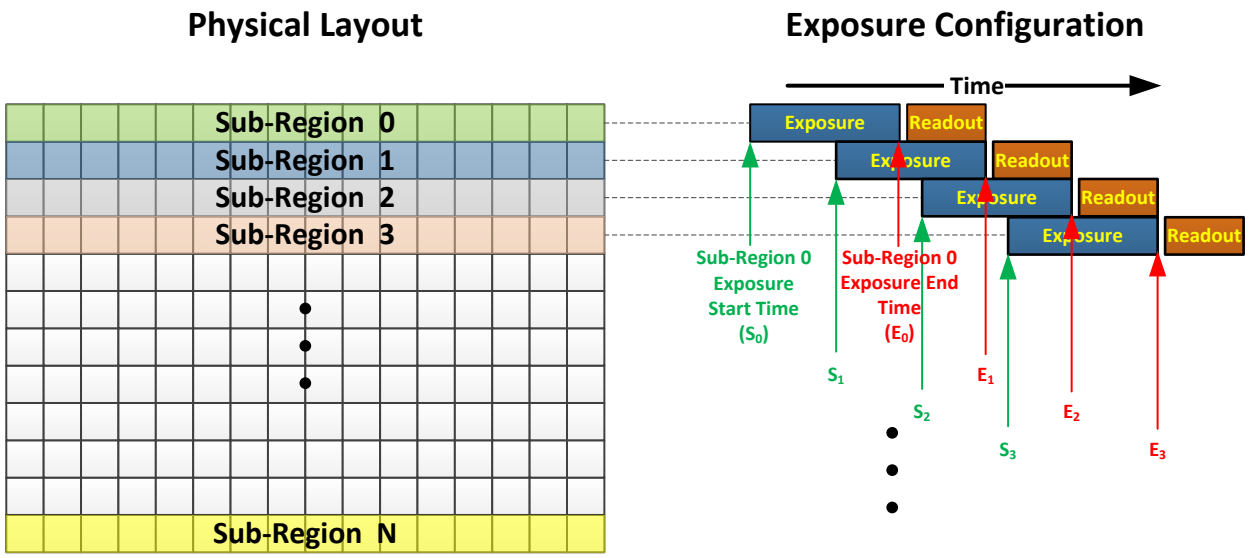


Figure 3-15: Illustration of a Detector Group with N+1 Sub-Region and a Rolling Shutter Exposure Configuration

In Figure 3-15 the Exposure and Readout process begins with the exposure of the detectors in Sub-Region 0 (Exposure Start Time = S_0) followed by the Readout of the Samples from Sub-Region 0. A short time after S_0, there is exposure of the detectors in Sub-Region 1 (Exposure Start Time = S_1) followed by its Readout, which begins after the Readout of Sub-Region 0 is completed. This pattern continues for the entire Detector Group. For simplicity, other operations for clearing the detector or registers are not shown. This process continues producing N+1 Raw Measurements (one for each row of detectors) to form a complete Image. The resulting Exposure Configuration for Figure 3-15 is listed in Table 3-3.

Table 3-3: Exposure Configuration for Figure 3-15

Sub-Region Number	Exposure Start Time	Exposure End Time	From		To	
			Row	**Column**	**Row**	**Column**
0	S_0	E_0	0	0	0	M
1	S_1	E_1	1	0	1	M
2	S_2	E_2	2	0	2	M
3	S_3	E_3	3	0	3	M
Etc.						
N	S_N	E_N	N	0	N	M

Using the information from Table 3-3, the Exposure Time (Exposure Start Time and Exposure Duration) for each detector can be determined. The total time to read out the Detector Group is then $T_{DG} = E_N - S_0$.

If the example is adjusted for non-homogeneous Sub-Regions, then an additional offset value Δ must be accounted for in each Sub-Region.

3.3.2.3.2 Exposure Configuration Example 2

Example 2 demonstrates a more complicated Exposure Configuration, where the Detector Group employs a layout of Regions with different orientations and Exposure's. The Regions are physically separate, but can perform readout operations at the same time as other Regions. Figure 3-16 illustrates a Detector Group with sixteen 3x4 detector Regions, each with the orientation shown by the arrows. Each Region is composed of 3x1 detector Sub-Regions (the first Sub-Region is shaded). Each arrow color represents a Region with the same timing. In this example Regions 1A, 1B, 1C and 1D are all exposed with the same timing. Each Sub-Region is homogeneous.

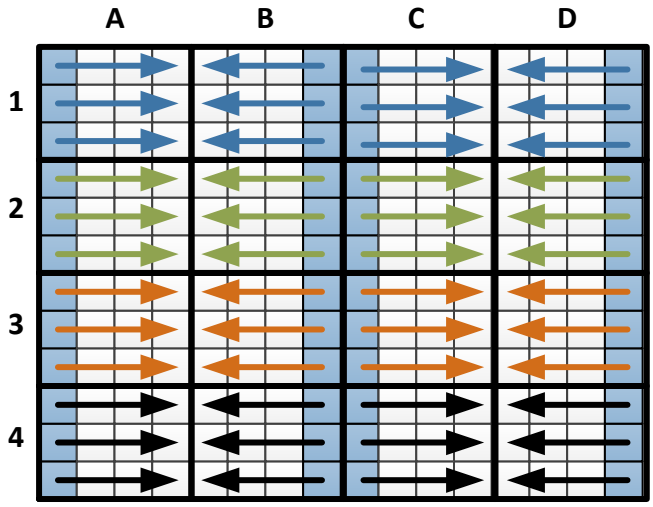


Figure 3-16: Illustration of Regions in a Detector Group used to define an Exposure Configuration

To determine the Exposure Configuration for a complete Detector Group, each Region needs to have its orientation, area and timing defined for every Sub-Region as shown in Table 3-4. The Exposure Start Time and Exposure End Time are denoted as $S_{R,j}$, $E_{R,j}$ where R is the Region identifier (i.e. 1A, 1B, …) and j is the j^{th} Sub-Region within Region R. In this example, all Sub-Regions that start with 1 operate at the same time in parallel, so their Exposure Start Time is based on S_{1A}, likewise for Regions that start with 2, etc.

Table 3-4: Exposure Pattern for Figure 3-16

Region	Sub-Region	Exposure		Orientation	From		To	
		Start Time	**End Time**		**Row**	**Column**	**Row**	**Column**
1A	0	$S_{1A,0}$	$E_{1A,0}$	→	0	0	2	0
1A	1	$S_{1A,1}$	$E_{1A,1}$	→	0	1	2	1
1A	2	$S_{1A,2}$	$E_{1A,2}$	→	0	2	2	2
1A	3	$S_{1A,3}$	$E_{1A,3}$	→	0	3	2	3
1B	0	$S_{1A,0}$	$E_{1A,0}$	←	0	7	2	7
1B	1	$S_{1A,1}$	$E_{1A,1}$	←	0	6	2	6
1B	2	$S_{1A,2}$	$E_{1A,2}$	←	0	5	2	5
1B	3	$S_{1A,3}$	$E_{1A,3}$	←	0	4	2	4
Etc. for Regions 1C & 1D								
2A	0	$S_{2A,0}$	$E_{2A,0}$	→	3	0	5	0
2A	1	$S_{2A,1}$	$E_{2A,1}$	→	3	1	5	1
2A	2	$S_{2A,2}$	$E_{2A,2}$	→	3	2	5	2
2A	3	$S_{2A,3}$	$E_{2A,3}$	→	3	3	5	3
2B	1	$S_{2A,0}$	$E_{2A,0}$	←	3	7	5	7
2B	1	$S_{2A,1}$	$E_{2A,1}$	←	3	6	5	6
2B	2	$S_{2A,2}$	$E_{2A,2}$	←	3	5	5	5
2B	3	$S_{2A,3}$	$E_{2A,3}$	←	3	4	5	4
Etc. for remaining Regions								

This table defines the timing and readout orientation of every Sub-Region. The total time to read out the Detector Group is then $T_{DG} = E_{4A,3} - S_{1A,0}$.

3.3.2.4 Exposure Patterns

As indicated earlier, the quantity of metadata to describe an Exposure Configuration can be large. An Exposure Pattern captures the constant values and repetitions within an Exposure Configuration to help minimize this metadata. A global Exposure Pattern includes both the *Inter*-Region patterns, which are the layout of Regions of the Detector Group, and the *Intra*-Region patterns, which are the timing within the Region (i.e. Sub-Region timing).

3.3.2.4.1 Inter-Region Exposure Patterns

To describe an Exposure Configuration for any Detector Group, an Exposure Pattern is needed for each Region. As discussed in Section 3.3.2.1, a Region is defined by its position, area and orientation. The position and area are specified using two opposite (diagonal) corners of the Region. Let the first corner -- the Region Starting Corner be denoted D_0, and the second corner -- the Region Ending Corner be denoted D_3. D_0 is the location of the first Detector of the first Sub-Region, while D_3 is the last Detector of the last Sub-Region in the Region. To establish the

orientation, the first Detector of the second Sub-Region is specified -- denoted D_2. As a convenience for the Intra-Region timing, a fourth Detector is specified that indicates the last Detector of the first Sub-Region -- denoted D_1.

Figure 3-17 illustrates the Exposure Pattern of the Detector Group from Figure 3-16. Each Region in this example is a single vertical line of three detectors.

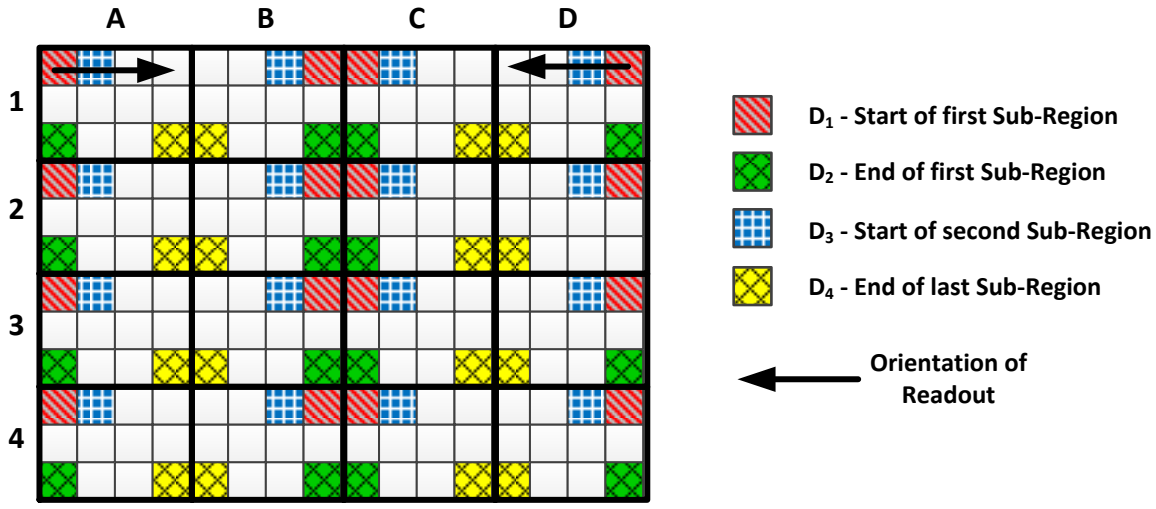


Figure 3-17: Example Exposure Pattern

As shown in Figure 3-17, once all of a Detector Group Region's patterns are defined, the start and end of each Region can be determined. To complete the Exposure Configuration, the timing information associated with each Sub-Region is defined as described in Section 3.3.2.4.2.

3.3.2.4.2 Intra-Region Exposure Pattern

Intra-Region Exposure Patterns are defined using the starting and ending timing of specific Detectors within Sub-Regions. The notation used to specify starting or ending time of a specific Detector is $S_{R,j,i}$ or $E_{R,j,i}$ respectively for the Rth Region, jth Sub-Region and ith Detector within the Sub-Region.

Within a single *homogeneous* Region, an Exposure Pattern is defined that describes all of the Sub-Region timing information using only three values: the first Sub-Region's Exposure Start Time and Exposure End Time ($S_{R,0,0}$, $E_{R,0,0}$) and the Exposure Start Time of the second Sub-Region ($S_{R,1,0}$). These values are used to compute two values, the Exposure Duration (λ) and Sub-Region Delay (γ), which are assumed constant for the entire Sub-Region. This assumption may reduce the accuracy of the timing, but the overall effect is minor. As defined in Section 3.3.1, the Exposure Duration is the difference between the Exposure End Time and Exposure Start Time; that is $\lambda = E_{R,0,0} - S_{R,0,0}$. The Sub-Region delay is the time between the Exposure Start Time of each Sub-Region, $\gamma = S_{R,1,0} - S_{R,0,0}$.

Within a Region R, knowing $S_{R,0,0}$, $E_{R,0,0}$, $S_{R,1,0}$ and the number of Sub-Regions, the complete Region's Exposure Configuration can be computed. Each $S_{R,j,i}$ can be computed using $S_{R,j,i} = S_{R,0,0} + j * \gamma$, and $E_{R,j,i}$ can be computed by adding the duration (λ) to $S_{R,j,i}$.

To illustrate the reduction of data possible using a Configuration Pattern for a Region's Exposure Configuration, Table 3-5 shows how each Exposure Start Time and Exposure End Time from Table 3-3 are computed. For these time values, Table 3-3 would produce 2*N values to report the Exposure Start Time and Exposure End Time for this example, while Table 3-5 only produces three values (i.e. $S_{R,0,0}$, $E_{R,0,0}$, $S_{R,1,0}$) to define all the times.

Table 3-5: Exposure Metadata for Figure 3-16 using Exposure Pattern ($S_{R,0,0}$, $E_{R,0,0}$, and $S_{R,1,0}$)

Sub-Region Number	Exposure Start Time	Exposure End Time
0	$S_{R,0,0}$	$E_{R,0,0} = S_{R,0,0} + \lambda$
1	$S_{R,1,i} = S_{R,0,0} + 1 * \gamma$	$E_{R,1,i} = S_{R,1,i} + \lambda$
2	$S_{R,2,i} = S_{R,0,0} + 2 * \gamma$	$E_{R,2,i} = S_{R,2,i} + \lambda$
3	$S_{R,3,i} = S_{R,0,0} + 3 * \gamma$	$E_{R,3,i} = S_{R,3,i} + \lambda$
Etc.		
N	$S_{R,N,i} = S_{R,0,0} + N * \gamma$	$E_{R,N,i} = S_{R,N,i} + \lambda$

When the Sub-Regions are not homogeneous an additional value is needed to determine the offset (Δ) between the Exposure Start Time for each detector within the Sub-Region; this value is assumed to be constant for all detectors within a Sub-Region. The offset can be computed by reporting any Exposure Start Time after the first Exposure Start Time within the first Sub-Region. The chosen value to use is the last Exposure Start Time of the first Sub-Region ($S_{R,1,M}$); with this value the offset is $\Delta = \frac{S_{R,0,M} - S_{R,0,0}}{M+1}$. With a homogeneous Sub-Region $S_{R,0,M} = S_{R,0,0}$ so $\Delta = 0$. For non-homogeneous Sub-Regions, the equations for the Exposure Start Time in Table 3-5 are updated to $S_{R,j,i} = S_{R,0,0} + j * \gamma + i * \Delta$.

3.3.2.4.3 Complete Exposure Pattern

Combining the information from both the Inter and Intra Exposure Patterns a Detector Group's Exposure Pattern can be identified. Table 3-6 lists the information needed to define the timing for each Region, R, in a Detector Group. The first column lists the value identifiers defined in Sections 3.3.2.4.1 and 3.3.2.4.2; the second column defines the values that are used; the third column is the count of values. The byte size is dependent on the method of transferring the values.

Table 3-6: Single Region Exposure Pattern

Value Identifiers	Values	Value Count
$D_{R,0}$, $S_{R,0,0}$, $E_{R,0,0}$	row, col, time offset, time offset	4
$D_{R,1}$, $S_{R,1,M}$	row, col, time offset	3
$D_{R,2}$, $S_{R,1,0}$	row, col, time offset	3
$D_{R,3}$	row, col	2
	Total	12

To reduce the number of values (and bandwidth) there are situations where some of the values can be eliminated:

1. When all Sub-Regions within the Region share the same Exposure Start Time, then ($D_{R,2}$, $S_{R,2}$) is not required. If this value is omitted then $S_{R,2}$ is equal to $S_{R,0}$.
2. When all Detectors within the same Sub-Region share the same Exposure Start Time, then ($D_{R,1}$, $S_{R,1}$) is not required. If this value is omitted then $S_{R,1}$ is equal to $S_{R,0}$.

3.3.2.5 Exposure Pattern Types

There are many different Exposure Patterns that can be defined with the techniques defined in Section 3.3.2.4. There are four types of Exposure Patterns: Global Shutter, Type 1 Rolling Shutter, Type 2 Rolling Shutter and Multi-Group. The first three are more common than the last.

Table 3-7 lists the Exposure Pattern Types and further details are provided in the following sections. In some cases, the rules from Section 3.3.2.4.3 are applied to minimize the timing data needed for describing the Exposure Pattern.

Table 3-7: Typical Detector Group Sensor Types

Type	Description (Section)	Value Identifiers	Value Count
Global Shutter	3.3.2.5.1	($D_{R,0}$, $S_{R,0,0}$, $E_{R,0,0}$) ($D_{R,3}$)	6
Type 1 Rolling Shutter	3.3.2.5.2	($D_{R,0}$, $S_{R,0,0}$, $E_{R,0,0}$) ($D_{R,2}$, $S_{R,1,0}$) ($D_{R,3}$)	9
Type 2 Rolling Shutter	3.3.2.5.3	($D_{R,0}$, $S_{R,0,0}$, $E_{R,0,0}$) ($D_{R,1}$, $S_{R,0,M}$) ($D_{R,2}$, $S_{R,1,0}$) ($D_{R,3}$)	12
Multi-Region	3.3.2.5.4	see below	see below

3.3.2.5.1 Global Shutter

With a Global Shutter sensor, all Detectors within the Detector Group are exposed simultaneously, i.e. all Detectors in the Detector Group use the same Exposure Time. The only information needed to describe a Global Shutter sensor is the Exposure Start Time and Exposure End Time for one Detector, which in turn describes the timing for the whole group. With Global Shutter sensors the complete image is "frozen" in time (provided the exposure time is short enough such that there is no change in the scene during the exposure time). The advantage of a Global Shutter sensor is superior motion capture capability.

3.3.2.5.2 Type 1 Rolling Shutter

A Type 1 Rolling Shutter sensor has one Region for the whole Detector Group, and all detectors within each Sub-Region share the same Exposure Start Time. In a Type 1 Rolling Shutter sensor, each Sub-Region of the image is measured separately, thus introducing image effects. These image effects are discussed in Section 3.7.2.

3.3.2.5.3 Type 2 Rolling Shutter

A Type 2 Rolling Shutter sensor has one Region for the Detector Group, but each detector in within a Sub-Region does not share the same Exposure Start Time. In a Type 2 Rolling Shutter sensor, each row (or column) of the image is measured separately, thus introducing image effects. These image effects are discussed in Section 3.7.2.

3.3.2.5.4 Multi-Region Shutter

A Multi-Region Shutter sensor can expose different areas of the sensor at different times and in different orientations. Multi-Region shutter sensors will have image effects which are similar to Rolling Shutter effects but may be more chaotic in nature.

3.3.2.5.4.1Interlaced

An Interlaced Detector Group is a Multi-Region Shutter system that has N/2 Regions with each Region containing only two Sub-Regions (or rows); where N is equal to the number of rows in the Detector Group. Each Region shares the same timing. The first Sub-Region is exposed, and then the second Sub-Region is exposed a short time later. This is equivalent to the odd rows of the Detector Group being exposed simultaneously and the even rows are exposed after the odd rows have been exposed. This disjoint capture in time of image data introduces image effects. These image effects are discussed in Section 3.7.3.

3.3.2.6 Detector Group Readout and Stabilization

Some Detector systems can perform stabilization within the sensing unit by moving the area sampled to compensate for small movements of the Detector Group. The area of the Detector Group sampled is called a Detector Window. Figure 3-18 illustrates the effect of the stabilization, and the Samples measured from the Detector Group.

In Figure 3-18 illustration (1), a set of Samples is read from the Detector Group in the red area defined from Detector (2, 2) to (9, 12). In the subsequent illustration, the Sensor, and therefore Detector Group, moves (because of some disturbance), so the stabilization algorithm compensates to maintain the image on the same area of the Scene; this causes the Detector Window to shift. In illustration (2), a set of Samples is read from the Detector Group in the Detector Window defined from Detector (1, 4) to (8, 14). In the last illustration (3), another movement causes a shift, and a set of Samples is read from the Detector Group in the Detector Window defined from Detector (3, 1) to (10, 11). In these three illustrations, the Detector Window does not change size; only the location within the Detector Group changes. However, it is possible for a Detector Window to change size if necessary.

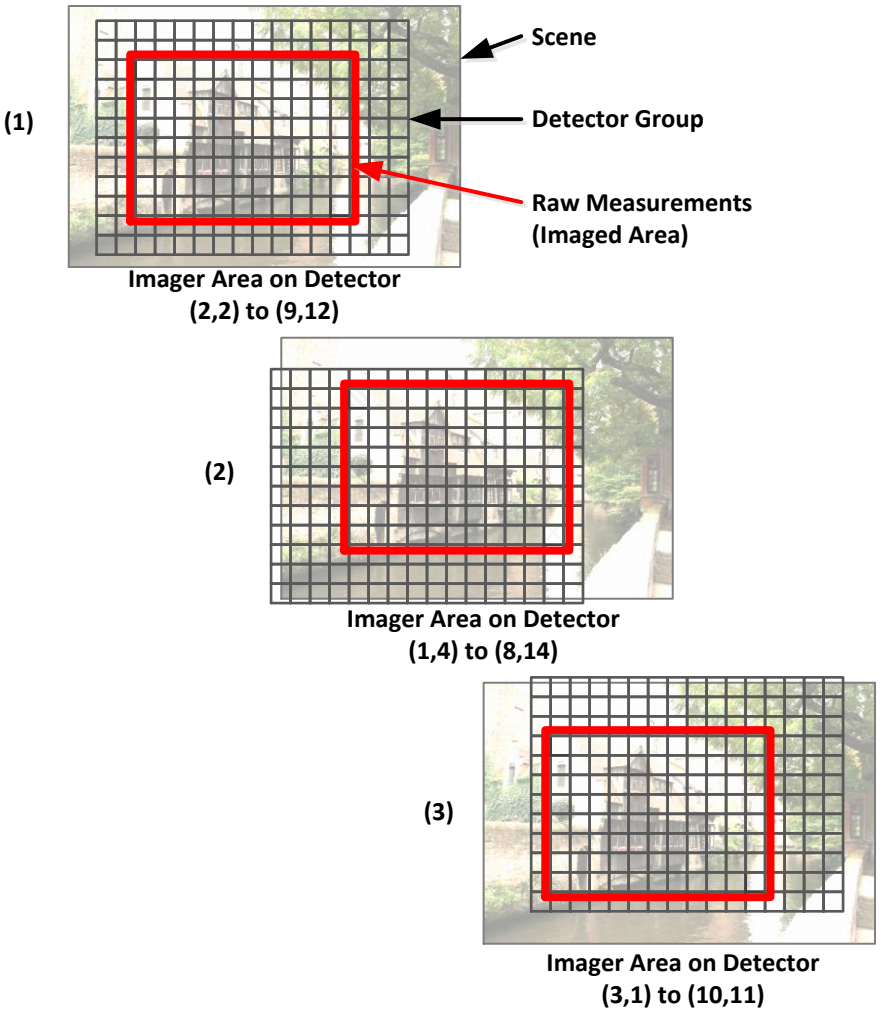


Figure 3-18: Stabilization of Raw Measurements and Detector Group movement.
(Bruges Belgium 2014 Lat: N51 12'18" Lon: E3 13'48")

3.3.2.6.1 Detector Window

An image sampled from a Detector Group can be geometrically smaller than the Detector Group for various reasons, such as dimensional requirements and the stabilization process. Although Detector Groups can be constructed to any density, standards specify dimensional requirements for practically reasons. For example, a Detector Group might contain 2048x2048 Detectors, but the application may require a 1920×1080 image; in this case, a match of Detectors to Image – or window of Detectors – is needed. Window information thus helps to define the two bounding corner Detectors used in a windowed process.

Additionally, the Detector Group and the Image will have different coordinate systems and orientation. The sensor can have any orientation relative to the scene, for example the light from the scene may be projected onto the Detector Group upside down or as a mirror image. Creating the Image from the Detector Group so that it is right side up requires the Window corners to have any orientation.

The necessary window information consists of two Detector corner coordinates C_1 and C_2, with each coordinate specifying a row and column from the Detector Group as listed in Table 3-8 and

illustrated in Figure 3-19. The corners can define any area within the Detector Group and can have any orientation. The resulting Image is mapped from the Detector window with C_1 being the upper right corner of the image and C_2 being the lower left corner of the image.

Table 3-8: Window Corner Values

Corner	Values	Description
C_1	row, column	First corner of the Detector Window. This corner becomes the upper left corner of the image.
C_2	row, column	Second corner of the Detector window, which is diagonally opposite from the first. This corner becomes the lower right corner of the image.

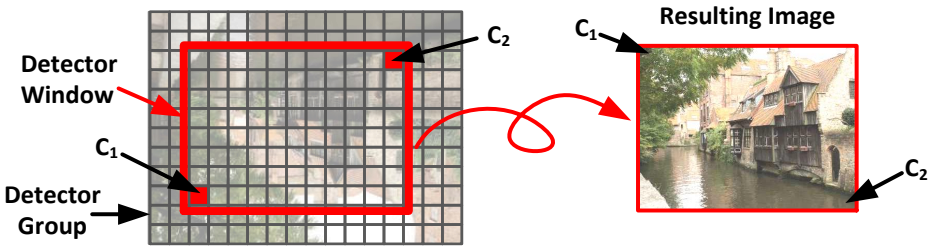


Figure 3-19: A Detector Window within a Detector Group (defined by C1 and C_2) and the resulting Image which is a rotation and flip (mirror image) of the projected light onto the Detector Group. ***(Bruges Belgium 2014 Lat: N51 12'18" Lon: E3 13'48")***

3.3.3 Sensor Configurations

There are a wide variety of sensors, lenses, prisms and filter combinations, which enable other types of phenomenology to be sensed; examples include multi-spectral, hyper-spectral imagery and plenoptic systems. With multi-spectral and hyper-spectral imagery prisms separate different bands of light, which are then individually detected at the same time. Plenoptic systems can use lenselet to send light to groups of individual detectors at the same time. The resulting imagery is lower in resolution, but can be post processed to enable post capture focusing on objects at different locations in the scene.

3.3.4 Other Sensing Topics

There are many methods used to improve the resulting detected image, including auto-exposure and binning. Auto-exposure is a process built into the sensor to adjust the exposure time based on the amount of energy being detected. Auto exposure changes the detectors timing without external controls to ensure that an unsaturated image is detected. Binning is a process where multiple detectors values are combined during readout to reduce noise and improve the signal-to-noise ratio.

