MISB MOTION IMAGERY STANDARDS BOARD **STANDARD** **Video Moving Target Indicator Metadata**	**MISB ST 0903.6** **21 October 2021**

1 Scope

This standard defines Video Moving Target[1] Indicator (VMTI) metadata for describing both moving and static objects such as a vehicle in motion or labeling buildings within the scene. Systems may use VMTI metadata to support populating situational awareness products, Common Operating Pictures with VMTI and track overlays on Motion Imagery, input to tracking and data fusion systems (e.g., NATO STANAG 4676 [1] ISR tracking systems), storing the results of Artificial Intelligence/Machine Learning (AI/ML) processing, or for other purposes consistent with the VMTI Processing Model.

The VMTI metadata extends and applies MISB Key-Length-Value (KLV) encoding, which is a derivation of SMPTE KLV. This standard defines VMTI metadata as a KLV Local Set (LS), which allows use as either an independent standalone metadata set or as subordinate to other MISB standards, which results in efficiencies through the leveraging of parent metadata items.

Note: This version is conditionally backward compatible with versions of this standard prior to ST 0903.6. See Appendix C.

2 References

MISB references cited here-in reflect versions current to the publication date of a document. In the event of a MISB document correction, the corrected document will have a single letter Minor Version appended to the Major Version number per the MISB Document Development Process [2]. For example, corrections to ST 0001.2, which has a Major Version of 2, result in a new version titled ST 0001.2a, which includes a Minor Version of "a". The MISB will not update the referring document (this document) with the Minor Version number change. When acquiring any MISB reference listed below from an NGA repository the latest version may be either a Major or Minor Version.

[1] STANAG 4676 NATO Intelligence, Surveillance and Reconnaissance Tracking Standard, Edition 2, 13 Oct 2021.

[2] MISB Document Development Process, Feb 2020.

[1] The term "target" rather than "object" aligns to a radar operation which discriminates a "target" from its background. Doppler radar detects moving objects resulting in "moving target indications" or MTI. In this context, "target" is used here for consistency.

[3] MISB ST 0601.17 UAS Datalink Local Set, Oct 2020.
[4] "OWL 2 Web Ontology Language Document Overview W3C Recommendation," 11 Dec 2012. [Online]. Available: https://www.w3.org/TR/owl2-overview/.
[5] MISB MISP-2022.1: Motion Imagery Handbook, Oct 2021.
[6] MISB ST 0107.5 KLV Metadata in Motion Imagery, Oct 2021.
[7] MISB ST 1201.5 Floating Point to Integer Mapping, Jun 2021.
[8] MISB ST 0603.5 MISP Time System and Timestamps, Oct 2017.
[9] MISB ST 0807.25 MISB KLV Metadata Registry, Jun 2020.
[10] MISB ST 1204.3 Motion Imagery Identification System (MIIS) Core Identifier, Feb 2020.
[11] ISO/IEC 9834:2014 Information technology - Procedures for the operation of object identifier registration authorities - Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers.
[12] Smith, et. al., "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration," *Nature Biotechnology*, vol. 25, no. 11, pp. 1251-1255, 2007.
[13] OBO Foundry, "Ontology Tools and Resources," [Online]. Available: http://www.obofoundry.org/resources.
[14] W3C, "SKOS Simple Knowledge Organization System Namespace Document," 18 Aug 2009. [Online]. Available: https://www.w3.org/2009/08/skos-reference/skos.html.
[15] W3C, "RDF Schema 1.1," 25 Feb 2014. [Online]. Available: https://www.w3.org/TR/rdf-schema.

3 Revision History

Revision	Date	Summary of Changes
ST 0903.6	10/21/2021	• Removed VTrack LS • Deprecated Requirements -4, -5, -8, -9, -11, -12, -16, -20, -21, -22, -23, -24, -25, -28, -29, -30 through -34, -41 through -55, -57, -58, -60, -65, -67, -68, -69, -70, -71, -72, -76, -77, -78, -79, -80, -81, -84, -86, -87, -88, -89, -90, -91, -95, -96, -109 through 115 • Added Requirements -116 through -143 • Deprecated VMTI LS Item 7 motionImageryFrameNum • Deprecated VTarget items 21_FpaIndex Pack, 102_VObject LS & 103_VFeature LS • Deprecated VObject items 01_ontology & 02_ontologyClass • Deprecated VFeature items 01_schema & 02_schemaFeature • Deprecated VTracker items 02_detectionStatus, 06_algorithm & 08_numTrackPoints • Added Item 23_detectionStatus to VTarget • Added items 05_version & 06_label to Ontology LS • Added items 03_onlology & 04_confidence to VFeature LS • Renamed VTracker item 05_boundarySeries to 05_trackBoundarySeries • Renamed VTarget items as follows:

Revision	Date	Summary of Changes
		02_boundaryTopLeft to 02_boundingBoxTopLeftPixel, 03_boundaryBottomRight to 03_boundingBoxBottomRightPixel, 13_boundaryTopLeftLatOffset to 13_boundingBoxTopLeftLatOffset, 14_boundaryTopLeftLonOffset to 14_boundingBoxTopLeftLonOffset, 15_boundaryBottomRightLatOffset to 15_boundingBoxBottomRightLatOffset, 16_boundaryBottomRightLonOffset to 16_boundingBoxBottomRightLonOffset, 18_targetBoundarySeries to 18_geospatialContourSeries • Renamed VMask LS Item 01_polygon[] to 01_pixelContour[] • Renamed VTracker items 03_startTime to 03_firstObsvTime & 04_endTime to 04_latestObsvTime • Renamed target states: Active to Active-Moving & Stopped to Active-Stopped; Added Active-Coasting state • Changed BoundarySeries to support only non-planar polygons (i.e., no 3D polygons) • Revised Ontology LS guidance • Updated all URIs to IRIs • Updated VTarget LS's targetId length to support ids up to 9 bytes • Updated Ontology LS's ontologyId & parentId to support ids up to 8 bytes • Reorganized/edited document contents to improve readability; updated/added graphics; added background material to facilitate understanding • Changed "Tag" to "Item" to refer to Tag-Length-Value triplet • Added acronyms, updated definitions • Added/Updated references
	01/21/2022	• Editorial changes; updated Figure 16; corrected names in Section 10.9, clarified instructions in Table 21 • Changes to KLV Figure colors to be consistent with MI Handbook: Figure 9, Figure 12, Figure 14, Figure 15, Figure 16, Figure 17, Figure 18, Figure 19, Figure 21, Figure 22, Figure 23 & Figure 24

4 Acronyms and Definitions

AI/ML	Artificial Intelligence / Machine Learning
BER	Basic Encoding Rules for Object Identification
DLP	Defined Length Pack
EO	Electro-Optic
EON	Electro-Optic Narrow
EOW	Electro-Optic Wide
FOV	Field of View
FPA	Focal Plane Array
FPS	Frames per second

GML	Geography Markup Language
HFOV	Horizontal Field of View
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IRI	Internationalized Resource Identifier
ISO	International Organization for Standardization
KLV	Key-Length-Value
LS	Local Set
LVMI	Large Volume Motion Imagery
MIIS	Motion Imagery Identification System
MISB	Motion Imagery Standards Board
MISP	Motion Imagery Standards Profile
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
RFC	Request For Comments
SMPTE	Society of Motion Picture and Television Engineers
ST	Standard
TLV	Tag-Length-Value
TS	MPEG-2 Transport Stream
UAS	Unmanned Aerial / Airborne System
UL	Universal Label
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VFOV	Vertical Field of View
VLP	Variable Length Pack
VMTI	Video Moving Target Indicator
W3C	World Wide Web Consortium

bounding box	The smallest rectangular area around a target, where the edges of the bounding box are parallel with the edges of the image frame (i.e., frame aligned)
embedded-VMTI	A VMTI LS included as a child to a parent metadata set which may leverage specific data within the parent set.
parent set	A metadata set with a metadata item to specifically include another LS.
user-MI	Motion Imagery in a stream e.g., MPEG-2 Transport Stream (TS) not associated with the VMTI-imagery which users use in conjunction with VMTI.
standalone-VMTI	A VMTI LS which is self-contained and does not rely on external metadata.
VMTI-MI	Motion Imagery input into a VMTI system.

VMTI-MI-Timestamp	When a VMTI system processes a VMTI-MI's image to create a VMTI LS packet, the VMTI-MI-Timestamp is the Precision Time Stamp of the VMTI-MI's image.

5 Introduction

Object detection and tracking provides a rich source of intelligence for determining activities in the scene. Standardized methods to describe and disseminate such information enables sharing and reuse during a mission, thereby enriching intelligence analysis. This document defines a metadata standard for characterizing objects in Motion Imagery and associated indicators of their motion. This metadata standard specifies the constructs for reporting the motions of entities, the history of their motions, and the types of the entities reported.