3.4 Raw Measurements

The result of the Sensing Process is one or more Raw Measurement datasets. Each Raw Measurement dataset is the detector data recorded for a single exposure time for some Detector Subgroup. The Raw Measurements datasets contain a measured value and location for each

detector in the subgroup. The Detector Subgroup's shape may not be a regular gridded shape, so the Raw Measurement rows and columns may not represent adjacent data (e.g. hexagonal Detector Group). Alternatively, depending on how the Detector Subgroup is shaped, the location may be represented by the data position in an array.

3.5 Image Creation Process

The Image Creation Process converts Raw Measurements into a set of regularly-spaced homogeneous samples in the shape of a rectangle. During this processing detector values may be averaged or removed either spatially, temporally or both. Data from different exposure times may be adjacent to each other in the final result. The end result of this process is an Image as defined in Section 1.1.3.

For color imagery, a common step during the Image Creation Processing is computing the color bands from the detector data's Bayer or color pattern. Other processing that can occur during the image creation process includes algorithms for digital zoom and stabilization.

3.6 Image Processing

The final step before an Image leaves a camera is Image Processing. Image Processing is performed in-part or on the whole Image; for example, color smoothing/correction or averaging multiple temporal images into a single image to match a desired temporal rate. The result of the Image Processing step is an Image as defined in Section 1.1.3.

3.7 Image

The output of the Image Processing Model is a series of Images. These Images may show undesirable effects depending on the detector settings and the intended application, such as Shutter to Scene, Rolling Shutter and Interlaced effects.

3.7.1 Shutters

When designing Motion Imagery Systems, the expected action within the Scene and the frame rate of the Imager needs to be considered to prevent *temporal aliasing*. Temporal aliasing occurs when the frame rate is slower than the motion in a scene; for example, the movement of a propeller may look like it is rotating slowly while in fact it is rotating very fast. This optical illusion is called the wagon-wheel effect and may cause users to incorrectly interpret what is happening within the Scene.

3.7.2 Rolling Shutter

In a rolling shutter Imager, as defined in Section 3.3.2.5.2 and shown in Figure 3-15, the rows of detectors are not exposed at the same time. For example, in a "top down" rolling shutter Imager the first row is exposed, then a short time later the second row, then the third, etc. until the whole Detector Group's lines are exposed and readout. The Exposure Times for all of the detectors in one row are the same, and the time difference between rows is constant; however, the Exposure Duration for each row can change resulting in partial exposure effects (see Section 3.7.2.2).

The advantage of the rolling shutter is the Imager can continue to gather energy during the acquisition process, thus increasing sensitivity. They are also low cost as compared to other types of sensors. The disadvantages of rolling shutter include distortion of fast-moving objects (see Section 3.7.2.1) and partial exposure effects. Most rolling shutter technology is found in the consumer market i.e. cell phones. Since the exposure process moves *through* the Image over some length of time, users should keep in mind the following issues when working with rolling shutter sensors:

3.7.2.1 Motion Blur and Wobble

Motion blur may occur depending on the speed of motion within the scene or motion of the whole scene. The motion blur can severely distort the imagery to a point where objects no longer appear natural and uniform. For example, the blades of a fan may take on irregular shape (see Figure 3-20). When the Imager is subject to vibration, the Image may appear to wobble unnaturally, sometimes called the "jello-effect".

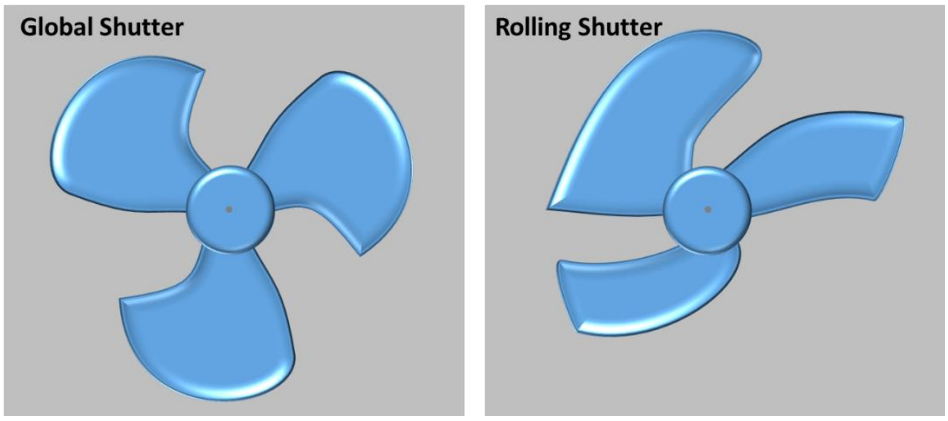


Figure 3-20: Example Motion Effects: Global vs. Rolling Shutter

When panning a rolling shutter Imager, objects in the scene may appear to lean away from the direction of motion as illustrated in Figure 3-21.

Figure 3-21: Illustration (simulated) of a rolling shutter image as the Imager pans quickly across the scene

3.7.2.2 Strobe Light and Partial Exposure

A rolling shutter Imager is not well-suited for capturing short-pulse light sources, such as strobe a light or flash. Unless the light source remains on for the duration of exposure, there is no guarantee of adequately capturing the light source. This will result in an Image with varying levels of illumination across the scene. As this effect manifests differently in successive Images of a Motion Imagery sequence, the imagery may appear to "breathe" with some content possibly washed out completely.

3.7.3 Interlaced

In an interlaced scan Imager, the Image is "imaged" in two passes: the odd-numbered rows of the Image are captured during the first pass, and then the even-numbered rows in the next pass. Thus, two complete passes (or scans) are required to capture a complete Image. One main drawback of interlaced scanning is that Images tend to flicker, and motion – especially vertical motion – appears jerky. Another drawback is Image detail, such as object edges can be "torn" demonstrating a stair-step "jagged" effect along an object edge, as shown in Figure 3-22. As the motion increases stair-stepping can become quite pronounced greatly distorting Image features essential to exploitation tasks. This distortion is further compounded when compressing an Image. Because the stair-stepping artifact introduces higher frequency detail, coding efficiency is reduced as the coder attempts to spend its allocated bits representing these artifacts. With higher compression ratios, these artifacts are even further degraded.

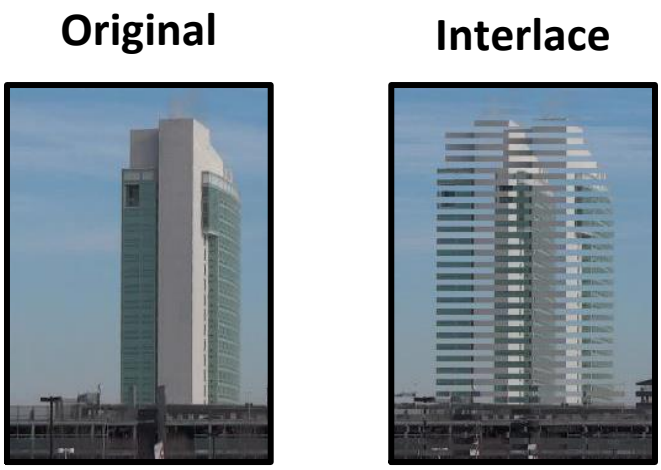


Figure 3-22: Illustration (simulated) of an interlaced image as the Imager pans quickly across the scene

Interlaced-scan is an older technology developed to deliver analog television within limited bandwidth criteria. Because of its time in the marketplace, it is inexpensive, which makes it attractive. However, it is a poor choice for surveillance applications, where poor edge definition and compression greatly reduces motion fidelity.

Chapter 4

Image Color Model

Color images are generally represented using three Bands comprised of a number of Samples per Band interpreted as coordinates in some color space. A color space is a mathematical representation of a set of colors. Several popular color spaces include RGB (Red-Green-Blue), Y'U'V' and Y'C'$_b$C'$_r$ (where Y' represents the Luma, (brightness information), and U'V' or C'$_b$C'$_r$ (the Chroma or color difference information). The "prime" notation indicates that the values represent the gamma-corrected versions of the corresponding signals. Gamma-correction describes the total of all transfer function manipulations, such as corrections for any nonlinearities in the capture process (see SMPTE EG 28 [4] for definitions).

Color spaces, such as Y'U'V' and Y'C'$_b$C'$_r$, which are derived as linear combinations of the R'G'B' values, are efficient representations to express color (e.g. the color difference signals require less bandwidth than Luma or the primary R'G'B' signals). As such, the color difference signals can be represented using fewer samples. Nomenclatures of 4:4:4, 4:2:2 and 4:2:0 denote spatial sampling of the color Bands. The graphic in Figure 4-1 helps to explain color sampling for these common encodings.

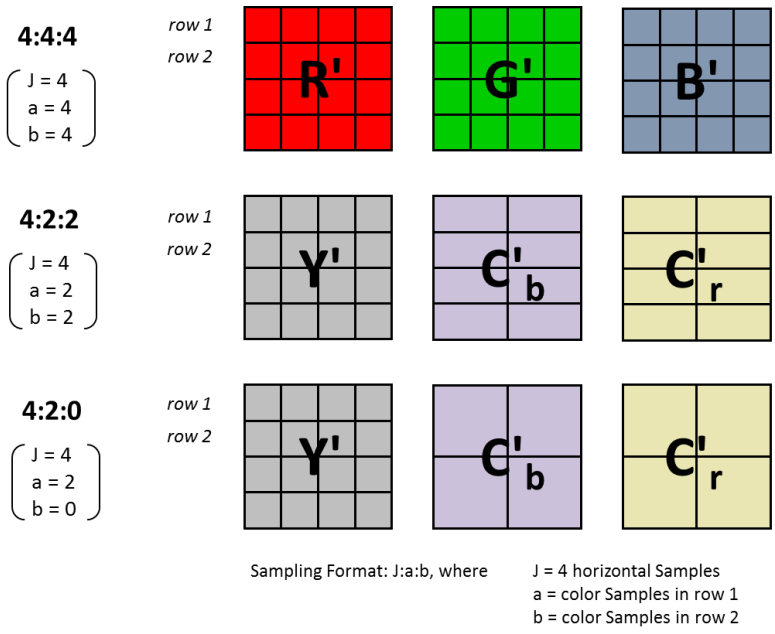


Figure 4-1: Examples of Formats with Chroma Subsampling

At the top of the figure, a set of 4x4 Sample arrays represent three color Bands, one each for Red, Green and Blue. Likewise, the middle and bottom show three Sample arrays that represent the color: one Band of Luma (Y') and two Bands of Chroma (C'$_b$C'$_r$). A sampling ratio with

notation J:a:b is used: "J" is the dimension of the array horizontally, in this case J = 4; "a" indicates the number of Samples in row 1, and "b" the number of Samples in row 2.

For example, in 4:4:4 (top of Figure 4-1), each of the three Bands have the same spatial sampling; that is, each Band (R'G'B' in the example) has a Sample that represents primary color information in each Pixel location. A 4:4:4 model contains the maximum number of Samples, which is 48 Samples ($16Y'+ 16C'_b + 16C'_r$).

In 4:2:2 (middle of Figure 4-1), every two Samples in row 1 share a Chroma Sample (a=2); likewise, every two Samples row 2 share a Chroma Sample (b=2). For 4:2:2 when forming a Pixel, a single Chroma Sample is reused by two Pixels (the Pixel's row-wise neighbor); this reduces the number of Samples by one-third to 32 Samples ($16Y'+ 8C'_b + 8C'_r$).

In 4:2:0 (bottom of Figure 4-1), every two Samples in row 1 share a Chroma Sample (a=2); row 2 shares its Chroma Sample with the top row. For 4:2:0 when forming a Pixel, a single Chroma Sample is reused by four Pixels (the Pixel's row-wise and column-wise neighbors); this reduces the number of Samples by one-half to 24 Samples ($16Y'+ 4C'_b + 4C'_r$).

Often a Pixel, such as "24-bit color Pixel" or "16-bit color Pixel", describes a 3-band set of Sample values. Determining the Pixel value for three Bands where each has the same spatial sampling is straightforward, i.e. Pixel Value Range = 3B, where B = bits per Sample for one Band. In the case of color sampling, an *equivalent* Pixel Value Range can be computed in reference to the Pixel arrangement shown in Figure 4-1. Note that in the color sampling for 4:2:2 and 4:2:0 the Chroma Bands have fewer Samples than the Luma Band.

In Table 4-1, the Pixel Value Range in bits per Pixel for the three-color samplings are listed for several Sample Value Ranges of 8, 10 and 12 bits per Sample. The Pixel Value Range is based on the number of possibly unique Samples within the Sample array. For instance, 4:4:4 has equal Sample spacing in each band, so there is one Sample in each band, i.e. full sample density. In 4:2:2 for every one Sample in Band 1 there are 0.5 Samples in Band 2 and 0.5 Samples in Band 3. Likewise, in 4:2:0 for every one Sample in Band 1 there are 0.25 Samples in Band 2 and 0.25 Samples in Band 3. Together the Samples across Bands represent one Pixel. The Pixel Value Range is then computed by multiplying the Average Number of Samples per Pixel by the Sample Value Range.

Table 4-1: Pixel Value Range for Various Color Sampling Formats

Color Sampling Format	3-Band color (Average Samples/Band)			Average Samples/ Pixel	Sample Value Range (bits/Sample)		
	Band 1	Band 2	Band 3		8	10	12
					Pixel Value Range (bits/Pixel)		
4:4:4	1	1	1	3	24	30	36
4:2:2	1	0.5	0.5	2	16	20	24
4:2:0	1	0.25	0.25	1.5	12	15	18

Chapter 5

Dissemination

Motion Imagery Data is often produced some distance away from where it is controlled and/or exploited. The action of transmitting Motion Imagery Data from a source (i.e. Imager, Platform or Control Station) to one or more users is called Dissemination. Transmitting Motion Imagery Data can affect end users in two ways: Quality and Latency. Motion Imagery quality is impacted by the compression applied to the Motion Imagery and data losses during transmission. Similarly, Metadata can also be impacted by data losses.

Latency is a measure of amount of the time it takes to move data from one point to another in a Motion Imagery System. Latency is impacted by the compression of the Motion Imagery and the transmission path taken. Total Latency is the elapsed time from an occurrence in the Scene to when that occurrence is viewed in the Motion Imagery at its destination. When Total Latency is significant, a platform pilot may not be able to accurately control the Imager(s), and an end user may not be able to coordinate with other users or Intel sources in real time. Therefore, minimizing Total Latency is an overarching design goal, especially for systems used for real time applications. There is always a balance between Quality and Latency as both are difficult to optimize at one time.

While the subject of transmission can be extensive, in this section common methods endorsed by the MISP for Dissemination for Motion Imagery Data are discussed.

5.1 Background

Although the MISP does not levy requirements on the transmission of Motion Imagery Data, the MISP does levy requirements on the Quality of Motion Imagery, which can be greatly impacted by the transmission; understanding some basic methods for transmission is beneficial. The health of a delivery path from Imager through Exploitation depends on many factors, and begins with the method of transmission.

5.1.1 Transmission Methods

There are three transmission methods typically used in MISP applications: Wireless, Wired and Mixed Use.

5.1.1.1 Wireless

Wireless transmission generally assumes a radio link, such as from an airborne platform to a ground station. Although wireless technologies are designed to support varied applications and have different performance criteria, they are susceptible to interference from other communications signals. Interference introduces distortion into the transmitted signal, which can cause data errors. Because errors in wireless transmission are anticipated, methods to detect and repair errors often are provided; for example, Forward Error Correction is one popular method

used in a digital link. Such processes add additional overhead to the data transmitted, and they are limited to correcting certain types of errors.

5.1.1.2 Wired

Wired transmission can be divided into circuit-switched and packet-switched technologies. In circuit-switching a dedicated channel is established for the duration of the transmission; for example, a Serial Digital Interface (SDI) connection between a Sensor and an Encoder. Packet-switching, on the other hand, divides messages into packets and sends each packet individually with an accompanying destination address. Internet Protocol (IP) is based on packet-switching.

5.1.1.3 Mixed Use

In a network infrastructure, a mix of wireless and wired transmission methods is usually present. For example, Motion Imager Data from an airborne platform might be transmitted wirelessly to a satellite, relayed from the satellite to a ground receiver, and then transmitted over a wired IP network. Each method of transmission has its own susceptibility to errors that must be understood by developers when implementing a Motion Imagery System, and by users who receive and use the data.

5.1.1.4 Bandwidth

Wired transmission, in general, offers greater bandwidth capacity than wireless; this has important implications in the dissemination of Motion Imagery Data. Because of the large data characteristics of Motion Imagery, compression is needed when delivering Motion Imagery over the more bandwidth-constrained wireless link. Compression and subsequent encoding increases the complexity of the data, which makes it susceptible to errors introduced in transmission.

5.1.2 Internet Protocols

Internet Protocols represent a family of protocols used in an Internet packet-switching network to transmit data from one system to another. Table 5-1 provides information about the Internet Protocol family.

Table 5-1: Internet Protocols

Protocol Name	Description
Internet Protocol (IP)	The principle communications protocol for relaying packets across networks. IP data packets (datagrams) are sent from a transmitting to a receiving system using switches and routers. IP is a low-level protocol that does not guarantee delivery, or when data arrives it will be correct (i.e. it could be corrupted).
User Data Protocol (UDP/IP)	UDP [5] uses a simple transport layer protocol based on IP. It does not guarantee data delivery or that data packets arrive in order. UDP specifies a network Port that enables multiple data sources from one system to be transmitted to multiple receiving systems. Data sent from one system to multiple systems is called multicasting. UDP provides one of the fastest methods of transmitting data to a receiver, which makes it suitable for time-sensitive applications (low latency). UDP multicasting is used in delivering Motion Imagery Data to multiple systems at once, which reduces overall network bandwidth.

Transmission Control Protocol (TCP/IP)	TCP [6] is a transport layer protocol that provides reliable guaranteed delivery of data. However, TCP does not guarantee time-sensitive delivery of data, but finds use in the transfer of non-time-sensitive data, such as Motion Imagery Data files.

When UDP/IP is used, there are several types of packet errors that can occur as shown in Table 5-2. These errors can affect any protocol or data that uses UDP/IP (i.e. RTP).

Table 5-2: UDP Error Types

Error Type	Description
Packet Loss	Packets can be lost in a number of ways, such as network routers/switches being overwhelmed, or network devices physically disconnected. When routers/switches are overwhelmed they will discard packets, which are then forever lost to all downstream devices. Other causes of packet loss include poor wiring and faulty equipment; these can cause intermittent packet loss and be hard to detect.
Packet Corrupted	Packets can be corrupted during the transmission from one device to another. Corruption can be caused by faulty equipment, poor wiring or from interference. Interference is primarily an issue with wireless technologies, although crosstalk in wired technologies can also be problematic. When routers/switches receive a packet and UDP error checking determines that the packet is corrupted, the packet is dropped and lost to a receiving system (see **Packet Loss**). If a corrupted packet passes its UDP error check, the corrupted packet could go undetected unless further error detection methods are used.
Packet Out of Order	Networks usually contain more than one router/switch, and typically there is more than one path for transmitting an IP packet from a source to a destination. Packets that take different paths may arrive at a destination out of the order they were transmitted. This condition is not detectable by UDP error checks, so other means for detecting and possibly reordering the packets need to come from additional information supplied within the transmitted data.

5.1.2.1 MPEG-2 TS Packets

The MPEG-2 Transport Stream (MPEG-2 TS [7]) is a widely used Container for disseminating Motion Imagery Data. For example, Motion Imagery Data transmitted from an airborne platform, as well as to points along a network that supports Exploitation, is typically in a MPEG-2 TS Container. Developed originally for wireless transmission of television signals, MPEG-2 TS is organized as successive 188-byte data packets with each packet including a 4-bit continuity count. This count can be used to detect whether a packet is either missing or received out of order; however, because of the small size of the continuity counter it only detects a small percentage of the possible discontinuities. MPEG-2 TS is commonly used in delivering Motion Imagery Data over IP as well. The MISP has standardized how to insert MPEG-2 TS packets into UDP packets in MISB ST 1402 [8]. Table 5-3 describes the effects of UDP errors on the MPEG-2 TS.

Table 5-3: MPEG-2 TS Error Types

Error Type	Description
Packet Loss	There are several types of packet loss in MPEG-2 TS. The first occurs when one (or more) UDP packet(s) are discarded. Up to seven MPEG-2 TS packets can be encapsulated into one UDP packet. Loss of one UDP packet can mean the loss of up to seven MPEG-2 TS packets. Such a loss can be detrimental to the decompression of the Motion Imagery, and the effects range from a Decoder that may stop working to intermittent losses of imagery. This significantly impacts Exploitation. A second type of packet loss is more localized to an individual MPEG-2 TS packet. Here, the internal information within a packet may be incorrect; this could result from a malfunctioning system component, or corruption in transmission. A packet could be discarded by receiving equipment if the error is an ill-formed packet. Depending on the contents of a discarded packet the effect could be major (i.e. timing or important decode information) or minor (i.e. loss of a portion of the imagery). In both types of packet loss, when a packet contains Metadata the information is likely unrecoverable.
Packet Corrupted	A MPEG-2 TS packet may be corrupted by a system issue, such as Encoder or Transport Stream multiplexer malfunction or information within a packet can become corrupted in transit. The packet may appear to be properly formed – and therefore not discarded, but the data contained inside is not meaningful. Issues like these are not easily diagnosed.
Packet Out of Order	As discussed in Table 5-2, out-of-order packets generally result from network device operation and varied network paths data may take. The 4-bit continuity count in each MPEG-2 TS packet provides a limited indication of packet sequence order; however, without information in advance on how many MPEG-2 TS packets are in a UDP packet it may be difficult to determine the actual MPEG-2 TS packet order.

5.1.2.2 Real Time Protocol (RTP)

RTP [9] is designed for end-to-end, real-time transfer of data. RTP was specifically designed for delivery of A/V (Audio/Video) services over IP. Each data type (i.e. Motion Imagery, Metadata, and Audio) is delivered as an independent data stream. Relational timing information for synchronizing the individual data streams at a receiver is published in Real Time Control Protocol (RTCP) – a companion protocol. RTP is encapsulated in UDP/IP, and includes a timestamp for synchronization and sequence numbers that aide in packet loss and reordered packet detection. RTP/RTCP is typically considered for bandwidth-constrained environments, where a choice among the supported data types can be made.

There are also advantages to encapsulating a MPEG-2 TS into RTP; in fact, this method is widely used by the commercial industry in long-haul delivery of video over IP. A receiver can use the RTP timestamp to measure inter-packet jitter to estimate the stress occurring in a network. Such information also indicates potential Decoder buffer over/under flows, which could cause Decoder failure. The RTP sequence number is 16-bits; much larger than the MPEG-2 TS 4-bit packet count, which enables a wider detection range for lost and reordered packets.

Delivery of Motion Imagery Data using RTP is subject to the errors similarly found in MPEG-2 TS (see Table 5-3). In RTP, however, data is organized into larger packets, which can be as large as the limits (called the Maximum Transmission Unit, or MTU) of UDP for a medium. A lost packet of RTP may have a greater effect than a lost packet of MPEG-2 TS as it contains more data; again, the impact to decompression and Exploitation would depend on the data contained within the packet.

Chapter 6

Time Systems

6.1 Overview

Time is fundamental within Motion Imagery Systems (MIS): it is used to coordinate subsystems and components internally, as well as for coordination among external systems; it also enables people to share and operate on data collaboratively. A **Temporal Event** is when time, in some form, is associated with an event. The action of assigning a time to an event is called **Timestamping**; the time value assigned is called a **Timestamp**. Measurements within a MIS while imaging a scene (i.e. "frame-time"), or actions within a scene being exploited by an analyst are examples of temporal events. Temporal events can be instantaneous or have finite duration.

Temporal Event: When time is associated with an event. The time can be a duration, defining a Start Time and End Time, or instantaneous, defining a Start Time (Note: Start Time equals End Time, in this case).

Timestamp: The time value(s) associated with a Temporal Event.

Timestamping: The action of assigning time to a Temporal Event.

Correlation of temporal events is critical for proper interpretation of activities within a scene. Analysis is enhanced when information and metadata across multiple systems are synchronized temporally. Although many systems for tracking time have been developed, the Motion Imagery Standard Profile defines its Time System based upon the international Time Systems.

6.2 Time System Elements

Timestamps depend on three elements of a Time System: **Clock**, **Epoch** and **Adjustment Criteria**.

Within this handbook, a **Clock** is a function that counts *units* of time, for example a second, minute, hour, etc. The Clock *value* is the number of units counted, which can include fractional units. The *clock-period* is the amount of time between two successive unit counts, measured in Standard International (SI) Seconds [10] (i.e. SI Second).

Depending on the type of Clock, the clock-period can be constant or vary. A Clock with a constant clock-period is called a *linear-clock*. A linear-clock with a clock-period equal to the SI Second is called a SI-clock. A *varying-clock* has a non-constant clock-period.

The **Epoch** is the reference point for the Clock, which includes the beginning date and time. Clock values represent a count of units since some defined Epoch.

Figure 6-1 illustrates a Clock using a number line. In the left diagram, each tick mark represents the Clock value as the count of seconds since the Epoch and the clock-period is indicated as the space between each tick mark. The middle and right diagram illustrates the difference between a linear-clock and varying-clock.

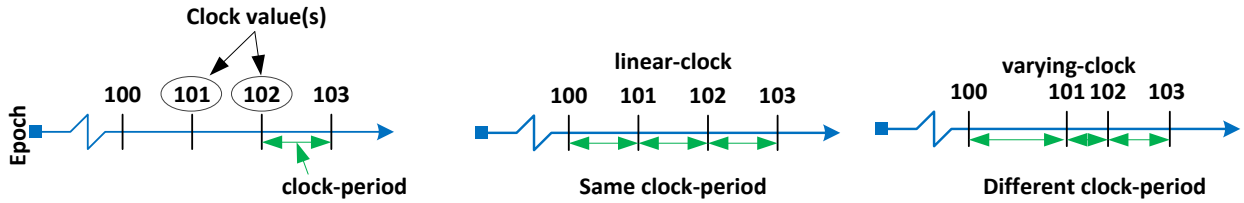


Figure 6-1: Illustration of Clocks

The **Adjustment Criteria** are processes or rules used to change a Time System's Clock value to match some given criteria. An example Adjustment Criteria is to change a Clock's value daily to match the time it takes for the earth to rotate (i.e. UT1 time system in Section 6.2.2.2).

The relationship between two Clocks can be described in terms of the Clock's clock-period and differences in when each clock increments its count. The smallest difference in time between one Clocks count incrementing, and a second Clocks count incrementing is called *delay*. Figure 6-2 illustrates the four primary cases of delay between two Clocks. Case (1) shows two linear-clocks with the same period, and the delay, d_0, is constant. Case (2) shows two linear-clocks with different periods; in this case the two delays (d_1 and d_2) are not the same. Case (3) shows two delays (d_3 and d_4) –of many possible different delays – between a linear-clock and varying-clock. Case (4) shows two delays (d_5 and d_6) of many possible different delays between two varying-clocks.

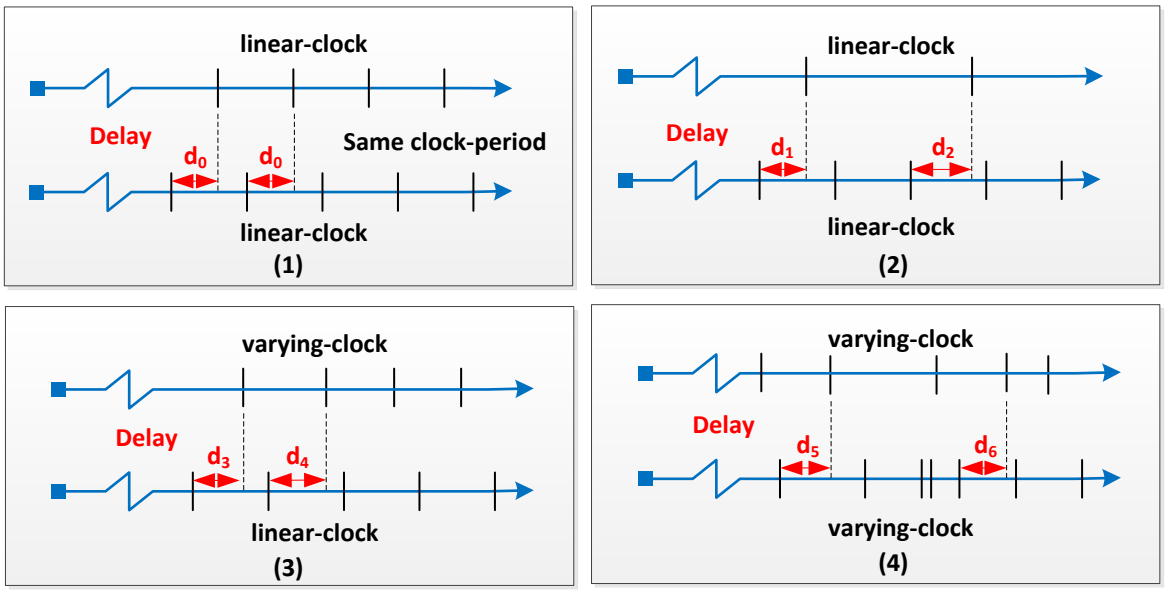


Figure 6-2: Illustration of delay between two clocks count increments.

Two Clocks that have the same clock-period and the delay is zero are called *locked*. Two Clocks that have the same clock-period and the delay is not zero are called *lock-delayed*. In the lock-delayed case the delay is a constant value, see Figure 6-2, case (1).

Clocks that are locked increment their count of units at the exact same instant of time. Such an example is a wall clock which increments its second hand at the same instant as the second hand

on a wrist watch (i.e. they "tick" at the same time). In this example, when the wrist watch ticks at the same clock-period but is slightly behind, the two clocks are lock-delayed.

Locked, however, does not guarantee that the time reported on each is identical; the watch's second hand may be pointing to the 10th second, while the wall clock is pointing to the 49th second. The difference in reported time between any two Clocks at the same instant is the o*ffset*. Two Clocks are *synchronized* when they are locked and have no offset between them; these Clocks then report the same time for all time values. Figure 6-3 illustrates these relationships using linear clocks.

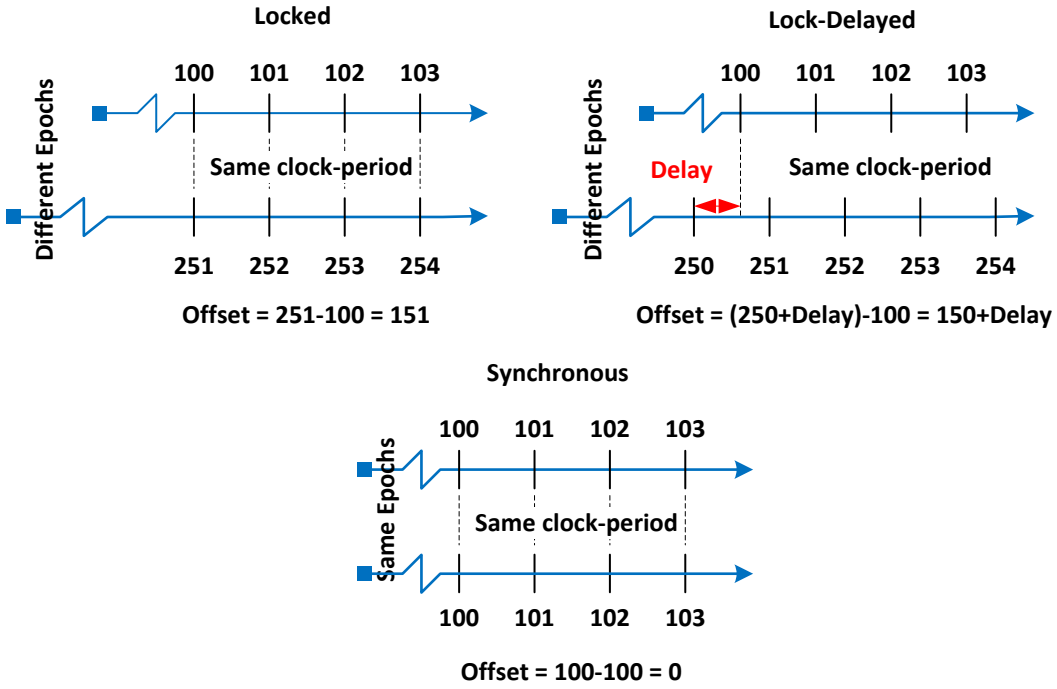


Figure 6-3: Illustration of Clock Relationships

Clock-period: The amount of time between each successive count of a single Clock, measured in SI Seconds.

Delay: The smallest difference in time between one clocks count incrementing and a second clocks count incrementing.

Locked: When two Clocks have the same clock-period and zero delay.

Lock-delayed: When two Clocks have the same clock-period and non-zero delay.

Offset: The difference in Clock values between any two clocks at the same instant.

Synchronized: When two Clocks are locked, and have zero offset.

Note, in the following discussion a Clock's "unit of time" is called a "second," which is not necessarily a fixed-length SI second.

Two methods are used to represent a Timestamp: Second-Count and Date-Text.

Second-Count: A count of seconds (and fractions of seconds) from an Epoch.

Date-Text: A date/time text value (year, month, day, hours, minutes, seconds, and fractions of seconds). Date-Text is a measure of time since the beginning of the Common Era (CE); that is, the Epoch of Date-Text is midnight January 1, year zero (0000-01-01T00:00:00.0Z).

Although there were several different calendars and methods of time keeping established throughout history, this document's equations and algorithms compute time as if the current method and calendar has been in place since the year zero Epoch.

6.2.1 Timing System Capability Levels

Three hierarchical capability levels for timestamping temporal events are defined: **Level 1 - Total Ordering**, **Level 2 - Relative Differencing** and **Level 3 - Absolute Time**. A fourth level defines an absence of the other capabilities, **Level 0 – Inconsistent Timing**.

Level 0 – Inconsistent Timing, a below baseline timing system that does not provide Level 1, 2 or 3 capabilities. With Inconsistent Timing, random temporal jumps, either forward or backward, prevent Total Ordering.

Level 1 - Total Ordering, a baseline capability of a timing system provides Total Ordering of Temporal Events. Total ordering means the timestamp for every Temporal Event provides the order in which the events occur. As an example, using a different Clock for metadata timestamping than that for the imager may not produce Total Ordering of the events; in this case it may not be possible to know if metadata came before, during or after the image – Total Ordering is not guaranteed. Total Ordering only determines the order of Temporal Events; it does not imply that the timestamp is related to any specific Absolute Time. Total Ordering also enables the indexing (the ability to quickly find Temporal Events in a list) of Temporal Events.

Level 2 - Relative Differencing capability builds upon Level 1 by adding the ability to compute the difference in time, using SI Seconds, between two timestamps. Accurately determining differences between Temporal Events enables durations to be computed for events which span any amount of time.

Level 3 - Absolute Time capability builds upon Level 2 by generating timestamps in relation to a known universal Absolute Time reference. This enables Temporal Events outside of the MIS to be coordinated with information produced from the MIS.

6.2.2 International Time Systems

Many standard time systems are in use worldwide. Five of these time systems are discussed to aide understanding the time system mandated by the MISP: TAI, UT1, UTC, GPS and POSIX Time. All but POSIX Time use the SI-clock. POSIX is included in this section only for historical reasons.

6.2.2.1 International Atomic Time (TAI)

International Atomic Time employs a number of atomic clocks to denote the passage of time by counting the number of SI Seconds from the TAI Epoch, 1958-01-01T00:00:00.0Z[1]. TAI is a SI-clock system where the duration of minutes, hours, and days does not change from 60, 3600 and 86,400 SI Seconds, respectively. TAI is a Level 3 time system as defined in Section 6.2. TAI does not have any Adjustment Criteria.

TAI can be represented using Second-Count of SI seconds since the Epoch, or with Date-Text. When converting to/from Date-Text leap days must be added (or subtracted) as needed (see Section 6.2.5)

As is discussed in Section 6.2.2.2, the length of a mean solar day is not exactly 86,400 SI seconds; therefore, the start of a TAI "day" drifts away from the mean solar day. When a resulting Date-Text is computed from a TAI Second-Count there will be a difference between the TAI Date-Text and UT1 or UTC Date-Text.

6.2.2.2 Universal Time 1 (UT1)

Universal Time 1 measures time by observing the length of a mean solar day in SI Seconds. A mean solar day is the period of time for the sun to move from noon-to-noon between two days. Adjustment Criteria are performed daily based on the Length of Day (LOD). The LOD can change based on planetary wobble and other effects (tidal and lunar). With UT1, the LOD is 86,400±Δ SI Seconds, where Δ is a small number in the milliseconds range. UT1 is represented *only* using Date-Text. At the end of each day, regardless of the Δ value, the Clock is reset to zero hundred hours (00:00:00.0). At the reset time, there is either a potential overlap between days, or a gap in time between the two days, in terms of the Date-Text. Figure 6-4 illustrates a timeline with Day 1 LOD 0.5 second *longer* than 86,400 seconds (shown as 23:59:60.5). Day 2 then effectively begins with an overlap of Day 1 by 0.5 second; the Date-Text for Day 1 that is greater than 23:59:60 "overlaps" with Day 2. If Day 2 LOD is 0.5 second *shorter* than 86,400 seconds (i.e. 23:59:59.5), there will be a gap between Day 2 and Day 3.

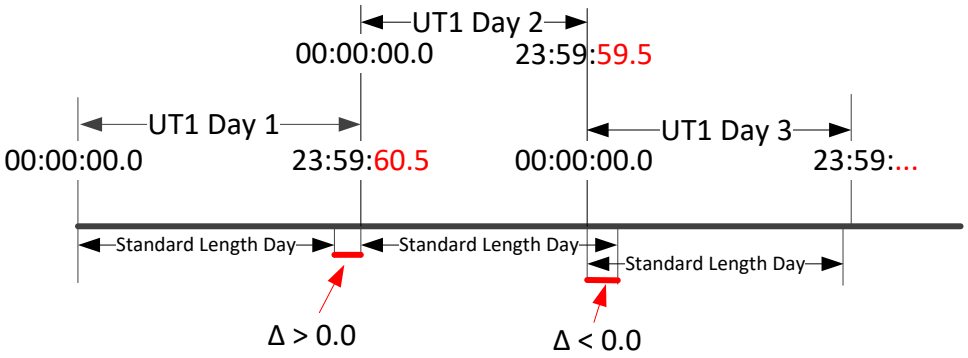


Figure 6-4: Illustration of UT1 LOD and the overlap and gaps that can occur.

[1] At the time TAI was developed UTC did not exist as a standard, so the Epoch is measured relative to Universal Time.

The LOD difference implies that UT1 and TAI are not locked even though both systems count SI Seconds. The Epoch for UT1 is year zero, since UT1 is only represented in Date-Text.

Since the LOD is constantly changing, UT1 is a Level 1 time system. Total Ordering is possible when comparing the UT1 Date-Text; however, Relative Differencing between times from two different days requires knowing the LOD of all days between the differenced times. TAI time can slowly diverge from the UT1 time because of LOD difference between TAI and UT1. The advantage of UT1 is that it is locked with the mean solar day.

6.2.2.3 Coordinated Universal Time (UTC)

UTC is the international time standard used extensively within international communities. Coordinated Universal Time combines the constancy of TAI with the UT1 LOD information. UTC has an Epoch of 1972-01-01T00:00:00.0Z, and it is Locked with TAI, but not synchronized. UTC's Adjustment Criterion is to add or remove "leap seconds" to keep alignment to UT1 to within ±0.9 SI Seconds. A leap second is a SI Second added or subtracted from UTC on a designated month as determined by U.S. and international standards bodies; since the UTC Epoch there have been a number of leap seconds added to UTC. Table 6-1 lists the start and end dates/times for when the given leap seconds are in effect. The Start Date/Time values are inclusive in the time range; however, the End Date/Time values are exclusive in the time range.

Table 6-1: Leap Seconds since January 1972

Start Date/Time (Inclusive)	End Date/Time (Exclusive)	Leap Seconds Offset	Comments
1972-01-01T00:00:00	1972-07-01T00:00:00	10	Original offset from TAI
1972-07-01T00:00:00	1973-01-01T00:00:00	11	1 leap second added
1973-01-01T00:00:00	1974-01-01T00:00:00	12	1 leap second added
1974-01-01T00:00:00	1975-01-01T00:00:00	13	1 leap second added
1975-01-01T00:00:00	1976-01-01T00:00:00	14	1 leap second added
1976-01-01T00:00:00	1977-01-01T00:00:00	15	1 leap second added
1977-01-01T00:00:00	1978-01-01T00:00:00	16	1 leap second added
1978-01-01T00:00:00	1979-01-01T00:00:00	17	1 leap second added
1979-01-01T00:00:00	1980-01-01T00:00:00	18	1 leap second added
1980-01-01T00:00:00	1981-07-01T00:00:00	19	1 leap second added
1981-07-01T00:00:00	1982-07-01T00:00:00	20	1 leap second added
1982-07-01T00:00:00	1983-07-01T00:00:00	21	1 leap second added
1983-07-01T00:00:00	1985-07-01T00:00:00	22	1 leap second added
1985-07-01T00:00:00	1988-01-01T00:00:00	23	1 leap second added
1988-01-01T00:00:00	1990-01-01T00:00:00	24	1 leap second added
1990-01-01T00:00:00	1991-01-01T00:00:00	25	1 leap second added
1991-01-01T00:00:00	1992-07-01T00:00:00	26	1 leap second added
1992-07-01T00:00:00	1993-07-01T00:00:00	27	1 leap second added
1993-07-01T00:00:00	1994-07-01T00:00:00	28	1 leap second added
1994-07-01T00:00:00	1996-01-01T00:00:00	29	1 leap second added
1996-01-01T00:00:00	1997-07-01T00:00:00	30	1 leap second added
1997-07-01T00:00:00	1999-01-01T00:00:00	31	1 leap second added
1999-01-01T00:00:00	2006-01-01T00:00:00	32	1 leap second added
2006-01-01T00:00:00	2009-01-01T00:00:00	33	1 leap second added
2009-01-01T00:00:00	2012-07-01T00:00:00	34	1 leap second added
2012-07-01T00:00:00	2015-07-01T00:00:00	35	1 leap second added
2015-07-01T00:00:00	2017-01-01T00:00:00	36	1 leap second added
2017-01-01T00:00:00	Unknown	37	1 leap second added

From 1961-01-01 to 1972-01-01, UTC used the equivalent of fractional leap seconds that would be added daily. Equation 2 is used to compute the leap second offset (TAI-UTC) for a given day D. D is used to look up values B, S and R from Table 6-2.

	$Leap\ Second = \ TAI - UTC = B + (D - S) * R$	Equation 2

Where:

B	Offset in seconds. A lookup value from Table 6-2 based on D
D	Day for which leap seconds are accounted in Modified Julian Days (MJD).
S	TAI Reference Date in MJD. A lookup value from Table 6-2 based on D.
R	Rate factor. A lookup value from Table 6-2 based on D.

For MJD conversions there are many internet web tools.

Table 6-2: Leap Second Computation for Dates Ranging from 1961-01-01 to 1972-01-01. Data derived from U.S. Navy Observatory file (ftp://maia.usno.navy.mil/ser7/tai-utc.dat)

Start Date (Inclusive)	End Date (Exclusive)	B	S	R
1961-01-01	1961-08-01	1.4228180	1961-01-01	0.001296
1961-08-01	1962-01-01	1.3728180	1961-01-01	0.001296
1962-01-01	1963-11-01	1.8458580	1962-01-01	0.0011232
1963-11-01	1964-01-01	1.9458580	1962-01-01	0.0011232
1964-01-01	1964-04-01	3.2401300	1965-01-01	0.001296
1964-04-01	1964-09-01	3.3401300	1965-01-01	0.001296
1964-09-01	1965-01-01	3.4401300	1965-01-01	0.001296
1965-01-01	1965-03-01	3.5401300	1965-01-01	0.001296
1965-03-01	1965-07-01	3.6401300	1965-01-01	0.001296
1965-07-01	1965-09-01	3.7401300	1965-01-01	0.001296
1965-09-01	1966-01-01	3.8401300	1965-01-01	0.001296
1966-01-01	1968-02-01	4.3131700	1966-01-01	0.002592
1968-02-01	1972-01-01	4.2131700	1966-01-01	0.002592

Example 6-1: Computation of Leap Second for 1970-01-01T00:00:00.0

D = MJD(1970-01-01T00:00:00.0) = 40587.0

From last row in Table 6-2:

B = 4.2131700

S = MJD(1966-01-01T00:00:00.0) = 39126.0

R = 0.002592

$$Leap\ Second = \ TAI - UTC = B + (D - S) * R = \ 4.21317 + (40587 - 39126) * 0.002592$$

$$Leap\ Second = 8.000082$$

From 1972-01-01 and on, leap second additions and subtractions to UTC Date-Text are described in ITU-R TF.460-6 [11]:

> *2.1 - A positive or negative leap-second should be the last second of a UTC month, but first preference should be given to the end of December and June, and second preference to the end of March and September.*
> *2.2 - A positive leap-second begins at 23h 59m 60s and ends at 0h 0m 0s of the first day of the following month. In the case of a negative leap-second, 23h 59m 58s will be followed one second later by 0h 0m 0s of the first day of the following month.*

Figure 6-5 illustrates the addition and subtraction of leap seconds. The top illustration shows the addition of a leap second by adding a 61st second to the last minute of the last day of a UTC month; the time increments from 59 to 60 seconds and then to 00 seconds. The bottom illustration shows the removal of a second from the last minute of the last day of a UTC month; the time increments from 58 seconds to 00 seconds (of the next day), the 59th second is skipped.

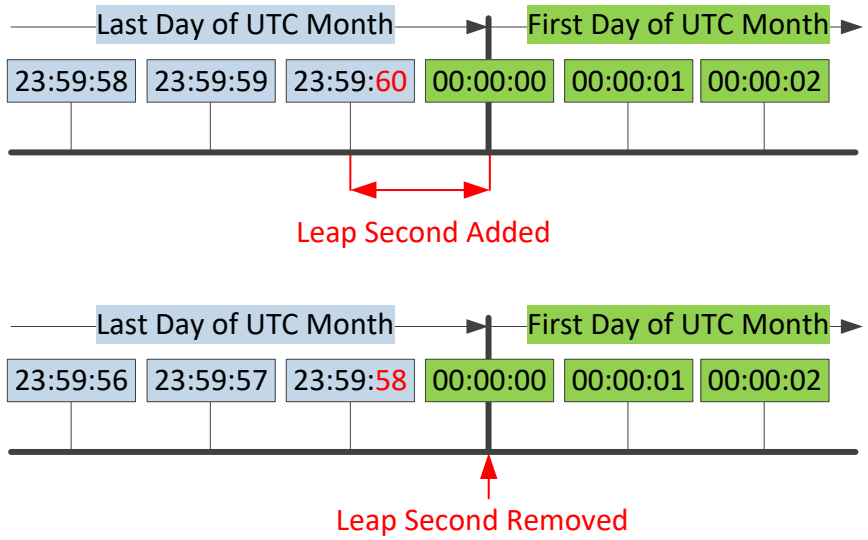


Figure 6-5: Illustration of Leap Seconds Added or Removed from UTC and the associated Date-Text

Although UTC is represented only using Date-Text, it can be converted to a Second-Count from any given Epoch occurring after the UTC Epoch – provided the number of leaps seconds are known and included for the specific date and Epoch.

Since leap seconds are added or removed UTC does not meet any of the capability levels as defined in Section 6.2; Total Ordering is not possible beyond the day. Since there is the possibility of added leap seconds, Relative Differencing between times from two different days requires knowing the leap seconds between the differenced times.

UTC is offset from TAI with a defined integer number of SI Seconds plus leap seconds.

6.2.2.4 Global Positioning Time (GPS)

The GPS time system has its Clock locked with the TAI Clock with an Epoch of 1980-01-06T00:00:00.0Z (UTC). For Date-Text, GPS is equivalent to TAI with an offset of 19 SI Seconds. For Second-Count, GPS is equal to TAI with an offset of the GPS Epoch Second-Count plus the 19 SI Seconds (see Table 6-4). GPS time does not have any Adjustment Criterion. GPS is a capability Level 3 time system as defined in Section 6.2.

The GPS system relies on a collection of Atomic time clocks and provides position information enabling geo-location anywhere on or above the earth. The GPS system consists of a constellation of satellites orbiting the earth, where each satellite provides location and time information to GPS receivers. Many GPS receivers produce multiple time systems including UTC time. GPS supplies a message, which contains UTC corrections that allows a receiver to convert GPS Time into UTC or any time zone. This message includes the time difference in whole seconds between GPS time and UTC.

6.2.2.5 POSIX Time

Historically, POSIX time was the basis of the time systems used in older versions of the MISP and supporting documents. However, because of vagueness in the POSIX standard about if and how to adjust for UTC leap-second changes, the POSIX time system is no longer used as the basis for the MISP. The lack of clarity led to different interpretations and implementations. One interpretation included UTC leap second corrections, while the other (originally intended) interpretation did not include UTC leap second corrections. The information in this section is included for historical and reference only.

POSIX Time is a time system developed in POSIX IEEE Std. 1003.1 [12]. POSIX Time is defined as the Second-Count (along with a nanosecond counter) since the Epoch of 1970-01-01T00:00:00.0Z (UTC[2]) and does not include leap seconds. The duration of its second is not specified, but it is nominally equal to a SI Second; therefore, POSIX is not locked to TAI. POSIX Time does not have any Adjustment Criterion. The POSIX standard includes an algorithm for converting the Second-Count to Date-Text (called "broken-down time") as UTC, but this conversion does not include leap seconds, so the relationship between UTC and Second-Count is noted as unspecified in the standard. Because the value of the second is not mandated to be an SI second, the POSIX Time system is not locked with TAI; thus, it is a Level 1 time system as defined in Section 6.2.

POSIX-compliant systems provide methods for operating with POSIX Time and converting to/from broken-down time. If the appropriate values are added for leap seconds, POSIX-based systems provide a toolset for converting between UTC and POSIX Time.

6.2.3 MISP Time System

The MISP Time System clock counts SI-Seconds since the Epoch of 1970-01-01T00:00:00.0Z (UTC[3]). The MISP Time System is locked with TAI with a Leap Second offset of 8.000082 seconds (see Example 6-1 for computation). The MISP Time System does not include UTC leap-

[2] Leap second offset is computed using Equation 2

[3] Leap second offset is computed using Equation 2

seconds. The Epoch date is based on UNIX time (also known as POSIX time). The MISP Time System is essentially equivalent to the UNIX or POSIX time (without UTC leap second corrections) documented within prior versions of MISP standards, recommended practices and engineering guidelines. Historically, the fractional 82 microseconds were not included in the offset computation; existing systems did not need this degree of accuracy. However, in future systems, where more precise timing may be required, the 82-microsecond offset may be important when comparing measurements against TAI.

The MISP time system defines the **Precision Time Stamp** to represent the value of the count of the number of microseconds (based on the SI Second) since the UNIX Epoch time of 1970-01-01T00:00:00.0Z (UTC[4]). The MISP time system is a Level 3 time system as defined in Section 6.2.

The MISP time system also defines the **Nano Precision Time Stamp** to represent the value of the count of the number of nanoseconds (based on the SI Second) since the UNIX Epoch time of 1970-01-01T00:00:00.0Z (UTC[5]). The MISP time system is a Level 3 time system as defined in Section 6.2.

6.2.4 Time Systems Summary

Table 6-3 lists the time systems and their properties.

Table 6-3: List of Time Systems

Time System	Clock Type	Epoch	Adjustment Criteria
TAI	SI-clock	1958-01-01T00:00:00.0Z	None
UT1	SI-clock	0000-01-01T00:00:00.0Z	Daily Length of Day Adjustments
UTC	SI-clock	1972-01-01T00:00:00.0Z	Potential Monthly Leap Second Adjustments[6]
GPS	SI-clock	1980-01-06T00:00:00.0Z	None
POSIX	varying-clock	1970-01-01T00:00:00.0Z	None
MISP	SI-clock	1970-01-01T00:00:00.0Z	None

Figure 6-6 shows the relationship of the various time systems. This figure does not include UT1, because Motion Imagery applications do not directly use it.

[4] Use Equation 2 to compute the Leap second offset

[5] Use Equation 2 to compute the Leap second offset

[6] While it is possible to adjust UTC with the addition or subtraction of one leap second at the end of each month, June and December are the preferred months to make this change. Historically, there has never been a need to make an adjustment outside of these two months.

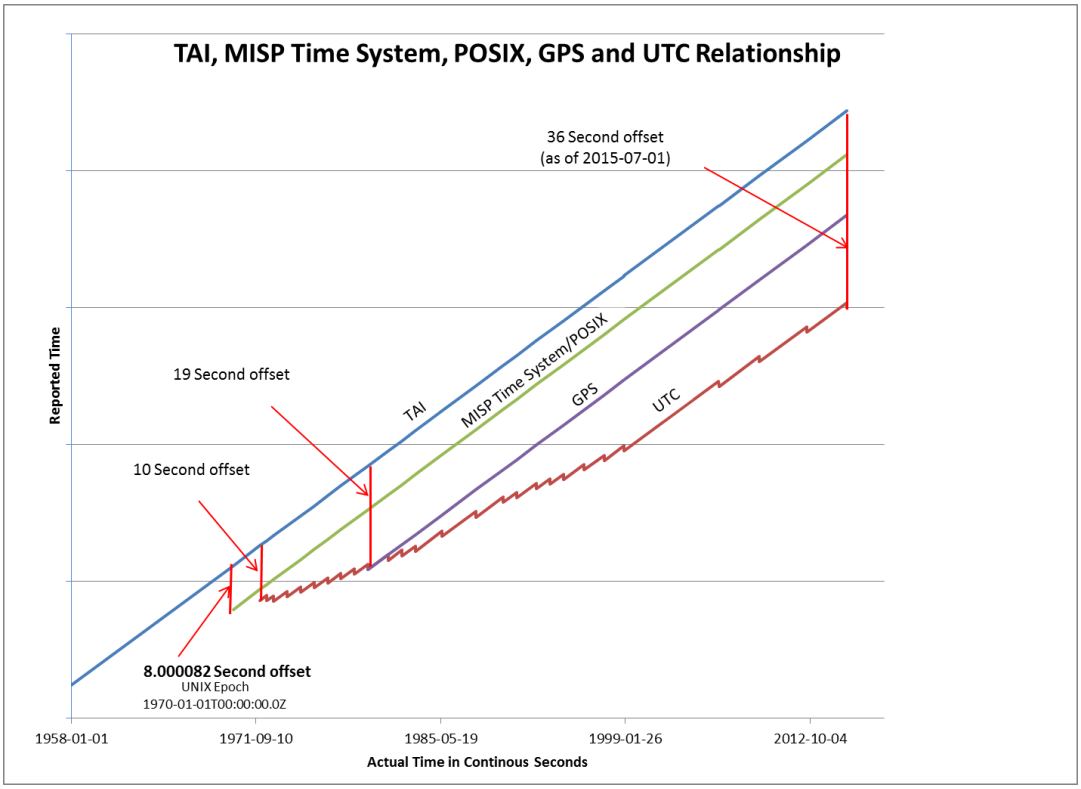


Figure 6-6: Relationships among Time Systems

Figure 6-6 shows TAI, POSIX (without UTC leap second corrections), MISP and GPS are all linear time systems with constant offsets from each other; this enables straightforward time conversions between these systems. UTC is piecewise linear; thus, the conversion of UTC to/from any other time systems requires a table of Offsets for each given time period where the Offset was in effect.