The VMTI LS offers a rich set of object detection qualities for identifying and detailing a multitude of objects within a single image or across frames of a Motion Imagery stream[2]. As such, this standard optimizes the VMTI metadata for situations where the metadata accompanies Motion Imagery over low bandwidth transmission links. Leveraging existing metadata within a parent MISB ST 0601 LS [3] is one way to reduce the data required. Appendix B provides additional guidance for minimizing data overhead.

6 VMTI Scenarios

The VMTI standard defines metadata to support two use-case scenarios: 1) detection of objects (i.e., targets) within Motion Imagery frames, and 2) detection and tracking of targets occurring over a time sequence of Motion Imagery frames. Figure 1 shows a nominal VMTI processing model for scenario 1 -- detection of objects.

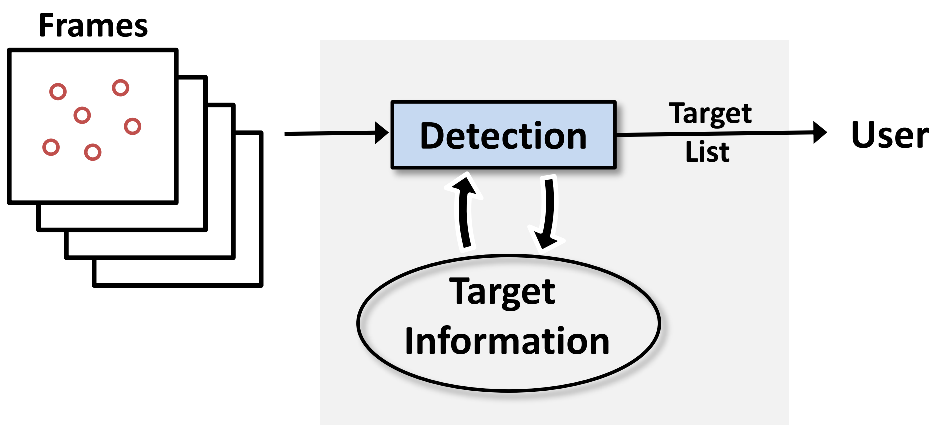


Figure 1: Scenario 1 – Nominal VMTI Processing Model

[2] The MISB prefers the term "Motion Imagery" rather than "Video" as Motion Imagery has strict criteria, such as providing intelligence information. The Motion Imagery Handbook discusses these differences. As this standard preceded this preference in terminology, and to provide continuity, this document continues with video in deference to Motion Imagery.

The input to the VMTI processing model is a sequence of Motion Imagery frames which may have zero, one, or more targets (shown as small circles). With scenario one, the Detection system detects targets and stores state information about each target. The system compares current target detections with earlier target detections to decide if it needs to a) update the target's information, or b) create a new target. Target information provides the details about the target, for example: location of target in the image and/or on the ground, target information such as shape and color, target priority, etc.

The VMTI processing model assumes systems maintain a lifecycle with state information for each target, which, at a minimum has the states described in Section 7.2. The model assumes systems constantly manage the target information, either updating a target's state, adding new targets, or removing targets. The methods of management, including timeouts for targets, are implementation dependent. At some frequency, decided by the system, the system generates a list of targets from the state information and reports them to the end user. The system decides which targets to report based on priority and bandwidth; therefore, the system does not need to report all targets in the target list. The target list does not include targets in the Dropped state.

Figure 2 shows a nominal VMTI processing with tracking model for scenario 2, which adds an optional Tracking system (shown with the dotted lines) producing a track for each target in the target list. When reporting the target list to the user, the target information includes the track data for each target. VMTI track data includes track identifier, position and time history, confidence, target kinematics, etc.