6.2.5 Time Conversions

Converting from one time system to another is performed with the equations in Table 6-4. The values denoting the various time systems T, G and M are in Second-Counts and U is in Date-Text.

Table 6-4: Time System Conversions

		To			
		TAI	UTC	GPS	MISP
From	TAI (T)		ComputeUTC(T)	$T - G_{Epoch}$	$T - M_{Epoch}$
	UTC (U)	ComputeTAI(U)		ComputeTAI(U) - G_{Epoch}	ComputeTAI(U) - M_{Epoch}
	GPS (G)	$G + G_{Epoch}$	ComputeUTC($G + G_{Epoch}$)		$G + G_{Epoch} - M_{Epoch}$
	MISP (M)	$M + M_{Epoch}$	ComputeUTC($M + M_{Epoch}$)	$M + M_{Epoch} - G_{Epoch}$	
Where M_{Epoch} and G_{Epoch} are offsets to the TAI Epoch: M_{Epoch} = Seconds(MISPEpoch + 8.000082, TAIEpoch) G_{Epoch} = Seconds(GPSEpoch + 19, TAIEpoch)					

This table does not include UT1 because it is not used directly in Motion Imagery applications.

6.2.5.1 UTC Conversions

When converting to UTC care must be taken near the leap second boundaries; the following algorithm is suggested:

Compute UTC Date-Text from TAI Second-Count (see Appendix C for Pseudocode notation)
(Note: Notional and untested)

```
Function ComputeUTC(TAI)
        TAI_date = DateText(TAI, TAIEpoch)
        //The following provides the range values (Start, End and Offset) for the leap second range before
                TAI_date. LeapTable is Table 6-1.
        R_before= LeapTable (TAI_date, "Before")
        //The following provides the range values for the leap second range after TAI_date
        R_after= LeapTable (TAI_date, "After")
        //The following provides the range values for the leap second range for TAI_date
        R = LeapTable(TAI_date)
        Period = 60 //This is the period to use when adding the leap second offset.
        LeapSeconds = R.Offset
        If (TAI_date-R_before.Offset is before to R_before.End) then
                If (R_before.Offset < R.Offset) then
                        Period = 61 //Leap second added
                Else
                        Period = 58 //Leap second removed
                End
                LeapSeconds = R_before.Offset
        Else if (TAI_date+R_after.Offset is after or equal to R_after.Start) then
                If (R.Offset<R_after.Offset) then
                        Period = 61 //Leap second added
                Else
                        Period = 58 //Leap second removed
                End
                LeapSeconds = R_after.Offset
        End
        UTC = TAI_date - LeapSeconds using modulus Period

        //The result is a UTC time that can have either have "60" for the second's value or 58 as the last second
                if the TAI time occurs when a leap second was added or removed.
End
```

When converting from UTC to TAI the algorithm is:

Compute TAI Second-Count from UTC Date-Text (see Appendix C for Pseudocode notation)
(Note: Notional and untested)

```
Function ComputeTAI(UTC)
        //The following provides the range values (Start, End and Offset) for the leap second range for UTC
        date. LeapTable is Table 6-1
        R = LeapTable (UTC)
        TAI_date = UTC + R.Offset
        TAI = Seconds(TAI_date , TAIEpoch)
End
```

6.2.5.2 Date-Text to Second-Count Conversion

The following algorithm converts Date-Text to a number of seconds since the given Epoch. This algorithm includes the computation of leap-years, but not UTC leap seconds (see Section 6.2.5.1 for UTC conversion).

Compute Second-Count from given Date-Text from given Epoch. (see Appendix C for Pseudocode notation)
(Note: Notional and untested)

```
Function Seconds(DateText, Epoch)
        SecondsSinceCE = SecondsCE(DateText)
        EpochSecondsCE = SecondsCE(Epoch)
        Seconds = SecondsSinceCE - EpochSecondsCE;
End
```

The following is a utility function for converting any Date-Text to Second-Count from start of Common Era (CE) i.e. year zero.

Compute Second-Count for given Date-Text (Year-Month-Day-Hours:Minutes:Seconds, seconds includes fractions of seconds) from Start of CE. (see Appendix C for Pseudocode notation)
(Note: Notional and untested)

```
Function SecondsCE(DateText)
        LeapYear = (Year%4==0) + (Year%400==0) - (Year%100==0)
        DayInYear = (Month>1)*31
        DayInYear += (Month>2)*28 + LeapYear
        DayInYear += (Month>3)*31
        DayInYear += (Month>4)*30
        DayInYear += (Month>5)*31
        DayInYear += (Month>6)*30
        DayInYear += (Month>7)*31
        DayInYear += (Month>8)*31
        DayInYear += (Month>9)*30
        DayInYear += (Month>10)*31
        DayInYear += (Month>11)*30
        DayInYear += DayInMonth
        SecondsSinceCE = Year*31,536,000 + DayInYear*86,400 + Hours*3600 + Minutes*60 + Seconds +
                Floor(Year/4) + Floor(Year/400) - Floor(Year/100)
        //Note: SecondsInDay = 24*3600 = 86,400
        //Note: SecondsInYear = 365*24*3600 = 31,536,000
End
```

6.2.5.3 Second-Count to Date-Text Conversion

The following algorithm converts a Second-Count for a given Epoch to Date-Text.

Compute Date-Text (Year-Month-Day-Hours:Minutes:Seconds, seconds includes fractions of seconds) from Second-Count since start of Epoch. (see Appendix C for Pseudocode notation)
(Note: Notional and untested)

```
Function DateText(Seconds, Epoch)
        SecondsCEOffset = SecondsCE(Epoch)+Seconds
        Days = SecondsCEOffset/86,400
        Year = Floor(Days)/365.2425
        DayInYear = Days – Floor(Year)*365.2425
        L = (Floor(Year)%4==0) + (Floor(Year)%400==0) - (Floor(Year)%100==0)
        //Accumulated days throughout year including leap day if its leap year.
        MonthDays = [0, 31, 28+31+L, 31+28+L+31, 31+28+L+31+30, 31+28+L+31+30+31, 31+28+L
        +31+30+31+30, 31+28+L+31+30+31+30+31, 31+28+L+31+30+31+30+31+31,
        31+28+L+31+30+31+30+31+31+30,
        31+28+L+31+30+31+30+31+31+30+31,31+28+L+31+30+31+30+31+31+30+31+30]
        //Zero based month number
        Month = (DayInYear>MonthDays[1]) + (DayInYear> MonthDays [2]) +
                (DayInYear> MonthDays [3]) + (DayInYear> MonthDays [4]) +
                (DayInYear> MonthDays [5]) + (DayInYear> MonthDays [6]) +
                (DayInYear> MonthDays [7]) + (DayInYear> MonthDays [8] ) +
                (DayInYear> MonthDays [9] ) + (DayInYear> MonthDays [10]) +
                (DayInYear> MonthDays [11])
        DayInMonth = DayInYear - MonthDays[Month]
        Hours = DayInMonth/24 – Floor(DayInMonth/24)
        Minutes = Hours/60 – Floor(Hours/60)
        Seconds = Minutes/60 - Floor(Minutes/60)

        DateText=Floor(Year) "-" Floor(Month+1) "-" Floor(DayInMonth) "T" Floor(Hour) ":" Floor(Minute)
        ":" Seconds
End
```

6.2.6 Time Sources

There are many different time sources, and each provides some degree of precision and accuracy. Time sources can be categorized into three types: **Atomic**, **Independent** and **Hybrid**.

Atomic Time Sources (ATS) are second-by-second synchronized (or better) with TAI time; however, the Epoch of the time source may differ from TAI. An example of an ATS is time output by a GPS receiver. An **Independent** time source is not synchronized with TAI, which may or may not count seconds at a constant rate (i.e. a varying-clock). An example of an independent time source is a non-networked computer using an internal clock as its time reference. A **Hybrid** time source operates as an independent source, but is periodically synchronized to an Atomic time source. In the period of time until resynchronization, a Hybrid time source is said to be *free-wheeling*. An example Hybrid time source is a GPS receiver connected to a computer; the GPS provides a periodic time signal update, which the computer synchronizes to, and then free-wheels until the next GPS update. During the free-wheeling period the clock may change its absolute relation to its reference source; this is one form of what is called *clock drift*.

6.2.6.1 GPS System

GPS provides an Atomic time source and position information enabling geo-location anywhere on or above the earth. The GPS system consists of a constellation of satellites orbiting the earth, where each satellite provides location and time information to GPS receivers.

Many GPS receivers output time information along with a one pulse per second (1PPS) synchronization signal. This time information may be in a variety of formats (UTC, GPS, etc.) and is usually accurate only to the SI Second. The 1PPS synchronization signal enables Hybrid time sources to be built for Motion Imagery Systems. A 1 PPS signal enables sub-second finer gradations of time (i.e. microseconds) to be derived by phase locking a high frequency (e.g. 1 MHz) clock to the signal. Some GPS receivers output an Inter-Range Instrumentation Group (IRIG) Standard 200 time signal from which times to the SI Second and time to the sub-second can be derived.

6.2.6.1.1 - GPS Time to UTC Conversion

Some receivers provide only GPS Week and GPS Seconds parameters. The offset of GPS Seconds is defined relative to the beginning of the current GPS week. GPS time is referenced to a UTC zero-time point originally defined as midnight (00:00 UTC) before the morning of 1980-01-06. The GPS Week parameter is 10 bits, which is modulo 1024, so the GPS week cycle is 1024 weeks (7168 days, or 19+ years). The latest zero-time point was 1999-08-22 00:00 GPS time (more modern GPS navigation systems use a 13-bit field that repeats every 8,192 weeks.) The following algorithm provides for calculation of the date and time to within one second (further precision may require provisions such as a local clock reference synchronized to the GPS signal):

Formula:

UTC = GPS – leap seconds

Since,	GPS	=	GPS Week + GPS Seconds + 1999-08-22 00:00
Then,	UTC	=	(GPS Week + GPS Seconds) + 1999-08-22 00:00 – leap seconds
		=	GPS Week + (GPS Seconds – leap seconds) + 1999-08-22 00:00

Algorithm:

Compute UTC Date-Text (Year-Month-Day-Hours:Minutes:Seconds, seconds includes fractions of seconds) from GPS. (see Appendix C for Pseudocode notation)
(Note: Notional and untested)

```
// If (GPS Seconds – leap seconds) < 0, add in one week to the GPS Seconds
// count and subtract one week from GPS Week count(avoids negative time)
If (gpsSeconds – Leap_Seconds) < 0
   gpsSeconds = gpsSeconds + (7 × 24 × 60 × 60) /* add week */
   gpsWeek = gpsWeek – 1      /* subtract week */
End If
tmpBeginning_of_current_week = (7 × gpsWeek) + 1999-08-22 00:00
tmpDay_of_week = floor((gpsSeconds – Leap_Seconds) / (24 × 60 × 60))
tmpSeconds_from_midnight = (gpsSeconds – Leap_Seconds) % (24×60×60)
utcCurrent_date = tmpBeginning_of_current_week + tmpDay_of_week
utcHours = floor(tmpSeconds_from_midnight / (60×60))
utcMinutes = floor((tmpSeconds_from_midnight % (60×60)) / 60)
utcSeconds = tmpSeconds_from_midnight % 60
```

6.2.7 Formatting Dates and Times in Text: ISO8601

All dates and times within the Handbook use the ISO 8601 [13] standard formatting of:

*CCYY-MM-DD***T***hh*:*mm*:*ss.sZ*

Where *CCYY* is a four digit year; *MM* is the month number; *DD* is the day within the given month; **T** is a placeholder to signify a separation between the Date and Time; *hh* is the hour number ranging from 0 to 23; *mm* is the number of minutes in an hour; *ss.s* is the number of seconds along with fractions of a second which can be more than one digit; *Z* is a single letter that signifies the time zone of the time, for this document all times are in the Zulu, or Z, time zone.

6.3 Timestamp Accuracy and Precision

Quantities that are measured always have some tolerance and disturbances that introduce uncertainty. Measurement uncertainties are composed of both **systematic** and **random** errors. **Systematic** error is caused by abnormalities in one or more system components and tends to shift all measurements in a systematic way, so that during a number of measurements the average value is constantly displaced or varies in a predictable way. For example, a sampling of a system's reference clock may be offset by some fixed amount from the delay of processing. **Random** errors may be caused by noise, lack of sensitivity, and other reasons; these errors vary in an unpredictable way.

Accuracy indicates how close a measured value is to its actual value. In timestamping, accuracy is the average difference between each timestamp (the measured time) and the actual time of the event (reference time). A measured time value relative to the time reference will either be exact or, more likely, not exact. In a time system, each measured value represents a new instance of time. As time increases, each new measured value is compared to the actual reference time and the differences between pair are collected to provide a "picture" of error – i.e. a histogram showing a distribution of the differences between a measured time and its reference.

In Figure 6-7 (left), measured values (red diamonds) of a time reference show deviation from the reference time; some measured values are near identical to the reference, while others are either greater (above the green dashed line), or less (below the green dashed line) than the reference time. Taking the difference between a measured value and the reference over a collection of many time measurements yields a graph similar to the graph on the right in Figure 6-7.

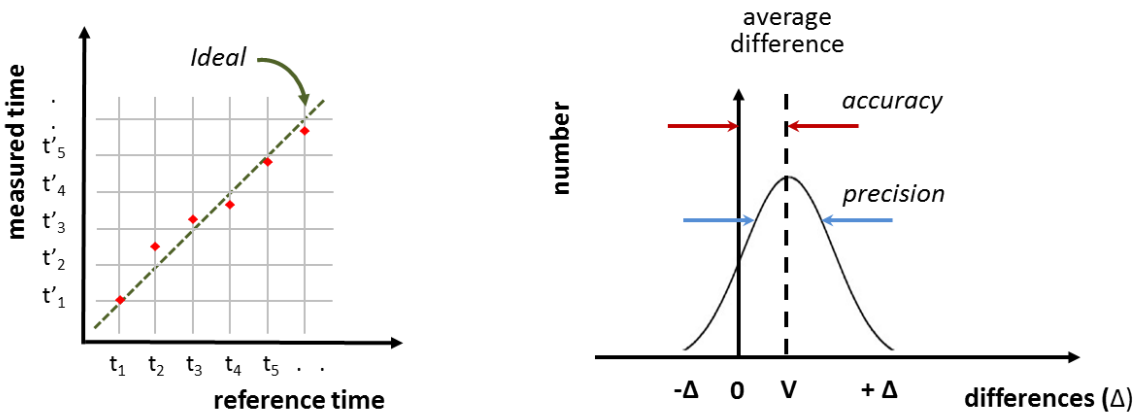


Figure 6-7: A series of time measurements (left). Errors plotted as a histogram (right).

The "average difference" represents the average of all the differences between each measured time value and its corresponding reference time. Ideally, this line would be centered at zero; the offset from zero (shown at "V") represents the accuracy of the data.

The presence of a consistent offset from zero indicates that the accuracy has a *bias* to it. Bias is equivalent to the total systematic error in the measurement, and a correction to negate the systematic error can be made by adjusting for the bias. An example of bias is processing latency, where the capture process of the measurement takes a certain quantity of time to produce the result. System implementers need to understand the source of bias – quantify the bias – and provide correction if necessary. Random error may also move the average difference from zero; this, however, will likely be a much smaller error component if bias is present.

Precision is the ability of a measurement to be consistently reproduced. In timestamping, precision is the variation of the difference between each timestamp and the actual time of the event (reference time). Precision and accuracy are different statistics based on the same set of values. The dispersion of difference values (indicated as ±Δ in Figure 6-7) about the average difference provides a measure of precision. Systems with measured values that tend to be nearer the average in error are said to have "good" precision, i.e. they are repeatable within an acceptable range of confidence. The standard deviation is the metric normally used to gauge this range of confidence. One standard deviation (1σ) about the average will contain 68.3% of the same measured errors.

Figure 6-8 illustrates an example where 500 samples of a time reference are plotted as a function of the error between the time reference and a sampling (measurement) of the time reference.

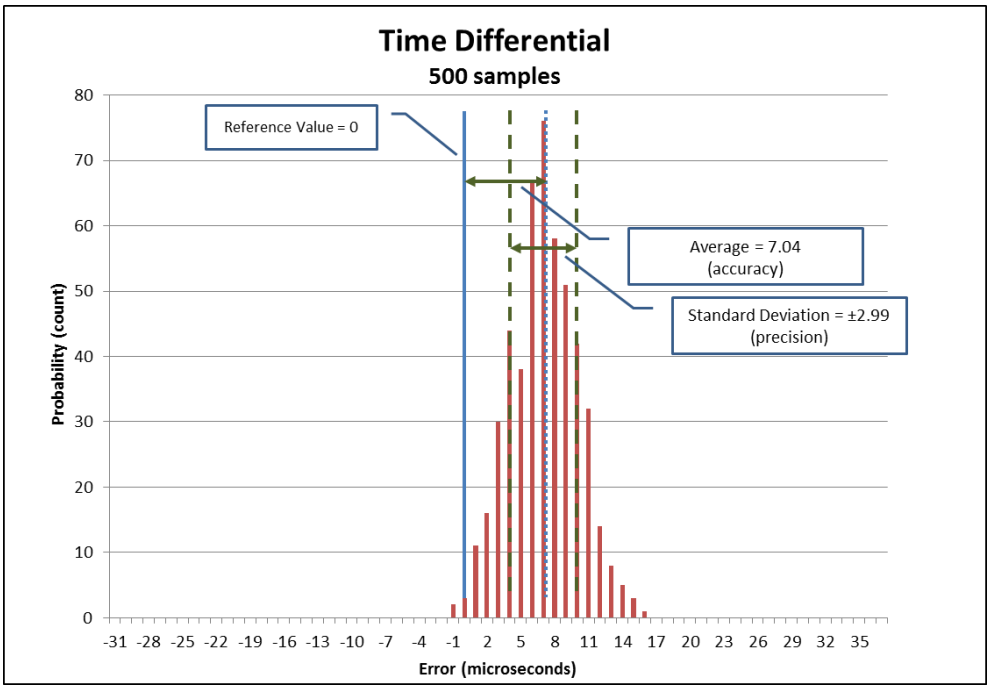


Figure 6-8: Example 1. Poor Accuracy, Good Precision

A plot with no errors would produce all zero values, and thus a single line at the value 0 microseconds. The line called "Reference Value = 0" is the ideal case of no error. Accuracy is a measure of the distance from this reference value to the average value of all sampling; in this case 7.04 microseconds. This histogram of values shows that most errors tend towards a value of 7.04 microseconds the average of all 500 samples produces an average of 7.04 microseconds. Whether an average error in sampling of 7.04 microseconds is acceptable is a systems implementation decision.

In Figure 6-8, the standard deviation is ±2.99 microseconds. Thus, the system in Figure 6-8, example 1, has an average accuracy of 7.04 microsecond's ±2.99 – so the timestamp of the time reference can be found to be within 4.05 to 10.03 microseconds 68.3% of the time.

In Figure 6-9, example 2, the accuracy is better, i.e. 1.46 microseconds, but the precision is worse i.e. ±9.78 microseconds than the case in example 1. The timestamp is more accurate than in example 1, but shows a wider dispersion of error. Thus, the timestamp can be -8.32 to 11.24 microseconds 68.3% of the time with respect to the time reference. This example is considered to have good accuracy, but poor precision as compared to example 1. Again, no judgement is made on whether this is acceptable for a systems implementation.

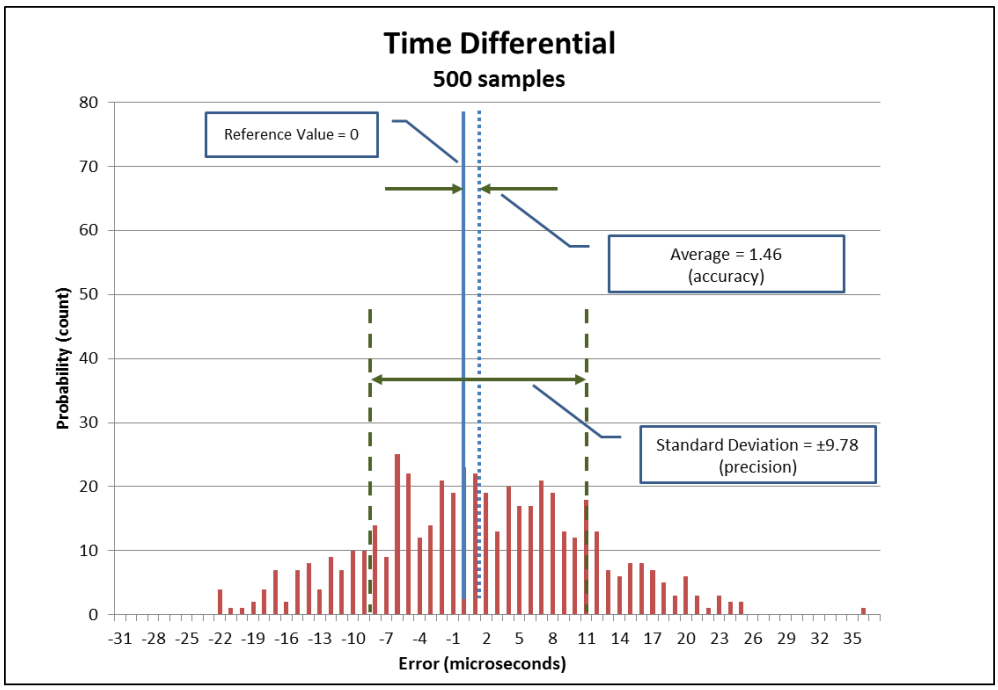


Figure 6-9: Example 2. Good Accuracy, Poor Precision

6.4 Time Transfer

Time transfer is the process of continuously *synchronizing*[7] (see Section 6.2) two or more clocks producing a *hybrid* time source. When performing time transfer one clock is a reference clock and the other clock(s) are receptor clocks. With Motion Imagery Systems time transfer is used to share the reference time of a well-known accurate clock (e.g. GPS) with a local receptor clock (e.g. GPS receiver). Time transfer is a continuous process where each receptor clock is constantly comparing and adjusting its time to match the reference clock.

Figure 6-10 illustrates a two-level time transfer system with two reference/receptor pairs. The first reference/receptor pair, called the primary reference/receptor pair, is a GPS time transfer system between the GPS satellite and GPS receiver. Within this time transfer system, the GPS satellite broadcasts a reference time, which after a brief time later (due to propagation time), is received at a receiver's antenna. The GPS receiver in turn synchronizes the primary receptor time to an internal secondary reference time adjusting for various effects, such as the propagation time to the antenna and relativistic clock rates. This introduces a secondary reference/receptor time system, wherein the GPS receiver produces a time reference (secondary reference) for listening receptor (secondary receptor) devices.

Because the following discussion applies to reference/receptor pairs in general, the qualifiers "primary" and "secondary" are suppressed.

[7] Italicized terms used in this section are the terms defined in Section 6.2

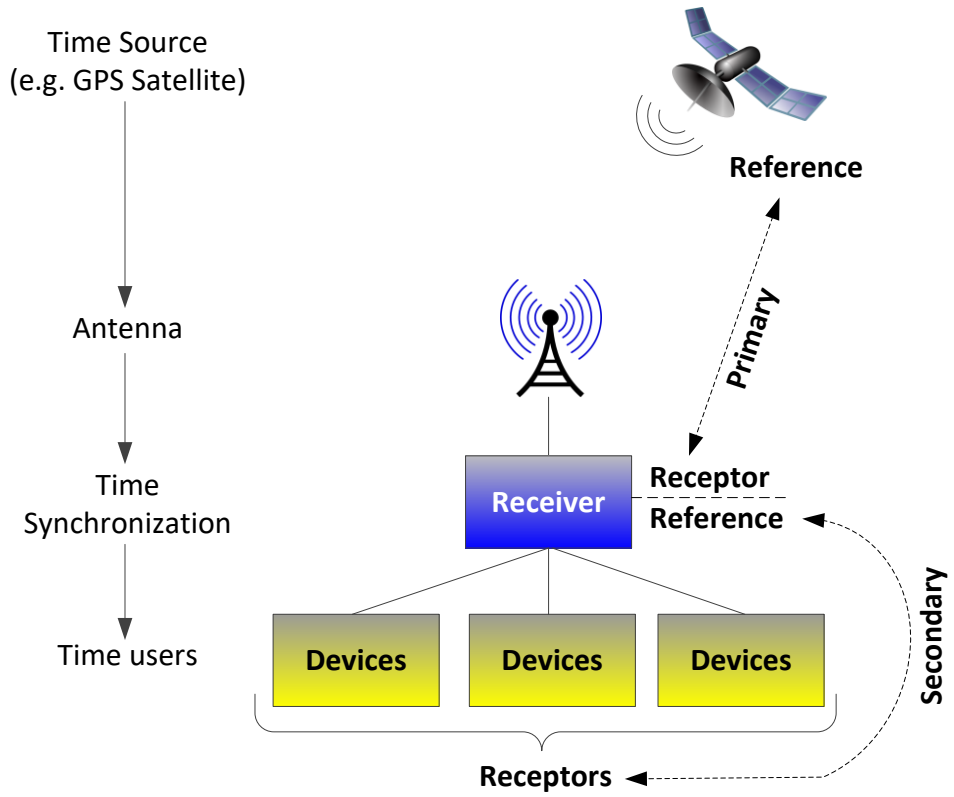


Figure 6-10: Illustration of GPS time transfer to devices.

In typical operation, the reference time is *synchronized* with a receptor, meaning that the two clocks are *locked* with almost zero *offset* between them. The receptor clock may show slight error because of latency and other errors, but these errors can be characterized statistically with a mean and standard deviation.

In cases of signal loss, interference, or other issues, the receptor clock may be temporarily disconnected, or unlocked, from the reference clock for some period of time. Clocks have internal time systems which allow them to continue counting time regardless of external input from a time reference. Such a clock is called *free-wheeling*, as it is independently counting time based on its own means. When a receptor clock becomes unlocked, and is free-wheeling, the clock will drift in time from the reference clock. This difference between the reference time and receptor time at a given instant is called *clock-drift*. Typically, *clock-drift* is linear as a function of time, and can be characterized by a drift-rate.

Figure 6-11 illustrates two clocks, a reference clock in blue and a receptor clock in red, measuring time in milliseconds. The bottom axis is true time (i.e. reference time), while the left axis reports time from the two clocks. The clocks are synchronized at the beginning time (1,000 milliseconds or 1 second). The receptor clock loses lock to the reference time at 9000 milliseconds. At this point, the receptor clock is free-wheeling and drifts away from the reference clock with a constant drift-rate until lock is reacquired at 66,000 milliseconds.

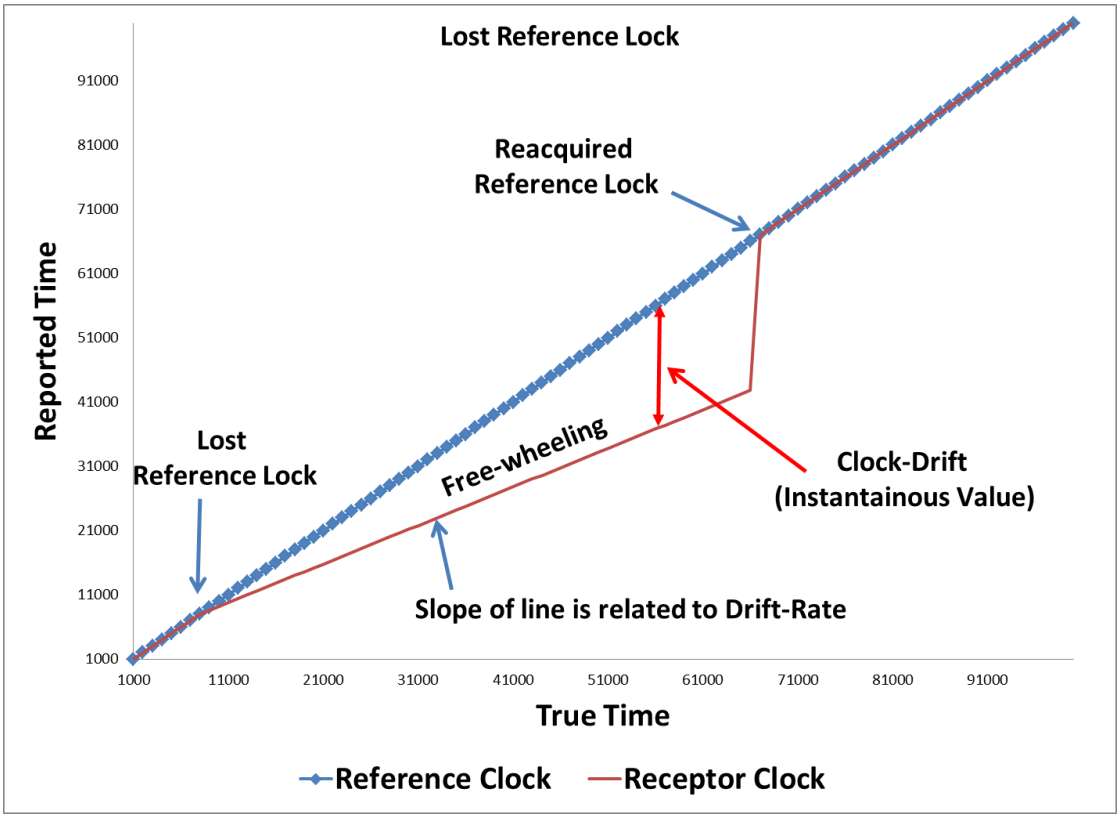


Figure 6-11: Illustration of Clock Synchronization

The receptor clock is in one of two states relative to the reference clock: locked or unlocked. In the locked state, the receptor time is reporting an equivalent value to the reference time, within some statistical error. In the unlocked state, the receptor clock is reporting its own internal time, which drifts away from the reference time with a device-specific drift-rate. The drift-rate is defined with three types of values: equal-to-one, less-than-one or greater-than-one. A drift-rate equal-to-one indicates the receptor clock matches the reference clock when free-wheeling.

A drift-rate less-than-one indicates the receptor clock is slower than the reference time; this is the condition shown in Figure 6-11. Unless the receptor clock is an atomic clock, when the reference lock is lost the receptor clock changes from a Level 3 to a Level 1 Time System (on the Timing System Capability Levels – see Section 6.2.1); however, adjustments can be made to maintain a Level 3 Time System either in real time or in post processing. When reference lock is reacquired, a slower receptor clock is corrected with a single time value update, which results in a rapid change to the time value, as shown in Table 6-5.

Table 6-5: Slow receptor clock reacquiring the reference clock. The large jump forward in the receptors time is highlighted in yellow.

Reference Clock	Reference Difference	Receptor Clock	Receptor Difference	State
62,000		40,400		
63,000	1,000	41,000	600	unlocked
64,000	1,000	41,600	600	unlocked
65,000	1,000	42,200	600	unlocked
66,000	1,000	42,800	600	unlocked
67,000	1,000	67,000	24,200	locked
68,000	1,000	68,000	1,000	locked
69,000	1,000	69,000	1,000	locked

This rapid change is allowable because the receptor's time remains strictly increasing, and therefore, total order is maintained (i.e. Level 1 Time System). After reference lock is reacquired, the receptor clock changes back to a Level 3. Knowing the unlock time, the re-lock time and the final clock-drift, the drift-rate can be computed, and a correction made in post processing to bring the clock back to Level 3. If the drift rate is known a priori, through testing or other means, real time corrections can be made to keep the receptor clock at Level 3.

When the drift-rate is greater-than-one, the receptor clock is counting time faster than the reference time. When the reference is reacquired, the correction for a faster receptor clock is not a simple update. In this case the reported time would jump backwards in time causing a discontinuity, and therefore, the clock would not be a Level 1 clock as shown in Figure 6-12. When this type of correction occurs, it is a Jam Correction.

Table 6-6: Fast receptor clock reacquiring the reference clock and adjusting the reported time incorrectly. The large jump backwards in the receptors time is highlighted in yellow and the discontinuity is highlighted in red.

Reference Clock	Reference Difference	Receptor Clock	Receptor Difference	State
62,000		83,200		
63,000	1,000	84,600	1,400	unlocked
64,000	1,000	86,000	1,400	unlocked
65,000	1,000	87,400	1,400	unlocked
66,000	1,000	88,800	1,400	unlocked
67,000	1,000	67,000	-21,800	locked
68,000	1,000	68,000	1,000	locked
69,000	1,000	69,000	1,000	locked

To maintain a Level 1 clock the best method is to increment the receptor clock one unit (e.g. 1 millisecond in these examples) per time "tick" until the reference clock catches up to the receptor

clock. The process of slowing the receptor clock down is called Slewing so this type of correction is called a Slew Correction. When the receptor clock is equal to or less than the reference clock, it can resynchronize to the reference clock as shown in Figure 6-13.

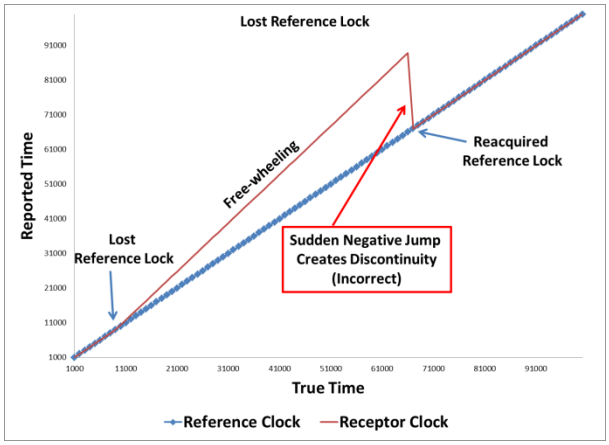


Figure 6-12:Incorrect method to reacquire lock to Reference Clock

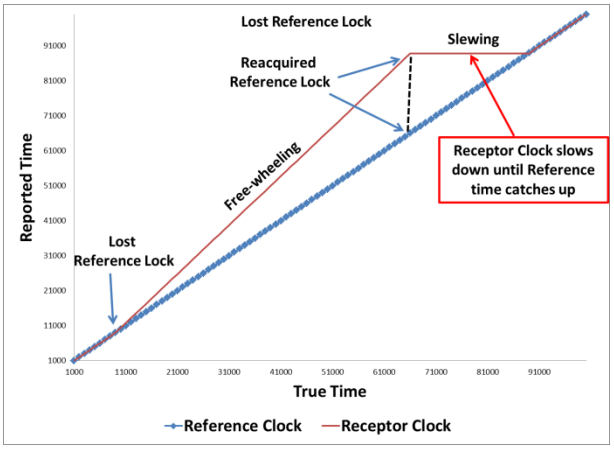


Figure 6-13:Correct method to reacquire lock to Reference Clock

Table 6-7 lists the values for the slew correction from the point of reacquiring the lock to resynchronization.

Table 6-7: Fast receptor clock reacquiring the reference clock and adjusting the reported time for a slew correction. The receptors time is incremented slowly (1 millisecond/tick) while the reference clock catches up. The highlighted yellow shows where the lock is reacquired, and the blue highlight shows where the re-synchronization occurs.

Reference Clock	Reference Difference	Receptor Clock	Receptor Difference	State	Synchronized
62,000		83,200			No
63,000	1,000	84,600	1,400	unlocked	No
64,000	1,000	86,000	1,400	unlocked	No
65,000	1,000	87,400	1,400	unlocked	No
66,000	1,000	88,800	1,400	unlocked	No
67,000	1,000	88,801	1	locked	No
68,000	1,000	88,802	1	locked	No
69,000	1,000	88,803	1	locked	No
…		…	…		
88,000		88,822	1	locked	No
89,000	1,000	88,823	1	locked	No
90,000	1,000	90,000	1,177	locked	Yes
91,000	1,000	91,000	1,000	locked	Yes
92,000	1,000	92,000	1,000	locked	Yes
93,000	1,000	93,000	1,000	locked	Yes

6.4.1 Time Transfer Metadata

To support the ability to correct the free-wheeling time value to the reference time requires metadata about the lock and synchronization status of the clocks. Figure 6-14 depicts a time transfer system that ensures a Level 3 Time System is possible in post processing.

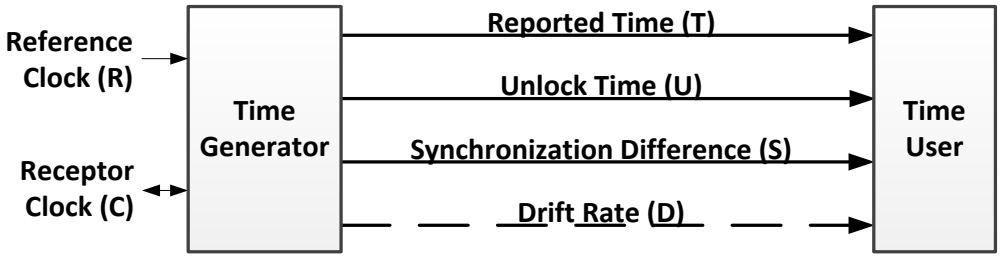


Figure 6-14: Time Generator

The reference clock (R) is assumed to generate a time value at a defined period. Such a clock "pulses" at a continuous, precisely spaced change in voltage; for example, GPS produces a 1 Pulse Per Second (1 PPS) clock period.

R is undefined when unlocked from the reference time. The receptor clock (C), is synchronous with the reference clock when locked (i.e. C=R); however, it will free-wheel otherwise.

The primary result of the time generator is the reported time (T) that is either locked to R or equal to C.

Between reference clock pulses R, the receptor clock C may drift; it is assumed that the reported time T is within required tolerances of the system (otherwise, either a larger number of PPS should be used, or a better time generator system used).

Secondarily, the time generator produces values used to correct the reported time. When correcting the time in post processing, the Unlock Time (U) and Synchronization Difference (S) are used. When the drift rate (D) is known a priori, it can be used to correct the T in real time.

The Unlock Time U, is a duration of time measured in receptor-time time-frame units since the last locked value; if the reference clock R never loses lock, then U will always be zero. The Synchronization Difference S, is the last known time difference between receptor clock C and reference clock R once resynchronization occurs; if C=R (i.e. R locked and synchronized to C), then S=0; otherwise, $S \geq 0$.

Given T, U, S and D clock drift can be removed during post processing or in real-time using Equation 8 below. Figure 6-15 shows an example where a clock has lost lock twice. The lock is lost a second time during slewing.

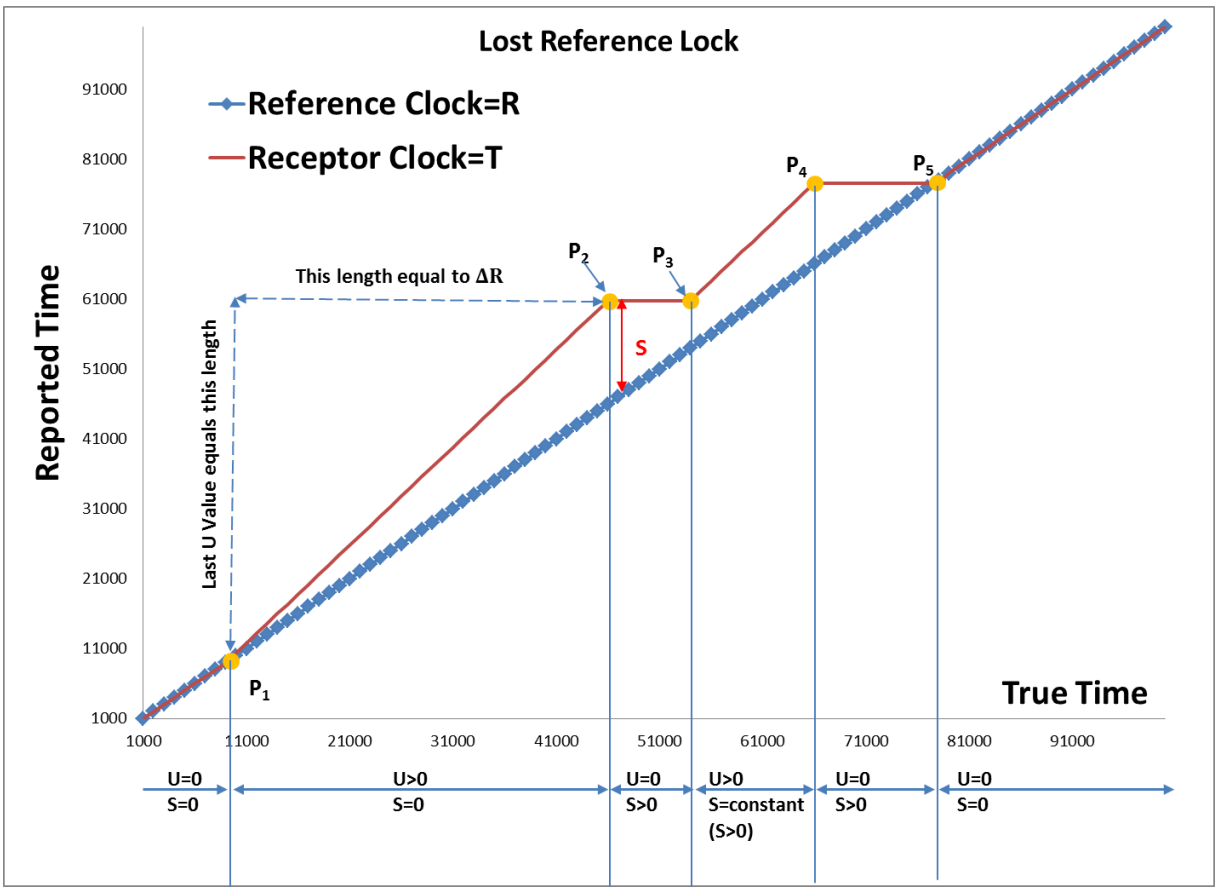


Figure 6-15: Example of losing reference lock twice, with the second loss during a time-convergence.

The figure shows five instances in time, labeled as P_1...P_5, where the lock/unlock status changes. Table 6-8 lists the values for R, C, T, U and S at each of these times along with the preceding time.

Table 6-8: Points of change in Figure 6-15.

	P_{1a} **locked**	P_1 **unlocked**		P_{2a} **unlocked**	P_2 **locked**		P_{3a} **locked**	P_3 **unlocked**		P_{4a} **unlocked**	P_4 **locked**		P_{5a} **locked**	P_5 **locked**
		free-wheeling			slewing			free-wheeling			slewing			
R	R_{1a}	N/A	...	N/A	R_2	...	R_{3a}	N/A	...	N/A	R_4	...	R_{5a}	R_5
C	$C_{1a}=R_{1a}$	C_1	...	C_{2a}	$C_{2a}+1$	...	$C'+1$	C_3	...	C_{4a}	$C_{4a}+1$	...	$C'+1$	$C_5=R_5$
T	C_{1a}	C_1	...	C_{2a}	$C_{2a}+1$	...	$C'+1$	C_3	...	C_{4a}	$C_{4a}+1$	...	$C'+1$	C_5
U	0	C_1-C_{1a}	...	$C_{2a}-C_{1a}$	0	...	0	C_3-C_{3a}	...	C_4-C_{3a}	0	...	0	0
S	0	0	...	0	$C_{2a}+1-R_2$	...	$C'+1-R_{3a}$	$C'+1-R_{3a}$	...	$C'+1-R_{3a}$	$C_{4a}+1-R_4$	...	$C'+1-R_{5a}$	0

The subscript "a" denotes the reported time immediately before the change occurs. The area marked as slewing shows where the receptor clock is slowly incrementing (by 1 unit i.e. micro or nanoseconds) until the reference clock catches up. The value C' is the previous clocks value in

the figure during slewing. The shaded region on the "R" row shows where the reference clock is not available. The reference and receptor clocks values are separated because they are not sent to the time user; only the T, U and S values are sent.

The following equations are used for time corrections that can be computed in post processing by monitoring two successive T, U, S sets preceding a re-lock of the clock. The values U and S are relatively small values compared to the T value, so the amount of data to support these values is minimal.

To compute a correction to C when it is free-wheeling between P_1 and P_2, the slope of the line from P_1 to P_2 is computed in Equation 3:

$$slope = m = \frac{C_2 - C_{1a}}{R_2 - R_{1a}} \approx \frac{C_{2a} + \Delta_{2a} - C_{1a}}{R_2 - R_{1a}} = \frac{U_{2a} + \Delta_{2a}}{(T_2 - S_2) - (T_{2a} - T_{2a})} \quad \text{Equation 3}$$

Where:

Δ_{2a} The difference between T_{2a} and the preceding time value, T_{2a-1}. This value can be computed on the fly by the time user.

The estimated corrected value for the i^{th} time, R_i, within the free-wheeling interval between P_1 and P_2 is derived in Equation 4 and Equation 5:

$$C_i - C_{1a} \approx m * (R_i - R_{1a}) \quad \text{Equation 4}$$

$$R_i \approx \frac{C_i - C_{1a}}{m} + R_{1a} = \frac{T_i - (T_{2a} - U_{2a})}{m} + (T_{2a} - U_{2a}) \quad \text{Equation 5}$$

When C is free-wheeling between P_3 and P_4, the correction has an additional offset from the slewing difference. Equation 7 compensates for the vertical offset using the S_{4a} value.

$$slope = m \approx \frac{C_{4a} + \Delta_{4a} - C_{3a}}{R_4 - R_{3a}} = \frac{U_{4a} + \Delta_{4a}}{T_4 - (T_{4a} - U_{4a})} \quad \text{Equation 6}$$

$$C_i - C_{3a} \approx m * (R_i - R_{3a}) + S_{4a} \quad \text{Equation 7}$$

$$R_i \approx \frac{C_i - C_{3a} - S_{4a}}{m} + R_{3a} = \frac{T_i - (T_{4a} - U_{4a}) - S_{4a}}{m} + (T_{4a} - U_{4a}) \quad \text{Equation 8}$$

Equation 6 and Equation 8 are the general case equations used for all cases. In the first example when free-wheeling between P_1 and P_2, the value of S_{2a} equals zero, so Equation 7 resolves into Equation 4.

The time generator can optionally include the Drift Rate, D which is the drift-rate of a receptor clock if it's known a priori. When the drift-rate is known, it is used for the slope (i.e. m) and Equation 8 can be applied in real time.

Chapter 7

Metadata

Metadata is data about data. The Motion Imagery Standards Profile (MISP) uses metadata to characterize both visual and non-visual information about collected Motion Imagery. These characteristics include the environment of Motion Imagery sensors, the kinematics of the sensors, timing, object information, classification and much more.

There are many ways to organize and format metadata, each with advantages and disadvantages. One efficient metadata encoding method used across many MISB standards is the Key-Length-Value (KLV) data encoding standard defined in SMPTE ST 336 [14]. In many government systems, the available delivery bandwidth from a sensor to a user is constrained. This warrants a tradeoff in the quality of the Motion Imagery versus bandwidth allocated for metadata. The MISB extends SMPTE ST 336 with additional rules and structure to further reduce the quantity of bytes needed to represent various metadata elements; this translates directly into reduced bandwidth for data exchange. To this end, the complexity in extending KLV encoding over that prescribed by SMPTE justifies this intent.

7.1 Metadata Theory

Sensors (e.g. cameras) create raw data or *essence* [15]. In a Motion Imagery System, one essence is the visual Motion Imagery; another collaborative essence is audio. Such essence types are far more meaningful with context, which comes in many forms, such as what, where, why or how an essence is created. The context may be apparent based on the familiarity of the essence; often it is not. To provide users the ability to definitively understand an essence, contextual information is supplied; this contextual information is metadata.

Contextual information about essence can **expand** and **change** as more effort is applied to understanding the data. Expansion can result from adding context from data sources associated with the collection of the essence (e.g. gimbal angles, timing, and GPS position); adding context by performing analysis; or reusing a contextual dataset from other sources. Changes can be attributed to corrections or improvements of measurements, removal of un-needed data, or other functions.

The process of expanding and changing the contextual information forms a life-cycle. This life-cycle is important to document as it provides traceability to events which occurred to the information; such traceability may be critical in forensic analysis of the data. Therefore, the history of changes needs to be recorded; this history information is called Provenance.

In Figure 7-1, Item 1 shows an Essence with its Collection Metadata. Item 2 illustrates potential life-cycle changes, where processing operations on the Collection Metadata, Essence or both produces Processed Metadata. Further data exploitation (either human or machine) may produce additional Analysis Metadata.

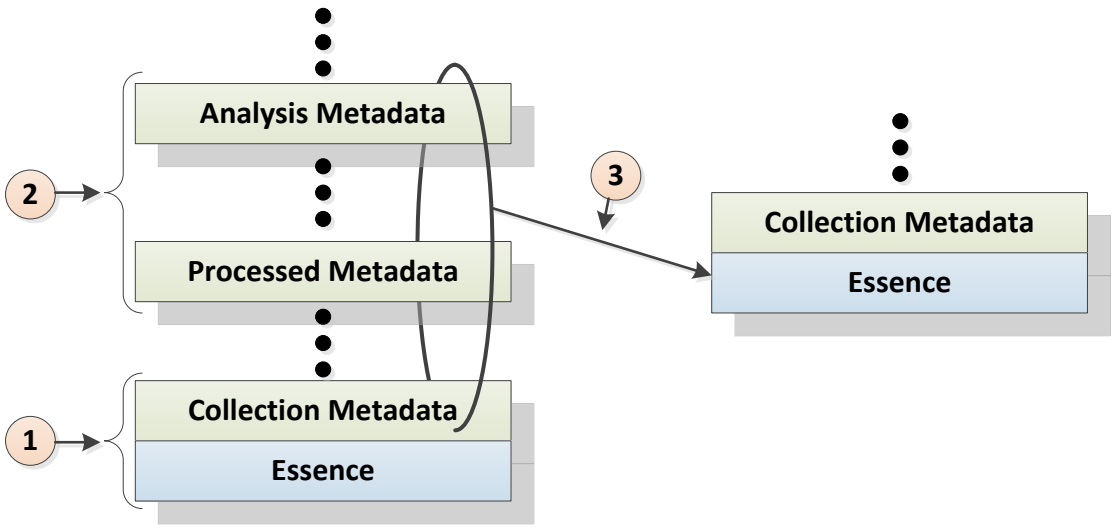


Figure 7-1: Metadata Creation, Life-Cycle and Spawning

As contextual information evolves in its life-cycle, it can itself become an essence. The goal of any analysis process is to reduce data to a set of suppositions, or ideally, facts. Suppositions rely on supporting information, or a link back to originating data used to create the suppositions; this link is Provenance Metadata as traced back to the data's source.

For example, suppose the goal is to track vehicle movement within a scene. Motion Imagery is collected along with associated metadata (such as the Motion Imagery's sensor position, movement, etc.). Tracking software detects and tracks the vehicles. While this provides metadata about the Motion Imagery content, it is also an essence itself. Extracting and evaluating the track information (suppositions) independent of the Motion Imagery promotes the track information into essence; in this case, the track information links to or references the source i.e. the captured Motion Imagery. Item 3 in Figure 7-1 illustrates spawning a new essence from existing metadata information, where a subset of the originating metadata forms a new essence.

7.1.1 Metadata Information

As discussed, Metadata is data about essence. Metadata are individual data elements of information providing either context for the essence, or context to the metadata itself. A data element has three parts: **value**, unique **identifier** and **definition**. The value is the data associated with the essence. The identifier is the name of the data element and links the value to its definition.

A *controlling document* – separate from the essence – defines the meaning of the value in a definition. For example, a data element *identifier* "Sensor Height Above Ellipsoid (HAE)" has a *value* of "100", and its *definition* is "a value of meters from the WGS-84 ellipsoid to height of the sensor above the ground." The identifier and its definition for this data element exists in a controlling document, while the identifier and its value exist as metadata for its essence (i.e. Motion Imagery). The controlling document provides the *specification* for the metadata (i.e. meaning, form and rules), while the metadata itself is an *instance* of that specification. The metadata, as an instance of the specification, is binary data.

The example in Figure 7-2 further clarifies the relationship between a controlling document and an instance of a metadata element. On the left side of Figure 7-2, a Sensor produces Essence per the rules of a chosen standard (i.e. Essence Standard), such as H.264/AVC. Supporting Devices to the Sensor provide measurements, which become the sensor's metadata. The meaning, format and rules for constructing the metadata follow the specification as defined in a Controlling Document (i.e. Metadata Specification). This creates a Metadata Instance linked to the Essence. The process for creating the metadata "*invokes"* the Controlling Document when making the Metadata Instance. The Essence and Metadata Instance are then combined, where now Users, who reference the same Essence Standard and Controlling Document, can understand how to process and view the Sensor essence and understand the meaning of the Supporting Devices metadata.