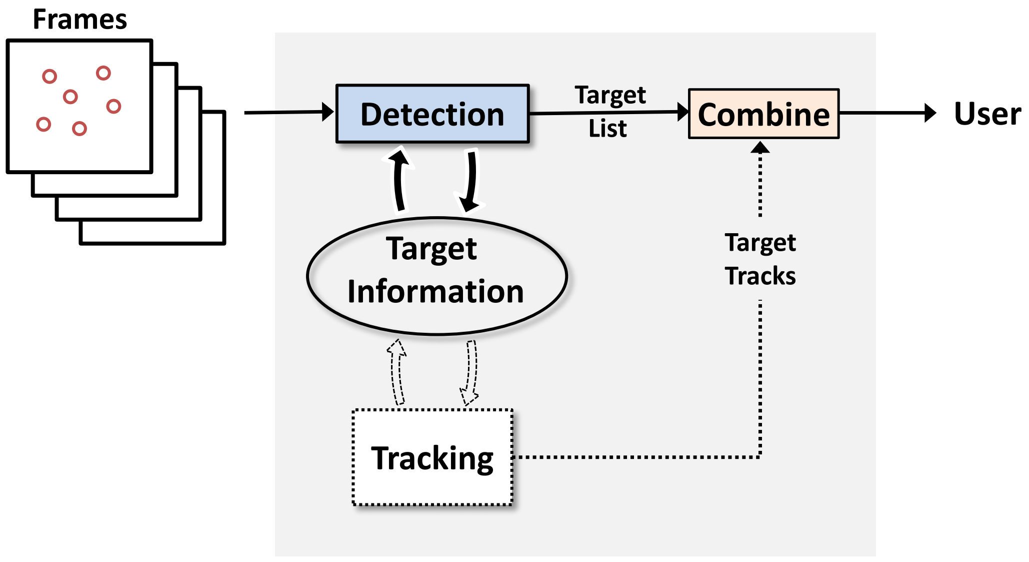


Figure 2: Scenario 2 – Nominal VMTI Processing with Tracking Model

7 VMTI Processing Model

The VMTI Processing Model manages a list of targets which originated from the Motion Imagery. Section 7.1 supplies an overview of what a target is and its attributes. Section 10.2 supplies details of a target's attributes and their encodings in KLV.

Each target has a lifecycle, starting at target creation and ending with the system dropping (forgetting) the target. Section 7.2 provides an overview of the target lifecycle.

7.1 Target Overview

Motion Imagery is a series of images, where each image is a picture of the scene. The scene has two types of objects of interest: Stationary Targets and Moving Targets. Stationary Targets are any object in the image which the system decides is important to detect and the object is not moving relative to the scene. Stationary objects include buildings, roads, bridges, parked vehicles, storage containers, etc.

Moving targets are any object in the image which the system decides is important to detect and is showing movement relative to the scene. Moving targets include vehicles driving, people walking, etc.

Each target, either stationary or moving, has a set of attributes organized into four groups: **Administrative**, **Image**, **Geospatial**, and **Tracking**.

7.1.1 *Administrative*

Administrative attributes provide information about the detection for management of the target. When the system detects a new target, the system assigns each target a unique identifier and sets the detection status to Active. Additionally, the system may assign priority, confidence, history, and an algorithm description. Table 1 provides short descriptions for each of these attributes.

Table 1: Target Attributes - Administrative

Attribute	Description
Target Identifier	The target's identifier is unique and does not change during the lifecycle of the target. The VMTI Processing Model assumes the VMTI system assigns an identifier to each target.
Detection Status	The detection status for these targets is set to indicate the target is currently visible. The detection status is the state value in the target lifecycle. Section 7.2 defines the full target lifecycle.
Priority	The priority of the target is the importance the system assigns the target for a given mission or purpose. For example, if the priority of the mission is pickup trucks, then pickup truck targets would have higher priority than all other vehicles.
Confidence	The confidence of the target is the level of certainty there is a detection.
History	The history is a count of detections over time. For example, if a system detects a target in 50 frames of the last 100 frames, the history is set to 50.
Algorithm Description	The algorithm specifies the algorithm name the system used to detect the target.

7.1.2 *Image*

Image attributes provide information about the target within the context of the image, and thus, the pixels themselves. The image attributes for a target are pixel contour, pixel bounding box, centroid, percentage of target pixels, color, intensity, image chip, image mask, features, and object label. Figure 3 illustrates an image with three targets and shows some of the image attributes. The blue value pairs at the corners are the pixel coordinates; all image positions are relative to these corners.