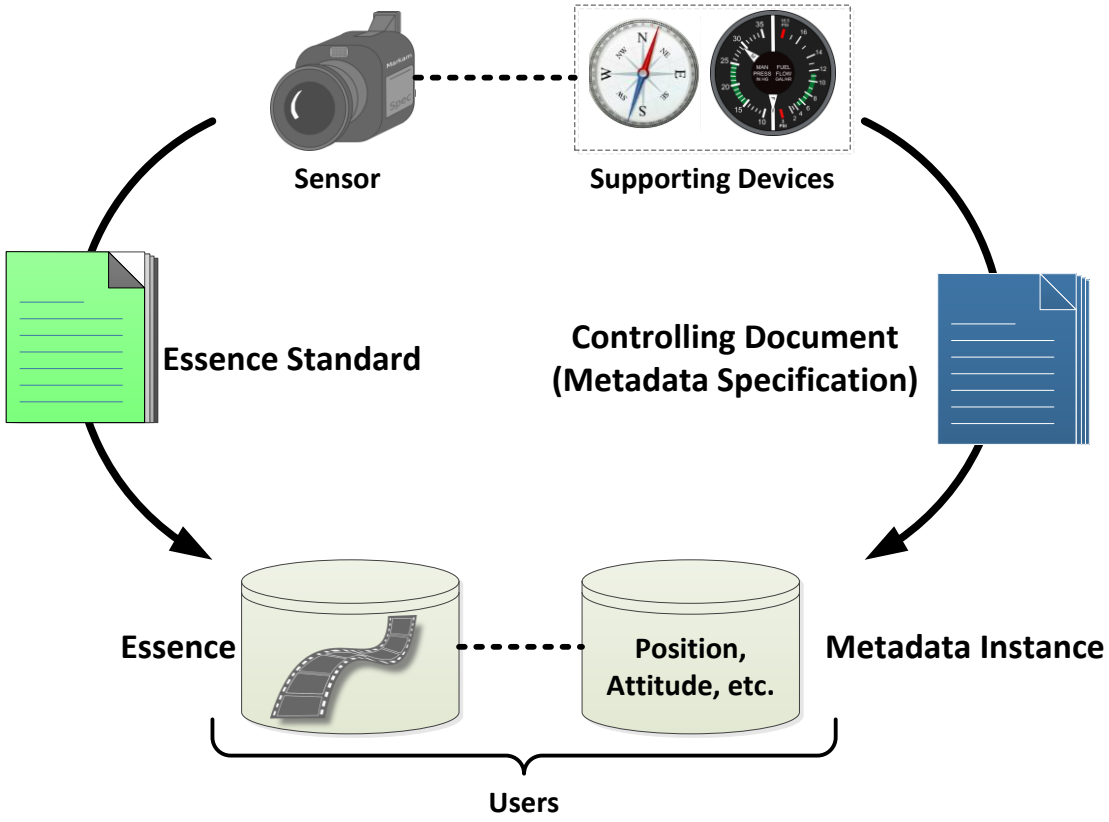


Figure 7-2: Relationship between a Metadata Specification and its Instance

A controlling document is typically cast into a standard, which provides both a metric for conformance and documented revision control. Metadata standards can use other metadata standards, whereby one standard invokes another standard.

There are multiple ways to group and organize metadata elements. Three common methods for organizing metadata are **Lists**, **Trees** and **Graphs**. The MISP standards directly employ both Lists and Trees; Graphs are indirectly used.

7.1.1.1 Lists

Lists are the most basic method of organizing a group of data elements. A list is a named group of elements, which is either ordered or unordered. An ordered list requires its data elements arranged in a specific sequence. The first column of Table 7-1 shows an example of an ordered list; the numbers indicate a required ordering. The second column shows an unordered list of the same information, but with random ordering (thus, with no numbers).

Table 7-1: Example of Ordered List and Unordered List

Ordered List	Unordered List
1. Temperature	Sensor Longitude
2. Sensor Latitude	Temperature
3. Sensor Longitude	Sensor Latitude
4. Sensor Height Above Ellipsoid (HAE)	Sensor Type
5. Sensor Type	Sensor Height Above Ellipsoid

An ordered list reduces the amount of information needed to convey metadata elements. A controlling document for the list defines the elements and the order of each element in the list. Individual data element identifiers in the list do not need to be represented, thereby saving bytes. In contrast, in an unordered list, where the order is not defined, or known, each data element identifier must be included to understand what each data element means. The advantage of an unordered list is flexibility. Thus, there is a tradeoff in data quantity versus flexibility between the two types of lists.

Figure 7-3 pictorially illustrates the difference between the two types of lists. The ordered list maintains its identifiers for its data elements in a controlling document along with a prescribed order to the data elements, while the unordered list does not. Because the identifiers are not present in the metadata, the ordered list consumes less bytes. Both types of lists do require a description (or definition) of each data element to understand what its values represent. The definitions are documented in a controlling document, which includes qualifiers such as units (SI measures), type and format – all necessary to properly interpret a data element's value.

The information in the ordered list of Figure 7-3 is five data values, while the unordered list has five identifiers *plus* five data values. An ordered list conserves bytes – but only when the entire list is used. For example, when Temperature and Sensor Type are the only two data elements specified, the ordered list will still have five data values (the other values would contain "filler"). The unordered list, on the other hand, would have two identifiers and two data quantities, and thus require fewer overall bytes.

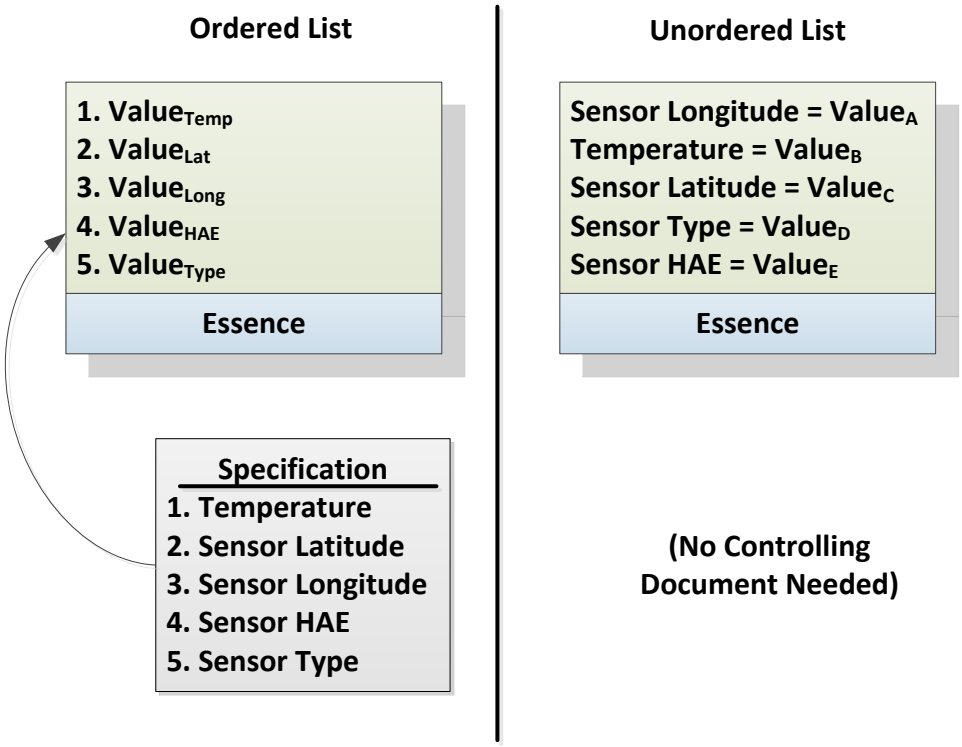


Figure 7-3: Ordered List and Unordered List Examples

7.1.1.2 Trees

Lists work well in many situations, but when the same data element is needed for different purposes within the same list, another structure is better suited. For example, consider two sensors, with each sensor having a position (Sensor Latitude, Sensor Longitude and Sensor HAE) and Sensor Type. When referencing "Sensor Latitude" in a list, it is unclear to which sensor it refers. Lists could be forced to support these situations by either writing a controlling document to support duplicates or creating names such as "Sensor Longitude for Sensor 1"; however, this is cumbersome and inflexible for a system with many sensors.

A **Tree** structure enables parent/child relationships, where the top-level parent is the "*root*" and a child is a "*branch*" of the tree (see Section 7.5.1 for definitions). In hierarchical fashion, a child can likewise have one or more children, thus making it a parent as well. A Tree structure offers a means to collect a group of information into its own list, and then include that list as a single item in a parent list. For instance, Table 7-2 shows an example of two different metadata trees for the same two sensors.

On the left, at the top of the tree (i.e. parent or root) are the data elements "Temperature", "Sensor 1" and "Sensor 2." This parent list of data elements contains two sub-lists (i.e. children or branches): the data elements "Sensor Latitude" through "Sensor Type" belonging as children to the parent "Sensor 1", and similarly, the data elements "Sensor Latitude" through "Sensor Type" belonging to the parent "Sensor 2". Note that each parent contains a sub-list of the same data element identifiers. Although the names are the same, the values for the two sets of identifiers may be different. Using levels to denote relationships, the parent is at *level 1* of the

tree; children beneath the parent are at *level 2*, and so forth. The data elements in the left column thus represents a two-level tree.

The right column of Table 7-2 shows an example of three-level tree. The root contains the data elements numbered 1, 2, 3; the children at level 2 contain the elements labeled “a” and “b”; the children at level 3 contain the elements labeled “i", “ii”, and “iii”. Parent or children lists can be either ordered or unordered lists.

Table 7-2: Example of a Tree Structure

Two Sensor Metadata	
Two-Level Tree	Three-Level Tree

1. Temperature
2. Sensor 1
  a. Sensor Latitude
  b. Sensor Longitude
  c. Sensor HAE
  d. Sensor Type
3. Sensor 2
  a. Sensor Latitude
  b. Sensor Longitude
  c. Sensor HAE
  d. Sensor Type
	1. Temperature
	2. Sensor 1
	  a. Position
	  i. Sensor Latitude
	  ii. Sensor Longitude
	  iii. Sensor HAE
	  b. Sensor Type
	3. Sensor 2
	  a. Position
	  i. Sensor Latitude
	  ii. Sensor Longitude
	  iii. Sensor HAE
	  b. Sensor Type

7.1.1.3 Graphs

Graphs enable relationships beyond a top down or bottom up relationship. Graphs are networks of elements, with any element linked to any other element. The MISP currently does not directly use Graphs, but may do so in the future.

7.1.2 Metadata Operations

Contextual information about essence is not static; it changes during analysis. The metadata life-cycle is the application of a series of operations that occur to the metadata. In this section, a simple set of operations define how metadata expands and/or changes through the subsequent steps of analysis. The metadata operations are: **Create**, **Add**, **Delete** and **Edit**.

The **Create** operation produces the initial (first generation) group of data elements. The **Add** operation augments or appends new data elements to an existing group. The **Delete** operation removes data elements from a data element group. The **Update** operation changes or substitutes a new value for an existing data element. The Update operation appears to be a combination of the delete and add operations. Using these two operations on an ordered list, however, would change the order of the list, so they are not equivalent to the function provided by the update operation. Figure 7-4 illustrates these operations on a set of data elements, which could be organized using the List, Tree or Graph method.

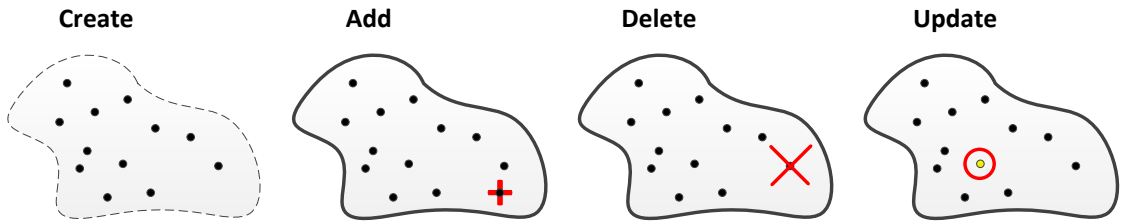


Figure 7-4: Metadata Operations

7.1.2.1 Data Preservation

In applying the metadata operations, it is critical to not destroy existing metadata. When metadata is created, it becomes the first generation of metadata. Add, delete and update operations create subsequent generations (second, third, etc.) of metadata. Maintaining all generations of metadata satisfies a legal requirement to "keep all data" (i.e. data preservation), and insures traceability of the essence and metadata back to its first-generation values, if necessary.

Data preservation is compromised when applying the delete and update operations. The purpose of the Delete operation is to remove data from a data element group. This violates data preservation, so a system of "marking" a data element as deleted, without removing it, is needed. The purpose of the Update operation is to change a data elements value. Likewise, this violates data preservation, so the original data element value must be retained while adding the replacement value. Operations that honor data preservation are called *Enhanced* operations. Since an Enhanced Update operation could be used multiple times, the "marking" methodology must also support updates-of-updates of metadata.

Three methods to "mark" data elements as either deleted or updated include: 1) modifying the data element identifier; 2) adding state information for each data element; 3) or, introducing a list structure called "Amend Metadata".

Modifying the data element identifier presents issues when working at the detailed level of metadata encodings, i.e. changing a Universal Key to denote a modified data element. This quickly becomes a data management issue, requiring documenting new keys and updating processing equipment. Therefore, this method is not considered. In general, modifying an identifier is not a good solution, especially when updates on top of updates are possible.

Supplying state information for each data element requires a prefix (or suffix) to be added, which must be processed when reading the data value. In an Update or Delete operation, the state information must be updated to reflect the change. Such an example is assigning a version number for each data element. When a data element is updated, the version number for the new data element is the next increment of the current data element's version number. This requires version numbers for every changed data element. When groups of changes are made (i.e. changing several data elements at once), the state information must be duplicated for each data element; this is inefficient and wastes bytes.

Adding an "Amend Metadata" list structure to a metadata group is a flexible solution which supports multiple tree levels and adds extra data only when changes occur. Table 7-3 provides an example of the Enhanced Delete operation, where the "Amend Metadata" list shows the deleted data element (i.e. 'Delete Sensor Type'). The first column contains the first-generation values;

the second column shows a Delete operation with the 'Sensor Type' missing from the list; and, the third column shows the Enhanced Delete operation, where the Amend Metadata has been added (shown in red) denoting the deleted element. Thus, there is data preservation.

Table 7-3: Example showing the difference of deleting the "Sensor Type" item using the Delete and Enhanced Delete operations for an unordered list.

First Generation Values	Delete	Enhanced Delete
Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible Amend Metadata = 'Delete Sensor Type'

Table 7-4 shows an example of how the Enhanced Update operation uses Amend Metadata to indicate an element (i.e. 'HAE') has been updated. The first column is the first-generation values before the update; the second column shows an Update operation with the 'HAE' value changing (in red); and, the third column shows the Enhanced Update operation, where the Amend Metadata (in red) describes the update. With the Enhanced Update operation, the original value of 100 is not changed, but the Amend Metadata indicates the 'HAE' value has been updated to 200.

Table 7-4: Example showing the difference of updating the "HAE" item using the Update and Enhanced Update operations for an unordered list.

First Generation Values	Update	Enhanced Update
Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 200 Sensor Type = Visible	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible Amend Metadata ='Update HAE = 200'

Using trees, discussed in Section 7.1.1.2, the Amend Metadata items can be stored in a child sub-list. Different changes can be grouped together into their own Amend Metadata list and each Amend Metadata list can be used independently. Table 7-5 is an example of two Amend Metadata sub-lists in red. The first Amend Metadata sub-list shows a deletion of 'Sensor Type' and an update of 'HAE' from 100 to 200. The second Amend Metadata sub-list shows an update of 'HAE' from 100 to 150. In both 'HAE' cases, if the child element exists in the parent then the change is an Update operation; otherwise, it is an Add operation.

Table 7-5: Example showing an Enhanced Delete-and-Update Amend Metadata as sub-lists of the first-generation metadata.

First Generation Values	Enhanced Delete and Update
Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible Amend Metadata Sensor Type=<null> HAE = 200 Amend Metadata HAE = 150

Per Data Preservation, there is an additional step to consider in the *interpretation* of the resulting metadata during processing. The interpretation must include choices based on the number of Amend Metadata lists included. For example, Table 7-6 shows several options in interpreting the metadata listed in the right column of Table 7-5. Note that all data elements listed at the right in Table 7-5 exist in the group of metadata; that is, no first-generation or subsequent-generation metadata are missing. However, from an end-user perspective, there is a choice in which data elements to use consistent with the three options.

Table 7-6: Possible interpretations of metadata by end user.

Option 1	Option 2	Option 3
Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 200	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 150 Sensor Type = Visible

This interpretation, of children sub-trees, is an "ORing" operation, meaning that Option 1 OR Option 2 OR Option 3 is displayed to, or used by the end user.

Amend Metadata can be used in a hierarchy, supporting multiple levels of sub-lists to represent updates-of-updates. Table 7-7 shows two levels of Amend Metadata.

Table 7-7: Example showing enhanced delete and update Amend Metadata as sub-lists of first-generation metadata.

First Generation Values	Enhanced Delete and Update
Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible Amend Metadata Sensor Type=<null> HAE = 200 Amend Metadata HAE = 150

The interpretation of Table 7-7 has several options as shown in Table 7-8:

Table 7-8: Possible interpretations of metadata by end user.

Option 1	Option 2	Option 3
Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 200	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 150

Amend Metadata sub-lists can include new data to be introduced as a change occurs. For example, a 'HAE' value is updated and the standard deviation 'HAE_SD' is computed with the new value. The standard deviation value is added as new metadata with the Amend Metadata as illustrated in Table 7-9.

Table 7-9: Example showing the Amend Metadata with an Enhanced Update operation which includes new metadata.

First Generation Values	Enhanced Delete and Update
Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Type = Visible Amend Metadata HAE = 150 HAE_SD = 2

7.1.2.2 Data Sharing

In certain applications, metadata trees contain similar information. To save bytes the trees can be combined into one structure; however, an interpretation process is needed to separate the trees back into their original contexts. For example, assume a Motion Imagery stream contains a split

screen of two sources from the same platform. Some of the metadata accompanying these two sources contains the same information as shown in Figure 7-5. The items highlighted in red indicate the items that are the same in each list; these items can be shared.

Temperature = 20	Temperature = 20	Common
Latitude = 44.98	Latitude = 44.98	
Longitude = -93.26	Longitude = -93.26	
HAE = 100	HAE = 100	
Sensor Azimuth Angle=10	Sensor Azimuth Angle=170	Unique
Sensor Roll Angle=0	Sensor Roll Angle=5	
Sensor Elevation Angle=40	Sensor Elevation Angle=30	
MIView=Left	MIView=Right	

Figure 7-5: Example of two Motion Imagery sources with common and unique data.
(Bruges Belgium 2014 Lat: N51 12'18" Lon: E3 13'48")

Using a tree structure, the Common elements are grouped at the parent level, while the Unique elements are grouped at a child level. A "Segment Metadata" list structure enables this tree structure, where Segment Metadata sub-lists contain child elements to a parent group of metadata. Table 7-10 shows the combination of the two metadata lists with the common data elements (in red) at the root of the tree, and the unique sensor-specific data elements in two children trees each identified with their own Segment Metadata list.

The reduction in bytes saved by not replicating common elements can be meaningful when using large datasets. Beyond the savings in bytes, however, is a method that affords expressing multiple metadata items using the same identifier with different values repeatedly without confusion.

With Data Sharing the metadata requires an additional step in its interpretation. The interpretation process must process all "branch" metadata back into their original lists, providing all lists to the end user. For example, a reader of the metadata in Table 7-10 needs to produce the two lists in Figure 7-5.

Table 7-10: Common Metadata Combined

Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Segment Metadata Sensor Azimuth Angle = 10 Sensor Roll Angle = 0 Sensor Elevation Angle = 40 MIView = Left Segment Metadata Sensor Azimuth Angle = 170 Sensor Roll Angle = 5 Sensor Elevation Angle = 30 MIView = Right

This type of interpretation, of children sub-trees, is an "ANDing" operation. So, given several Data Sharing options (e.g. in Figure 7-5 there are two lists of items so there are two options), Table 7-11 shows Option 1 AND Option 2 is displayed to, or used by the end user. Nesting of branches is an extension to the single element tree which is allowed, so a branch in a branch is allowed.

Table 7-11: Options in interpreting the metadata of Figure 7-5.

Option 1	Option 2
Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Sensor Azimuth Angle = 10 Sensor Roll Angle = 0 Sensor Elevation Angle = 40 MIView = Left	Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 200 Sensor Azimuth Angle = 170 Sensor Roll Angle = 5 Sensor Elevation Angle = 30 MIView = Right

7.1.2.3 Co-Use of Amend Metadata and Segment Metadata

Utilizing both Data Preservation and Data Sharing within a metadata tree is possible. The Amend Metadata and Segment Metadata are both uniquely identifiable and can be intermixed at the same level of a tree or at different levels. For example, the side-by-side example of Table 7-10 could include updates for sensor angles in one of the Segment Metadata lists as shown as red text in Table 7-12.

Section 7.5.8 provides details and references for implementing the Amend Metadata and Segment Metadata structures.

Table 7-12: Amend Metadata list in a Segment Metadata list

Temperature = 20 Latitude = 44.98 Longitude = -93.26 HAE = 100 Segment Metadata Sensor Azimuth Angle = 10 Sensor Roll Angle = 0 Sensor Elevation Angle = 40 Sensor FOV = 20 MIView = Left Amend Metadata Sensor Azimuth Angle = 15 Sensor Roll Angle = 1 Segment Metadata Sensor Azimuth Angle = 170 Sensor Roll Angle = 5 Sensor Elevation Angle = 30 MIView = Right

7.2 Metadata Modelling

Metadata Modelling is the process of describing the metadata used within a system or group of systems. Typically, metadata models describe metadata groups (lists, trees, graphs) with a modelling language, and since the MISB uses KLV as a metadata transmission method, the descriptive language is the KLV groups defined in the MISP. The current models are starting to become more complex, so the use of KLV as the modelling language is becoming cumbersome and inflexible. As an example, the use of MISB ST 1010 – Standard Deviation and Correlation Coefficient Metadata (SDCC), adds a level of complexity to the metadata structure. The use of the SDCC metadata links disparate metadata elements together in a graph structure, which KLV metadata does not support directly.

In future versions of the Motion Imagery Handbook, the purpose of this section will be to describe or reference a modelling language (e.g. UML) that will be independent of the data format. Users of the models will not need to be MISB KLV experts to understand the metadata structure.

The goal of the MISB is to produce metadata models common across the many varieties of systems both deployed, and in development.

7.3 Metadata Types

Metadata elements are values stored digitally with essence data. The digital values are represented in units of octets or bytes with each octet containing 8 bits. The term “byte” is used in all MISP documents to represent 8-bits of data. Bytes are the building block of many other data formats, such as numeric and character formats. Bytes are used in one of two ways: as a

numeric representation, or in table lookup. As a numeric representation, each bit, within one or more designated bytes, relates to each other, thereby providing encoded information used to compute a numeric value. Examples of numerical representations include Integer and Floating-Point formats. With table lookup, the bit values within a byte(s) represent a specific index to a pre-defined list of values. An example of a table lookup representation is ASCII text values.

In the numeric form each byte can be represented as a binary value ranging from zero to 255. Each bit, b_i, represents the inclusion of a power of two in a sum to create a value, V, as shown in Equation 9.

$$V = \sum_{i=0}^{7} b_i * 2^i \qquad \text{Equation 9}$$

In Equation 9 each bit within the byte has different significance; the first bit signifies a unit value (2^0); the second bit signifies a two value (2^1); the third bit signifies a four value (2^2), etc. up to the eighth bit (2^7). The term *most significant bit* or *msb* indicates the eighth bit in the byte; the term *least significant bit* or *lsb* indicates the first bit in the byte. Bytes can be transmitted with either the msb or lsb sent first. Likewise, bytes can be stored in memory or represented in diagrams with the msb on the left, right, top, or bottom (all byte level diagrams should include indicators of msb and lsb). When multiple bytes are combined into one value, for example as a multi-byte unsigned integer, the byte containing the most significant bit of the whole value is the *Most Significant Byte* or *MSB*; the byte containing the least significant bit is the *Least Significant Byte* or *LSB*. When data is transmitted with the MSB first, the transmission is termed big endian; when data is transmitted with LSB first, the transmission is little endian.

In ST 0107 [16] the MISB requires all metadata transmitted and stored to be big endian.

A data type is the pattern used to transform digital data representations into meaning. There are many data types; international standards define some types, while others are specific to Motion Imagery standards. There are two categories of data types: simple and structured. Simple data types are representations for single values. Structured data types are data representations for groups of values (lists, and trees) as was discussed in Section 7.1.1.

Table 7-13 lists the simple data types used in the MISB standards. The Data Type column states the full name of the data type. The Abbreviation is the term used in MISP standards for specifying the type. The Section is the section number for more detailed information. The Description provides a brief description of the data type.

Table 7-13: List of Simple Data Types used by the MISB

Data Type	Abbreviation	Section	Description
BER Length	ber	7.3.1	An integer value represented using ASN.1 Basic Encoding Rules for Length
BER-OID	ber-oid	7.3.2	An integer value represented using ASN.1 Basic Encoding Rules - Object Identifier. Depending on the encoding of this value, the number of bytes is adjusted within the stream itself
Binary	binary	7.3.3	Special binary data which does not match one of the other types listed
Character ISO7	iso7	7.3.4	ISO/IEC 646:1991, Information Technology – ISO 7-bit coded Character Set for Information Interchange
Character UTF8	utf8	7.3.5	A character value represented using the 8 bit Unicode value from ISO/IEC 10646:2014
Character UTF16	utf16	7.3.6	A character value represented using the 16 bit Unicode value from ISO/IEC 10646:2014
Enumeration	enum	7.3.7	An unsigned integer used as a lookup in a pre-defined enumeration table
Floating Point	float[1]	7.3.8	A floating-point value represented using IEEE 754 format
IMAP	imapa imapb	7.3.9	An integer value used to represent a floating-point value
Integer	int[1]	7.3.10	A two's complement integer which ranges from a negative min value to a max value based on the number of bytes
Unsigned Integer	uint[1]	7.3.11	An integer value that ranges from 0 to a max value based on its length

[1]*Each of these types can be defined with an unknown length (as shown) or with a fixed length.*

Three of the types in Table 7-13 i.e. Floating Point, Integer and Unsigned Integer vary in the number of bits used to describe them. A metadata specification can either state a fixed number of bits, or use the non-numeric abbreviation shown in the table. In specifying a fixed number of bits, the number of bits follow the type (e.g. int32, or float64) and the number of bits must be a multiple of eight. Using the non-numeric abbreviation indicates the number of bits is defined at run-time when the data is created.

7.3.1 BER Length

Section 8.1.3.3 of ASN.1 [17] defines the Basic Encoding Rules (BER) for Length. These rules provide a method for embedding the length of an unsigned integer within the integer value itself, thus eliminating the need to define the length in an external specification, or elsewhere within the metadata. The MISP uses only the Short and Long Form BER Length methods.

7.3.1.1 Short Form

For an unsigned integer less than 128, the short form can be used. As shown in Figure 7-6, the short form is one byte, where the most significant bit (bit 7) in the byte set to zero (0) signaling short form encoding. The last seven bits encode the unsigned integer, ranging from a value of zero (0) to 127.

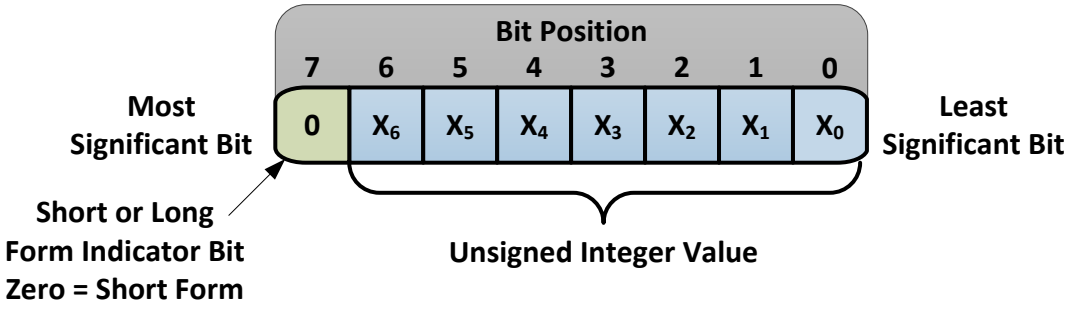


Figure 7-6: BER Short Form Encoding

The numeric value, V, is computed using Equation 10.

$$V = \sum_{i=0}^{7} x_i * 2^i \qquad \text{Equation 10}$$

7.3.1.2 Long Form

For an unsigned integer greater than or equal to 128, the long form is used. As shown in Figure 7-7, the long form uses multiple bytes, where the most significant bit in the most significant byte signals long form when set to one (1). The remaining 7 bits of the first byte ($N_0 \ldots N_6$) indicate the number of subsequent bytes used for the unsigned value. The unsigned value begins with the most significant byte (byte N) immediately following the first byte. The bytes that follow the leading byte are the encoding of an unsigned integer and consist of 8 times N number of bits. The maximum number of bytes possible for the value is 127, which would equate to a very large number (2^{127*8} -1)

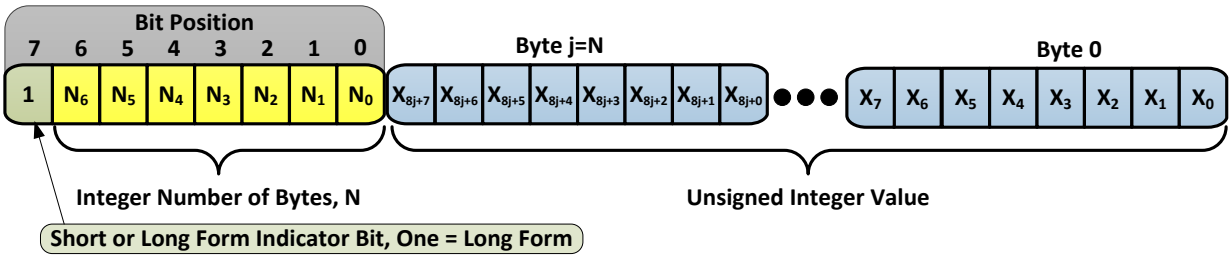


Figure 7-7: BER Long Form Encoding

The numeric value, V, is computed using Equation 11.

$$V = \sum_{i=0}^{N-1} \sum_{j=0}^{7} x_{i*8+j} * 2^{i*8+j} = \sum_{i=0}^{8N-1} x_i * 2^i \qquad \text{Equation 11}$$

7.3.2 BER-OID

The Basic Encoding Rules (BER) for an Object Identifier (OID) is defined in Section 8.19.2 of ASN.1 [18]. In a similar fashion to BER Length, these rules provide a method of embedding the length of an unsigned integer within the integer value itself, eliminating the need to have the length defined in an external specification or elsewhere within the metadata.

A BER-OID value is a series of one or more bytes, where the most significant bit (msb) of each byte signals if more bytes follow in defining a value. If the msb is one, a next byte is included; if the msb is zero, the byte is the last byte of the value. The remaining 7 bits of each byte combine into one unsigned integer. Figure 7-8 illustrates a one-byte BER-OID example, where the most significant bit (i.e. bit 7) is set to zero indicating there are no further bytes needed to represent the value. The single byte BER-OID form is an identical bit pattern to the short form BER encoding from section 7.3.1.1, including the method of computing the value; however, the meaning of bit 7 is different. Care must be taken to understand which type of BER encoding is being used.

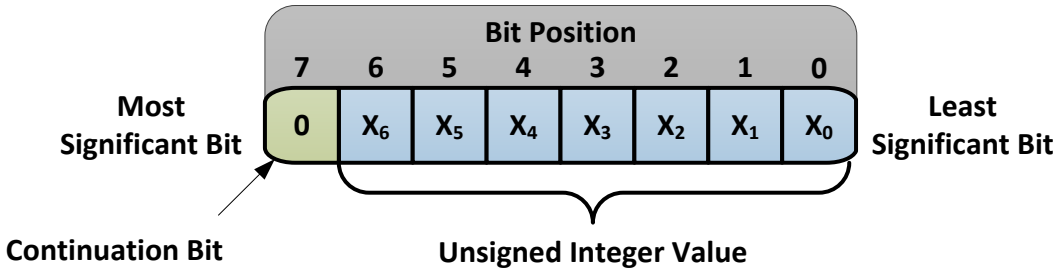


Figure 7-8: One-byte BER-OID Example

With multi-byte BER-OID values, bit 7 of each byte is the continuation bit. As shown in Figure 7-9, when the continuation bit is one, there is at least one more byte of data. A continuation bit value of zero (shown in red) signals the end of the unsigned integer. After all the bytes are identified, the remaining 7 bits of each byte are combined to construct the value.

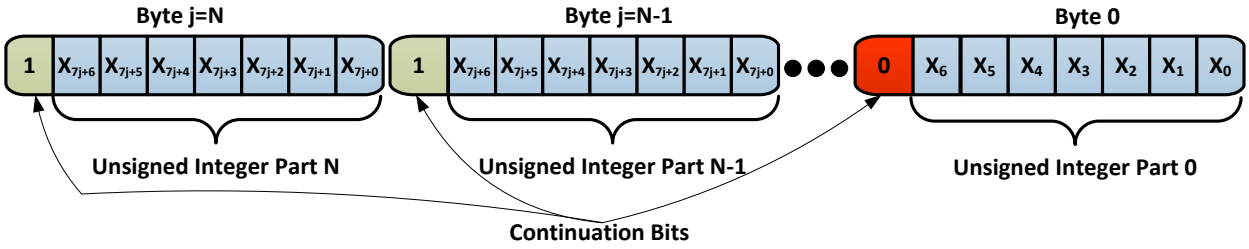


Figure 7-9: Multi-byte BER-OID Example

The numeric value, V, is computed using Equation 12.

$$V = \sum_{i=0}^{N-1} \sum_{j=0}^{6} x_{i*7+j} * 2^{i*7+j} = \sum_{i=0}^{7N-1} x_i * 2^i \qquad \text{Equation 12}$$

7.3.3 Binary

A Binary data type compacts a set of information into one or more bytes. The binary type can contain flags, small enumerations and other bit-specified controls. When MISP metadata standards use binary types, the standard describes how to interpret each bit of the binary value.

7.3.4 Character ISO7

The ISO7 data type is one or more characters from the ISO/IEC 646:1991 [19] character set. ISO7 uses the lower seven bits of a byte; the eighth bit is always zero.

ISO7 supports the Latin character set for English only. For this reason, the MISP is discontinuing ISO7 in document updates and new publications. Some existing MISP standards use ISO7 for historical and backward compatibility reasons. The replacement data type for ISO7 is UTF8.

7.3.5 Character UTF8

The UTF8 data type is one or more characters from the ISO/IEC 10646:2014 Universal Character Set (UCS) [20]. UTF8 is one of several character encoding schemes within ISO/IEC 10646:2014. UTF8 is a variable length value, which operates in a similar fashion to BER-OID encoding discussed in Section 7.3.2. With UTF8, the upper bit of the first byte determines if the character requires additional bytes. UTF8 characters use a minimum of one byte and can grow up to four bytes, which leaves a large address space for non-English character sets, such as dialects of Chinese. The first group of 127 UTF8 characters, with bit 8 equal to zero, is equivalent to ISO7.

MISP standards use UTF8 for character strings because it expands as necessary to support alternate languages in support of NATO countries.

7.3.6 Character UTF16

The UTF16 data type is one or more characters from the ISO/IEC 10646:2014 Universal Character Set (UCS) [20] UTF16 is one of several character encoding schemes within ISO/IEC 10646:2014. UTF16 is a variable length value, which operates in a similar fashion to BER-OID encoding discussed in Section 7.3.2. With UTF16, the upper bits of the first byte determine if the character uses more bytes. UTF16 characters use either a minimum of two bytes or a maximum of four bytes, leaving a large address space to support the full Unicode character set. The first group of 127 UTF16 characters, with all bits of byte one zero, and bit 8 of byte two equal to zero is equivalent to UTF8 and ISO7.

Older MISP standards have used UTF16, but all document updates and new publications will utilize UTF8 instead of UTF16.

7.3.7 Enumeration

An Enumeration type is an unsigned integer whose value maps to a predefined table of choices. A controlling document (e.g. standard) defines the range of allowed unsigned integer values beginning at zero to some maximum value along with the meanings for each value. An enumeration compacts a list of choices into a single unsigned integer value, thereby saving bytes.

7.3.8 Floating Point

The Floating-Point type is a subset of the IEEE 754-2008 Standard for Floating-Point Arithmetic and Floating-Point formats [21]. IEEE 754-2008 Floating-point numbers support decimal and binary interchange of values. The MISP standards only use binary interchange formats.

The basic form of the binary interchange format includes three values: sign bit, biased exponent and significand. IEEE 754 defines different binary interchange formats by specifying variations of the length – or number of bits – of a value, and the length for the biased exponent and significand. The MISP standards supports IEEE-754-2008 binary32 (four-byte Single Precision) and binary64 (eight-byte Double Precision) floating-point values. Future MISB standards may support binary16 (two-byte Half Precision) or binary128 (sixteen-byte Quadruple Precision) values.

IEEE-754 also defines special bit patterns for representing $\pm\infty$, Not a Number (NaN), and other values. The MISP standards do not allow these special values unless a controlling document specifies how to use the special value.

In a MISP standard, the type designator "float" indicates that an embedded method determines the length. For example, the length within a KLV triplet supplies the byte count used for the floating-point value. A MISP standard can also define a specific length for a floating-point value by appending the number of bits to use at the end of the "float" designator; for example, "float32" indicates a single precision floating-point value, and "float64" indicates a double precision floating-point value.

7.3.9 IMAP

The IMAP type is an unsigned integer, which is a mapping to a floating-point value as specified by MISB ST 1201 [22]. Knowing certain parameters (min, max, resolution) about the value enables this representation to use fewer bytes than an equivalent IEEE 754 floating-point value.

7.3.10 Integer

The Integer type is a two's complement integer, where the most significant bit (msb) is a sign bit and all remaining bits specify the integer value. The binary representation of a two's complement value is not obvious. The integer value for a two's compliment binary number with bits a_{N-1} (msb) to a_0 (least significant bit) can be computed using Equation 13. In this equation a_{N-1} is the sign bit.

$$value = -a_{N-1} * 2^{N-1} + \sum_{i=0}^{N-2} a_i 2^i \qquad \text{Equation 13}$$

Integer values have different bit lengths; the MISP Integer type defines the values in groups of eight bits, or one or more bytes. In a MISP standard, the type specifier of "int" indicates that another embedded method determines the length. For example, the length within a KLV triplet supplies the byte count for the integer value. A MISP standard can also define a specific length for an integer value by appending the number of bits to use at the end of the "int" designator; for example, "int8", "int16", "int32", etc.

7.3.11 Unsigned Integer

The Unsigned Integer type is a standard binary representation of a non-negative integer. Values range from zero to $2^N - 1$, where N is the number of bits used. The MISB defines the Unsigned Integer type in groups of eight bits, or one or more bytes. In a MISP standard, the type specifier of "uint" indicates an embedded method determines the length. For example, the length within a KLV triplet supplies the byte count for the unsigned integer value. A MISP standard can also define a specified length for an unsigned integer value by appending the number of bits to use at the end of the "uint" designator; for example, "uint8", "uint16", "uint32", etc.

7.4 SMPTE – Key, Length, Value (KLV) Metadata Formatting

The MISB has adopted KLV developed by SMPTE for transmitting and storing metadata in Motion Imagery Systems. This section briefly describes the SMPTE KLV standard as a basis for how the MISB has extended its used of KLV. For complete details about SMTPE KLV see [14].

KLV is a data encoding structure for transmission or storage. The basic concept of the KLV triplet is to encode a data value, along with a count of bytes needed to support the data value, and an identifier, into one block of binary data. The identifier links the value to its definition or meaning, and the value can be used for computation, display, etc. The identifier is the Key; the number of bytes needed to describe the data is the Length, and the data being transmitted is the Value. These three pieces of information are combined to make a single Key-Length-Value (KLV) item as illustrated in Figure 7-10.

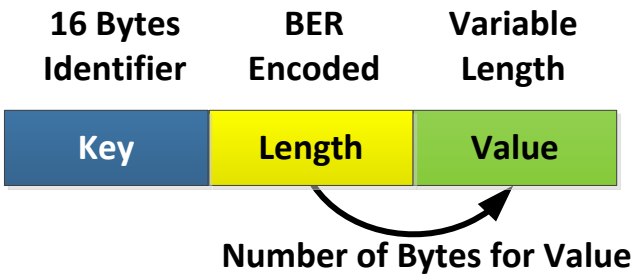


Figure 7-10: Illustration of Key Length Value Item

The Key is a reserved set of 16 bytes used as an index into a dictionary of registered Keys. The Key determines how to parse the value and provides insight into how to use the value. The byte(s) immediately following the key indicates the Length. The Length is the number of bytes needed to describe the value. For example, if a value is specified as a 32-bit integer (i.e. four bytes), the Length is four, representing the four bytes for the integer value. Immediately following the Length are the bytes for the Value. The Value is a block of bytes encoded using the rules specified for the given key. The Value can be a single numeric value (i.e. integer or float), string, a binary value, or a group of items (such as a list) of any length.

Once a Key, Length and Value combine, they form a KLV data block for transmission or storage as a singular unit. Other media embed KLV blocks as their metadata, such as Motion Imagery streams or files; or KLV blocks are used independently of any media.

7.4.1 KLV Dictionary

To insure each key is unique and properly defined, SMPTE defines a KLV Dictionary that lists all defined keys with their attributes e.g. name, data type and other information. This list provides the meaning of the value for a given key. For example, as a decoder parses a KLV block, the first item read is the KLV Key. This key is the index in the dictionary to look up the meaning of the value to determine how to process or display it. As an illustration, the key could reference the altitude of the camera as its imaging the scene, so the decoder would know how to display that information to the user (e.g. graphically or units used).

7.4.2 Key

A KLV key is a 16-byte identifier that has three components, **SMPTE KLV Identifier**, **Parsing Identifier** and **Table Lookup Identifier**, as illustrated in Figure 7-11.

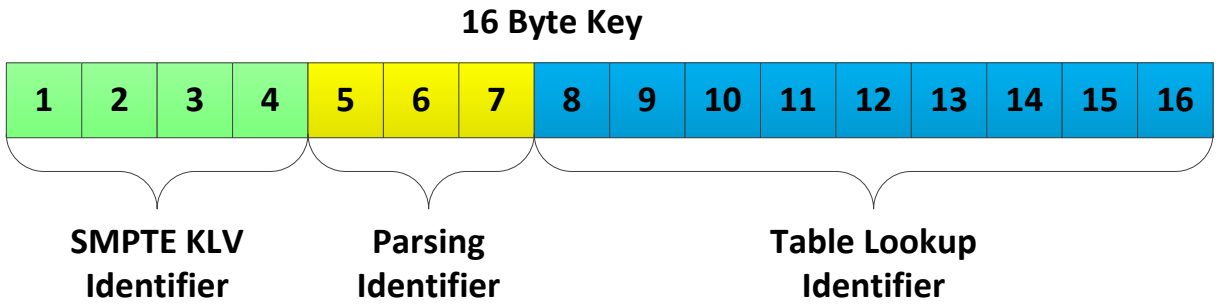


Figure 7-11: Key's Components

The **SMPTE KLV Identifier** indicates the bytes are a KLV key; these bytes are registered with ISO and are always 0x06-0E-2B-34. The **Parsing Identifier** of three bytes indicates the type the KLV value uses, either a single value, group of values or label (each discussed in Section 7.4.4). The parsing identifier can have several options with the details described in SMPTE 336 [14]. See Appendix E below for a quick reference of these options. The **Table Look-Up Identifier** is a set of bytes that reference a dictionary, which provides both syntactical and semantic information about the value.

7.4.3 Length

The length is one or more bytes encoded using BER Short or Long form encoding (see Section 7.3.1). The Length can grow as needed to specify the number of bytes for the value. A length of zero indicates a value is not included in the KLV triplet.

7.4.4 Value

The Value is a block of bytes interpreted using the information from the parsing identifier of the Key.

7.4.4.1 Single Item

A Single Item value is block of bytes convertible into one of the types listed in Table 7-13.

7.4.4.2 Group

A Group is a block of bytes containing a list of single items and potentially another group, so each group can recursively contain other groups. There are five types of groups listed in Table 7-14. The first column is the full name of the group; the second column is the short name of the group used as a type in the KLV dictionary; the final column is a short description of the group. In all cases, each group is transformable into a universal set and the items are referenceable in the KLV dictionary.

Table 7-14: Summary of KLV Groups

KLV Group Name	Short Name	Description
Universal Set	uset	Each element in the group uses a full 16-byte key, length and value. This type of group does not need a controlling document to understand how to parse the values. Parsers can extract data from these groups without any special software. Universal sets are not in use by the MISB anymore because a key is inefficient, consuming too many bytes to specify.
Global Set	gset	Same as the universal set except each element uses a subset of a KLV key. This type of group does not need a controlling document to understand how to parse the values. Parsers can extract data from these groups without any special software. Global sets are not in use by the MISB because a key is inefficient, consuming too many bytes to specify.
Local Set	ls	A Local Set requires a controlling document to define a list of predefined identifiers for each element contained in the local set. Each local set identifier maps to an element in the KLV dictionary. Local sets can specify elements in any order. The identifiers are Tags, instead of Keys. Tags are a list of serially numbered unsigned integers starting with zero. Tags can use different binary representations; however, the MISB uses BER-OID tags where possible. BER-OID tags allow sets to grow as needed without limits on a specific unsigned integer size.
Variable Length Pack	vlp	A Variable Length Pack requires a controlling document to define a list of items in a specific order. Because of the pre-defined order, the Variable Length Pack parses into a known list of items, which map into items of the KLV dictionary. Since the list is pre-defined, there is no need for a key or tag, thereby reducing the number of bytes needed to specify a value. A Variable Length Pack only specifies the length of each value followed by the value.
Defined Length Pack	dlp	A Defined Length Pack requires a controlling document to define a list of items in a specific order and with predefined lengths. Because of the pre-defined order, a Variable Length Pack parses into a known list of items, which map into items of the KLV dictionary. Since the list is pre-defined, there is no need for a key or tag, thereby reducing the number of bytes needed to specify a value. Since the length is pre-defined, there is no need for the length, thereby reducing the number of bytes needed. A Define Length Pack only specifies the values themselves without the key or length; thus, they are the most bit efficient method for specifying a group.

7.4.4.2.1 Local Set (LS)

A Local Set is a KLV construct consisting of a 16-byte key, local set length and a list of local set triplets. Each local set triplet consists of a Tag, value Length and Value as shown in Figure 7-12.

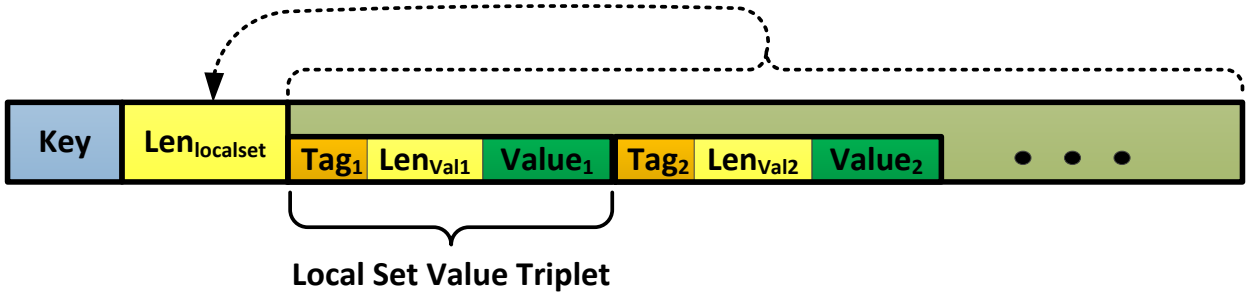


Figure 7-12: Illustration of a Local Set

7.4.4.2.2 Variable Length Pack (VLP)

A Variable Length Pack is a construct containing a 16-byte key, pack length and a list of VLP Value Pairs as shown in Figure 7-13.

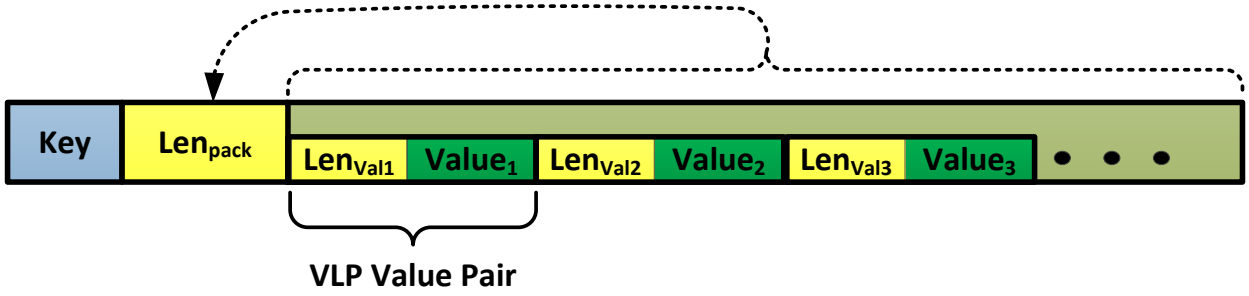


Figure 7-13: Illustration of a Variable Length Pack

7.4.4.2.3 Defined Length Pack (DLP)

A Defined Length Pack is a construct containing a 16-byte key, pack length and a list of values as shown in Figure 7-14.

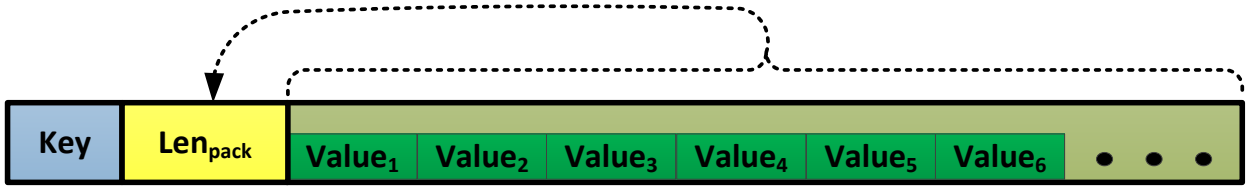


Figure 7-14: Illustration of a Define Length Pack

7.4.5 Parsing Rules

KLV parsing rules are simple; there are three items to parse: key, length and value. When parsing the key, the SMPTE KLV Identifier validates the Key is a proper KLV key. The Parsing Identifier determines how to parse the value, either a single item or group. The Table Lookup Identifier provides a lookup value into the dictionary, which supplies the specifics about the value within the dictionary. If the parsing software does not recognize the key (i.e. it is not in the

dictionary), the parser is expected to read and use the length to determine how many bytes to skip over the value; i.e. the value is ignored. The capability of skipping over unknown items enables backwards compatibility for future changes, and removes issues with fixed record lengths.

7.5 MISB KLV Extensions

The MISP standards extend and refine KLV with the overall goal of reducing the number of bits needed to convey metadata (i.e. reduced bandwidth) and improving the quality of transmission and interpretation of the metadata. These extensions/refinements are specific to the types of metadata Motion Imagery Systems produce.

7.5.1 MISB KLV Terms and Definitions

SMPTE KLV allows KLV groups within other groups forming a tree with parent/child relationships as discussed in Section 7.1.1.2. The top-most parent of the tree is the *Root,* and when the root is in a KLV stream or file, the root's data defines a *KLV Packet*. A KLV Packet always starts with a 16-byte key header and includes all children of the root). A KLV Packet is an independent element in a stream of KLV packets so that one packet does not rely on another packet for interpretation.

KLV Root (or just root): The top-most parent of a KLV hierarchy.

KLV Packet: The KLV Root in a metadata stream or file; defined by the 16-byte key and fully encompasses all data (i.e. parent data and children data) of the root.

7.5.2 MISB KLV Dictionary

MISB KLV uses definitions from two different KLV Dictionaries: the SMPTE KLV Dictionary discussed in Section 7.4.1, and the MISB KLV Metadata Dictionary MISB ST 0807 [23]. Most of the MISB KLV keys used in the MISB standards are registered in the MISB KLV Dictionary; however, to maintain backwards compatibility and for historical reasons some keys are registered in the SMPTE dictionary.

The MISB maintains its own dictionary for two reasons. First, additions and updates to the SMPTE dictionary can take a minimum of six months to a year with the attributes of the keys under scrutiny by SMPTE members, who are outside of the Motion Imagery community. In some cases, SMPTE has changed the name, type, or meaning of a proposed value. Second, the MISB applies different rules to what is registered, and the use of data types as discussed below.

7.5.2.1 KLV Value Registration

One SMPTE KLV rule is all values are to be registered within a KLV dictionary. This rule is not practical in some MISB KLV documents because of the sheer volume in registering such values.

For instance, Motion Imagery Systems rely on measurements to extract useful information about the Motion Imagery, such as geographical positions from sensor location and orientation. With any measurement, there exists some uncertainty along with cross correlation between measurements. The MISP registers the primary measurement value (i.e. latitude, longitude,

sensor depression angle, etc.) in the KLV dictionary, but not secondary statistics, such as standard deviations and cross correlations between the primary values. There is far too much overhead in registering the volume of these statistics. Cross correlation alone could produce an N^2 increase in KLV dictionary items; that is, ten primary elements would produce 100 statistic elements. Instead of registering these statistical elements, the MISP reports standard deviation and cross correlation values using MISB ST 1010 [24].

7.5.2.2 Data Type Processing

The MISB Data Types are normative, but MISB standards using Local Sets or Packs may override the format used to conserve bytes. For example, MISB standards referencing keys in the MISB dictionary which use the floating-point type may encode the floating point as an integer with techniques from Section 7.5.5. Another example is a MISB standard which references keys in the MISB dictionary using the unsigned integer type but may encode the unsigned integer as a BER-OID encoded value. With MISB standards and dictionary definitions, the standard describing how the bytes are encoded into KLV defines the format used. Where a MISB standard does not specify a special or adjusted format, the format indicated in the KLV dictionary is used.

MISB standards utilizing Universal Sets do not override the formats defined in the MISB dictionary.

7.5.3 Value Sizing

One method to reduce the number of bytes in a Local Set is to adjust the number of bytes used to represent a value based on the value. For example, if a Local Set defines a uint value and the value to encode is the number 12, only one byte is needed. The Local Set would define the length of this value to be one, resulting in three bytes to send the value 12; one byte for the tag, one byte for the length and one byte for the value.

7.5.4 MISP KLV Groups

MISP metadata is principally specified as one or more local sets, variable length packs and defined length packs for grouping items. Universal and global sets require many more bytes consuming unnecessary bandwidth.

The MISP imposes an additional rule to local sets beyond SMPTE rules. When forming a data block in a SMPTE local set, the element order can be random, so when parsing and using the data it is an unordered list. With MISP local sets, element order may be important so when parsing and using the data the order must be retained. For example, MISB ST 1010 - Standard Deviation and Cross Correlation Metadata (SDCC) requires the items in the parent local set preceding the SDCC to be in a specific order. Defined order in a local set improves the efficiency in processing the metadata. For example, knowing the time value is the first element in a local set reduces the need to search for that element amongst the local set items, thereby reducing processing time.

The defined length pack used in MISP standards have three refinements: Flexible Length Values, Floating Length Packs and Truncation Packs.

7.5.4.1 Flexible Length Values

The MISP allows flexible length values within a pack. "Flexible length" values are values where the length can be determined by the value itself. BER-OID values, see Section 7.3.2, are flexible length, because the value includes a continuation bit in each byte. As discussed previously, BER-OID is the Basic Encoding Rules for Object Identification. The MISP uses BER-OID as a method for optionally typecasting any unsigned integer, not just Identifiers. BER-OID values are very useful within defined length packs to determine the length of a value within a pack without directly specifying the length.

7.5.4.2 Floating Length Pack

A "floating length" pack is identical to a defined length pack except the last elements' length can change or "float", enabling the pack to contain another variable length item, or group within it. To compute the length of the last element, subtract the length of the first n -1 elements from the total pack length. Figure 7-15 illustrates a "floating length" pack.

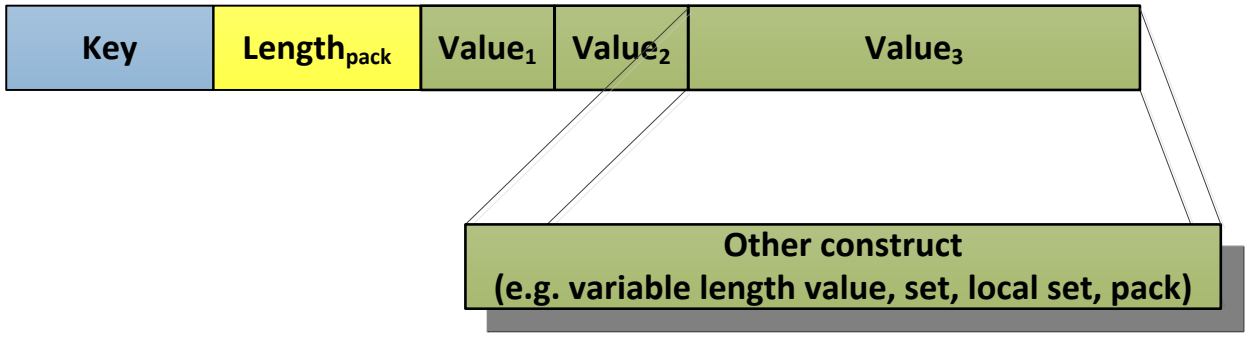


Figure 7-15: Floating Length Pack Illustration

The Key (in blue) is a pack key (16 bytes); the $Length_{pack}$ (in yellow) is the total number of bytes used by $Value_1$, $Value_2$ and $Value_3$; $Value_1$ and $Value_2$ are defined with specific lengths in the controlling document. $Value_3$ is variable in length and can contain other KLV groups (sets, packs, etc.). The length of $Value_3$ is computed by subtracting the lengths of $value_1$ and $value_2$ from the length of the pack ($Length_{pack}$): $Length_3 = Length_{pack} – (Length_1 + Length_2)$.

7.5.4.3 Truncation Packs

A "truncation" pack is a KLV defined length pack, where its length value changes based on a subset of values used. For example, Figure 7-16 shows four values in a pack; the length of this pack is 8+12+8+16 = 44 bytes.

Key	Length$_{pack}$ 8+12+8+16=44	Value$_1$ (8 bytes)	Value$_2$ (12 bytes)	Value$_3$ (8 bytes)	Value$_4$ (16 bytes)

Figure 7-16: Defined Length Pack Example

If an application decides Values 3 and 4 are not necessary, the pack can be truncated by setting the length to: 8+12 = 20 bytes as shown in Figure 7-17.

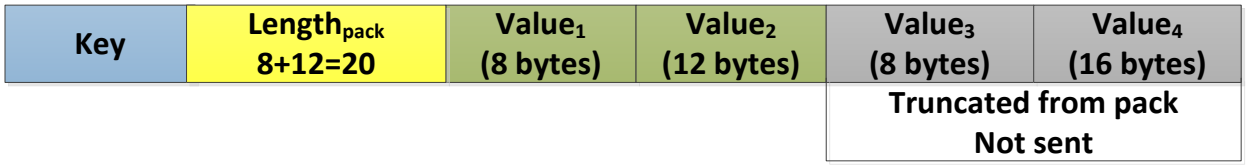


Figure 7-17: Truncation Pack Example

There are two considerations in using a truncated pack: element order and incomplete values. First, element order is important when defining a truncation pack: the most important items come first – if truncation occurs, the lower priority items are removed. Secondly, applications need to remove complete values and not partial values. If a truncation results in a partial value, decoders must skip over decoding the partial value.

An example of truncation pack order, shown in Figure 7-18, is a pack containing Lat, Lon, Elev, Sig Lat, Sig Lon, Sig Elev, Comp Method and Time offset. By locating the latitude, longitude and elevation items first, the remainder of the pack can be truncated leaving the most important items (the Lat, Long and Elev). Adjusting the $Length_{pack}$ for a smaller pack and truncating unneeded values conserves bandwidth.

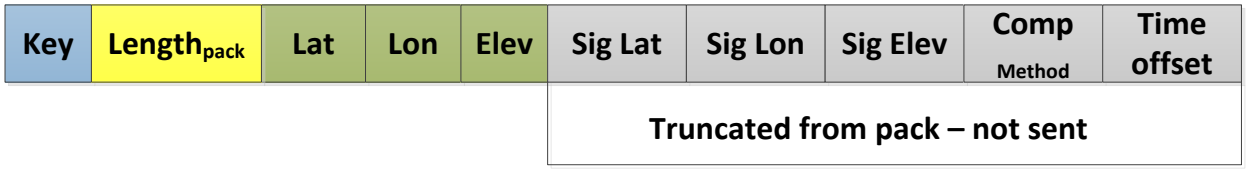


Figure 7-18: Example of Truncation Pack

All values in a pack must be supplied prior to truncating the pack - values cannot be skipped or dropped. Sometimes it is necessary to supply filler values when a pack value is unknown. Each value listed in a truncation pack's controlling document must define a filler value; an example is to use IEEE NaN for floating-point values. Figure 7-19 shows an example where the Lat/Lon, Sig Lat and Sig Lon are valid values, but the Elev value is unknown so it must use a filler value of NaN.

Figure 7-19: Example of Truncation Pack with Filler Value

7.5.5 Float to Integer Mapping

The IEEE 754 Floating-Point Standard [21] supports large ranges of values along with fractional values. In most Motion Imagery, measurement values do not need to use the full floating-point range or full fractional resolution. Any measurement value has a known min/max range and resolution. For example, an angle measure has a range from 0° to 360° and, depending on the system, a fixed measurement resolution. The min, max and resolution enable a mapping between a floating-point value and an integer representation, where the integer representation may use

fewer bytes. MISB ST 1201 Floating Point to Integer Mapping [22] defines the methods and rules for performing the mapping. Before ST 1201, some standards, such as MISB ST 0601, defined their own float-to-integer mapping for each value.

7.5.6 Data Structures

The MISP has defined standards to ensure implementers consistently work with certain common data structures: Standard Deviation and Cross Correlation (SDCC) and Multi-Dimensional Arrays.

7.5.6.1 Standard Deviation and Cross Correlation (SDCC-FLP)

The SDCC-FLP is a data structure which enables the efficient encoding of Standard Deviation and Cross Correlation information. The data structure encoded is a triangular matrix of values, where the main diagonal is the standard deviation values and the upper triangular area is the cross-correlation values. The standard deviation and cross correlation values are encoded as either IEEE floating-point values or using MISB ST 1201 Floating Point to Integer Mapping. The standard deviation values are not required to be encoded with the same method as the cross-correlation values. The cross-correlation values can be sparsely encoded, saving bytes with large cross correlation matrixes. Refer to MISB ST 1010 [24] for details on using and implementing the SDCC-FLP.

7.5.6.2 Multi-Dimensional Array Pack (MDARRAY)

The MDARRAY structure defines the efficient encoding of vectors and arrays of consistent type and consistent size data elements. The MDARRAY can encode any dimension of data elements (e.g. 1, 2, 3 (or more) dimensional data). Typical MDARRAY elements are integers, floating-point values or ST1201 Floating Point to Integer mapped values. Other types can be grouped including whole sets or packs as long as they are all the same length. Refer to MISB ST 1303 [25] for details on using an implementing the MDARRAY.

7.5.7 MISP Baseline Local Set Structure

The MISP Baseline Local Set Structure defines basic information needed in MISP local sets to enable accurate time, versioning and data reliability. Not every MISP local set uses this structure; however, updates to existing standards may enforce this structure in the future.

The MISP baseline local set structure includes three values: **Timestamp**, **Version** and **CRC** (Cyclic Redundancy Check). The **Timestamp** is defined in MISB ST 0603 [26] and provides the base time for all temporal values in a local set. Temporal values are measurements at a given time. Sub-groups within the local set may refine or override a timestamp as needed.

The **Version** value is the minor version number of the standard used to define the local set. The minor version of a document is the “decimal” value after the four-digit standard number. For example, with ST 1201.1, the minor version is the value “1” after the period. This value links a specific version of a controlling document to the metadata generated. This value informs a receiver on how to interpret the values and rules for the parsed metadata, or detect when data conforming to an older standard needs corrective processing.

The **CRC** is a value used to validate metadata delivery from a source to a receiver. In most cases, when metadata becomes corrupted or is lost, a CRC computation detects an issue. CRC’s do not

correct data; they only detect a problem, so applications are made aware and can take corrective actions, such as skipping the potentially corrupted data. CRC's apply to the root local set. A root local set is one not contained in any other group; it is parent of all other sets for a given packet of metadata. Root local sets always begin with a 16-byte local set key.

7.5.7.1 CRC Implementation

A Cyclic Redundancy Check (CRC) value is the result of an error-detecting algorithm processed over a block of bytes. The MISP baseline local set structure requires the CRC value, and therefore the CRC triplet, to be the last item of the local set. The CRC value is computed using the block of bytes starting with the first byte of the key all the way through to the last byte before the CRC value as shown in Figure 7-20.

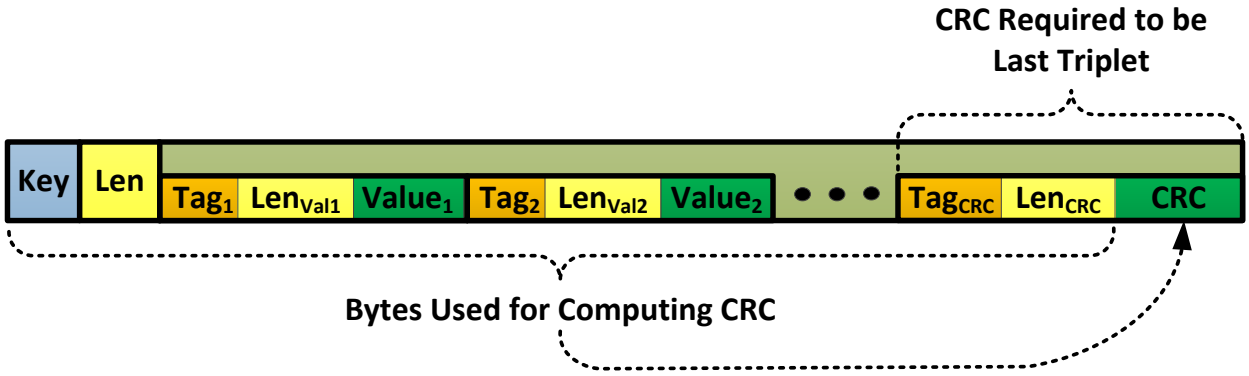


Figure 7-20: Illustration of Local Set CRC Computation

The CRC value is a two-byte value computed using the CRC-16-CCITT ($x^{16} + x^{12} + x^5 + 1$) algorithm executed over the defined data block; Appendix D provides java code, which implements the CRC-16-CCITT algorithm. Systems that build a local set compute the CRC value and include it into the local set. Users of the local set perform the same computation over the data block and compare the result to the CRC value. If the CRC matches the computed value, the metadata likely was transmitted without error. If the CRC fails to match the computed value, the whole data block is suspicious, and an error recovery process is started. The data error could have been introduced anywhere in the data block, including the lengths. If a pack length is corrupted, skipping the bad data block may skip more elements than desired, so a more extensive evaluation process should be used. Since the location of an error is unknown, a scanning process searches for the next occurrence of the SMPTE KLV Identifier (0x06-0E-2B-34) starting after the first four bytes of the key. After finding the next SMPTE KLV Identifier, normal KLV decoding continues with key validation, etc.

7.5.8 Implementation of Data Preservation and Data Sharing

The MISP supports Data Preservation and Data Sharing by using the Amend Metadata and Segment Metadata list structures discussed in Section 7.1.2. The implementation of these lists requires an additional step of processing beyond a typical SMPTE local set. MISB ST 1607 [27] defines the Amend Metadata Local Set and Segment Metadata Local Set with its governing rules for use.

7.5.9 Data Rates and Bandwidth

Bandwidth is a major issue with Motion Imagery Systems, so many of the MISP standards utilize a report-on-change system. A report-on-change system generates metadata only when values change. This reduces reporting the same value repeatedly, which wastes bandwidth. The report-on-change system works well when monitoring a stream constantly or from the start, but in many cases, users connect to a Motion Imagery source at random, thereby missing historic unchanged data. To rectify this, the MISP standards require all metadata reports at least once within a 30-second period. Thus, for users who join a Motion Imagery stream all metadata is available after each 30-second period.

Appendix A References

[1] MISB MISP-2019.1 Motion Imagery Standards Profile, Nov 2018.

[2] Chris McGlone, Manual of Photogrammetry, Sixth Edition, Bethesda Maryland: American Society for Photogrammetry and Remote Sensing, 2013.

[3] Committee on Developments in Detector Technologies; National Research Council, Seeing Photons: Progress and Limits of Visible and Infrared Sensor Arrays, Washtington D.C.: The National Academies Press, 2010.

[4] SMPTE EG 28-1993 Annotated Glossary of Esential Terms for Electronic Production.

[5] IETF RFC 768 User Datagram Protocol, Aug 1980.

[6] IETF RFC 793 Transmission Control Protocol, Sep 1981.

[7] ISO/IEC 13818-1:2015 Information technology - Generic coding of moving pictures and associated audio information: Systems.

[8] MISB ST 1402.2 MPEG-2 Transport Stream for Class 1/Class 2 Motion Imagery, Audio and Metadata, Oct 2016.

[9] IETF RFC 3550 RTP: A Transport Protocol for Real-Time Applications, Jul 2003.

[10] NIST Special Publication 330 The International System of Units (SI) 8th Edition, 2008.

[11] International Telecommunications Union ITU-R TF.460-6 Standard-frequency and time-signal emissions, 2002.

[12] IEEE Std 1003.1-2008 Standard for Information Technology – Portable Operating System Interface (POSIX®).

[13] ISO 8601:2004 Data Elements and Interchange Formats – Information interchange – Representation of Dates and Times, 2004.

[14] SMPTE ST 336:2017 Data Encoding Protocol Using Key-Length-Value.

[15] SMPTE ST 330-2004 Unique Material Identifier (UMID).

[16] MISB ST 0107.3 KLV Metadata in Motion Imagery, Nov 2018.

[17] ISO/IEC 8825-1:2015 (ITU-T X.690) Information Technology – ASN.1 Encoding Rules - Length defined in Section 8.1.3.3.

[18] ISO/IEC 8825-1:2015 (ITU-T X.690) Information Technology – ASN.1 Encoding Rules - Sub-Identifier defined in Section 8.19.2 .

[19] ISO/IEC 646:1991 Information Technology – ISO 7-bit coded character set for information exchange.

[20] ISO/IEC 10646:2014 Information technology – Universal Coded Character Set (UCS).

[21] IEEE 754-2008 Standard for Floating-Point Arithmetic [and Floating-Point formats].

[22] MISB ST 1201.3 Floating Point to Integer Mapping, Oct 2017.

[23] MISB ST 0807.22 MISB KLV Metadata Registry, Jun 2018.

[24] MISB ST 1010.3 Generalized Standard Deviation and Correlation Coefficient Metadata, Oct 2016.

[25] MISB ST 1303.1 Multi-Dimensional Array Pack, Nov 2018.

[26] MISB ST 0603.5 MISP Time System and Timestamps, Oct 2017.

[27] MISB ST 1607 Constructs to Amend/Segment KLV Metadata, Oct 2016.

Appendix B Acronyms

CCD	Charge Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
EG	Engineering Guideline
EMR	Electro-Magnetic Radiation
FMV	Full Motion Video
HF	High Frequency
HSI	Hyper-Spectral Imagery
IP	Internet Protocol
IR	Infrared
ISO	International Standards Organization
KLV	Key Length Value
LIDAR	Light Detection and Ranging
MI	Motion Imagery
MISB	Motion Imagery Standards Board
MISP	Motion Imagery Standards Profile
MPEG	Moving Pictures Expert Group
MSI	Multi-Spectral Imagery
MTU	Maximum Transmission Unit
RADAR	Radio Detection and Ranging
RFC	Request for Comment
RGB	Red Green Blue
RP	Recommended Practice
RTCP	Real Time Control Protocol
RTP	Real Time Protocol
SAR	Synthetic Aperture Radar
SDI	Serial Digital Interface

SMPTE	Society of Motion Picture and Television Engineers
ST	Standard
TCP	Transmission Control Protocol
TS	MPEG-2 Transport Stream
UDP	User Data Protocol
UHF	Ultra-High Frequency
UTC	Coordinated Universal Time
VHF	Very High Frequency

Appendix C Pseudocode Description

In parts of the Handbook Pseudocode is used to describe algorithms. The purpose of this appendix is to describe the notation used. Text below in brackets, < >, are symbol names.

Syntax	Meaning

Function <Name>(<P1>, …)
  <operations>
  Return <result>
End
	Definition of a function:
	  <result> is the returned value the function with the given name <Name> and parameters <P1>,<P2>…
	  The "Return" can happen anywhere within the operations.

If <condition> then
  <condition true operations>
End else
  <condition false operations>
End if
	Traditional if/then/else statement (see any computer language text book):
	  <condition> is any boolean logical statement.
	  <condition true operations> statements performed if <condition> is true.
	  <condition false operation> statements performed if <condition> is false.
	  else component is optional.

(<condition>)	Logical equation: if (condition is true) then result=1 else result=0.
a % b	Modulus (or remainder) operation: Remainder of the division of a/b where a and b are integers
a+=b	Accumulation operation: Equivalent to a=a+b
Floor(a) or $\lfloor x \rfloor$	Floor operation: Defined as rounding any non-integer value down towards -∞. The notation used is $\lfloor x \rfloor$ and the floor function is defined as: $\lfloor x \rfloor = \max\{m \in \mathbb{Z} \mid m \leq x\}$. Examples: $\lfloor -1.1 \rfloor = -2$; $\lfloor 1.1 \rfloor = 1$. Note: Microsoft Excel (pre 2010) does not perform this operation correctly.
a.name	Record construct: "a" is a record that contains multiple parameters, in this example, "name" is a parameter
a[x]	Array notation (zero based index): Element "x" of array "a"

Appendix D CRC Algorithm

The CRC algorithm detects corruptions when transmitting data from one point to another. The algorithm chosen for MISP is CRC-16-CCITT. The following is an example implementation in Java. This example contains some test values for validation.

```
public class CRC_16_CCITT {
  public final static int[] table = createTable(4129, 16, 8);
  /**
  ****************************************************************************
  * createTable - Makes the CRC table
  ****************************************************************************
  */
  private static int[] createTable(int poly, int width, int lookUpSize) {
    int size = (int)Math.pow(2, lookUpSize);
    int[] table = new int[size];
    for(int i = 0; i < size; i++) {
      table[i] = tableElement(i, poly, width, lookUpSize);
    }
    return table;
  }

  /**
  ****************************************************************************
  * tableElement - Makes the table element
  ****************************************************************************
  */
  private static int tableElement(int ival, int poly, int width, int lookUpSize) {
    int topBit = (int)Math.pow(2, (width - 1));
    int index = ival * ((int)Math.pow(2, lookUpSize));
    int mask = (int)Math.pow(2, width) - 1;
    for(int i = 0; i < lookUpSize; i++) {
      if((topBit & index) != 0) {
        index = (index * 2) ^ poly;
      } else {
        index *= 2;
      }
      index &= mask;
      // System.out.println(topBit + ", " + index + ", " + mask);
    }
    return index;
  }
  /**
  ****************************************************************************
  * getCRC - Gets the CRC value for this byte array
  ****************************************************************************
  */
  public static int getCRC(byte[] key) {
    // poly = x^16+x^12+x^5+1
    int tVal = table[255];
    int aVal = (tVal >> 8);
    int lastBVal = (tVal & 255);
    int lookUp = (aVal ^ 255);
    int bVal;
    for(int i = 0; i < key.length; i++) {
      tVal = table[lookUp];
      aVal = (tVal >> 8);
      bVal = (tVal & 255);
      lookUp = aVal ^ lastBVal ^ key[i] & 0xFF;
      lastBVal = bVal;
    }
    tVal = table[lookUp];
    aVal = (tVal >> 8);
    bVal = (tVal & 255);
    lookUp = (aVal ^ lastBVal);
    int returnVal = ((lookUp * 256) ^ bVal);
```

```
    return returnVal;
  }
  /**
  ********************************************************************************
  * printVal
  ********************************************************************************
  */
  public static String strVal(int val) {
    return  val + " ("+ Integer.toHexString(val).toUpperCase() + ")";
  }
  /**
  ********************************************************************************
  * main
  ********************************************************************************
  */
  public static void main(String argv[]) {
    System.out.println("CRC Tests\n table:");
    //for(int x = 0; x < (table.length - 1); x++) {
    //  System.out.print(table[x] + ",");
    //}
    System.out.println(table[table.length - 1]);
    System.out.println("Table.length=" + table.length);
    String s = "";
    for(int x = 0; x < 256; x++) {
      s += "A";
    }
    System.out.println(" test 1 --> " + strVal(getCRC(s.getBytes())) +" --> 256 A's, should be
59704 (0xE938) ");
    byte[] shorty = {3, 5, 11};
    System.out.println(" test 2 --> " + strVal(getCRC(shorty)) + " --> 3 5 11, should be 1730");
    byte[] shorty2 = {65};
    System.out.println(" test 3 --> " + strVal(getCRC(shorty2)) + " --> 65, should be 38009
(0x9479)");
    s = "123456789";
    System.out.println(" test 4 --> " + strVal(getCRC(s.getBytes())) + " --> chars 1 to 9, should
be 58828 (0xE5CC)");
  }
}
```

Appendix E KLV-Key Quick Reference Chart

The following page contains a printable KLV-Key Quick Reference Chart to aid in understanding the meaning behind the first six bytes of KLV Keys.

KLV-Key Quick Reference Chart

↧	Used by MISB (Y=Yes, H=Historical or Deprecating (not used anymore), Blank=Not Used)						
	06	OID					
	06	0E					
	06	0E	UL Size				
	06	0E	2B	UL Code			
	06	0E	2B	34	SMPTE Designator		
	06	0E	2B	34	01	Dictionaries	
Y	06	0E	2B	34	01	01	Metadata
	06	0E	2B	34	01	02	Essence
	06	0E	2B	34	01	03	Control
	06	0E	2B	34	01	04	Types
	06	0E	2B	34	02	Groups	
H	06	0E	2B	34	02	01	Universal Sets
	06	0E	2B	34	02	02	Global Sets - Length ASN.1 BER short or long
	06	0E	2B	34	02	22	Global Sets. Length = 1 Byte
	06	0E	2B	34	02	42	Global Sets. Length = 2 Bytes
	06	0E	2B	34	02	62	Global Sets. Length = 4 Bytes
Y	06	0E	2B	34	02	03	Local Sets. Length = ASN.1 BER Short or long, Tag = 1 Byte
Y	06	0E	2B	34	02	0B	Local Sets. Length = ASN.1 BER Short or long, Tag = ASN.1 OID BER
	06	0E	2B	34	02	13	Local Sets. Length = ASN.1 BER Short or long, Tag = 2
	06	0E	2B	34	02	1B	Local Sets. Length = ASN.1 BER Short or long, Tag = 4
	06	0E	2B	34	02	23	Local Sets. Length = 1, Tag = 1
H	06	0E	2B	34	02	2B	Local Sets. Length = 1, Tag = ASN.1 OID BER
	06	0E	2B	34	02	33	Local Sets. Length = 1, Tag = 2
	06	0E	2B	34	02	3B	Local Sets. Length = 1, Tag = 4
	06	0E	2B	34	02	43	Local Sets. Length = 2, Tag = 1 Byte
	06	0E	2B	34	02	4B	Local Sets. Length = 2, Tag = ASN.1 OID BER
	06	0E	2B	34	02	53	Local Sets. Length = 2, Tag = 2
	06	0E	2B	34	02	5B	Local Sets. Length = 2, Tag = 4
	06	0E	2B	34	02	63	Local Sets. Length = 4, Tag = 1
	06	0E	2B	34	02	6B	Local Sets. Length = 4, Tag = ASN.1 OID BER
	06	0E	2B	34	02	73	Local Sets. Length = 4, Tag = 2
	06	0E	2B	34	02	7B	Local Sets. Length = 4, Tag = 4
Y	06	0E	2B	34	02	04	Variable Length Packs. Length = ASN.1 BER short or long
	06	0E	2B	34	02	24	Variable Length Packs. Length = 1 Byte
	06	0E	2B	34	02	44	Variable Length Packs. Length = 2 Byte
	06	0E	2B	34	02	64	Variable Length Packs. Length = 4 Byte
Y	06	0E	2B	34	02	05	Defined Length Packs
	06	0E	2B	34	03	Wrappers and Containers	
	06	0E	2B	34	03	01	Simple Wrappers and Containers
	06	0E	2B	34	03	02	Complex Wrappers and Containers
	06	0E	2B	34	04	Labels	
	06	0E	2B	34	04	01	Labels Register