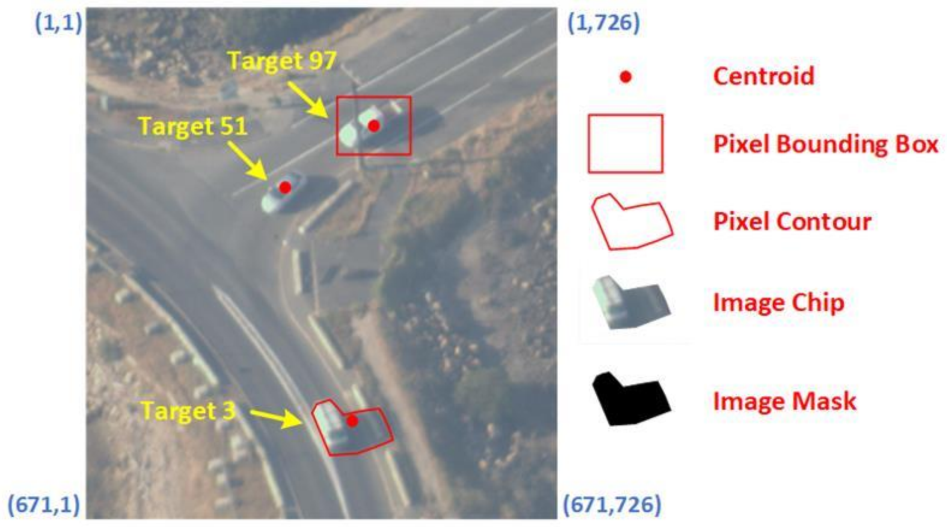


Figure 3: Image Attributes

Table 2 provides short descriptions for each of the image attributes.

Table 2: Target Attributes - Image

Attribute	Description
Pixel Contour	The pixel contour is the closed pixel polygon encompassing the target within the scene. Figure 3's Target 3 illustrates the contour of the van, where each point in the polygon are pixel coordinates. In this example, the Target 3 detection includes the van's shadow.
Pixel Bounding Box	The pixel bounding box is a two-point rectangle encompassing the target, with the top, bottom, left, and right sides parallel to the frame. The bounding box's corner values are the minimum and maximum pixel locations of the pixel contour. Figure 3 shows the bounding box of Target 97.
Centroid	The centroid is the center position within the image of the target's area. Figure 3 shows centroids as red dots for each target. The centroid for Target 97 is the center of its bounding box. The centroid for Target 51 is the object's center point (as determined by the system). The centroid for Target 3 is the center of mass of the pixel contour (in pixels), which in this example includes the van's shadow.
Percentage of Target Pixels	The percentage of target pixels is the ratio of pixels describing a target to the number of pixels within the bounding box, multiplied by 100 to make it a percentage.
Color	The color is the dominate color of the target. For example, tracking a red car will have this attribute set to the color code for red.
Intensity	The intensity is the dominant intensity of the target primarily for tracking targets in infrared or monochrome. When detecting targets in color imagery, use the Color attribute.
Image Chip	An image chip is a small image of the target, cropped either from the original image, from another frame, or even from an image from another sensor. Figure 3 shows an example image chip of Target 3.
Image Mask	Image mask is a bitmask describing an area of the original image that includes only the target. Figure 3 shows an example image mask of Target 3.

Features	Features are a list of attributes about the target. For example, an observation that Target 97, a truck, does not have a load of materials in its bed.
Object Label	The object label defines the type of object the target is. For example, Target 97 is a truck, Target 51 is a car, and Target 3 is a van. See Section 7.3 for more information.

7.1.3 Geospatial

Geospatial attributes provide information about the target within the scene. The geospatial attributes for a target are location, geospatial bounding box, and geospatial contour. Table 3 provides short descriptions for each of the geospatial attributes.

Table 3: Target Attributes - Geospatial

Attribute	Description
Location	The location is the geospatial location of the target's centroid within the scene, i.e., the latitude, longitude, and height above ellipsoid (HAE) of the target centroid.
Geospatial Bounding Box	The geospatial bounding box is a two-point rectangle encompassing the target specified in geospatial coordinates. Figure 3's Target 97 illustrates a pixel bounding box in the image; however, the associated geospatial bounding box is in the scene and on the earth's surface, therefore the coordinates for the two corners are in geospatial coordinates.
Geospatial Contour	The geospatial contour is the geospatial closed polygon encompassing the target within the scene. Figure 3's Target 3 illustrates a contour in the image; however, the geospatial contour is in the scene and therefore the points are in geospatial coordinates.

7.1.4 Tracking

Tracking attributes only apply to systems that construct tracks from multiple detections. The VMTI Processing Model does not require performing the detection process and tracking process at the same time. For example, a VMTI system may create detections in real-time on-board an aircraft, while the tracking process may operate at a ground control station, taking time to analyze larger temporal windows of the detections and build tracks.

Tracking attributes provide historical context to a detection. When tracking, the system may assign a unique track identifier and detection status. Tracking algorithms may identify two or more historical detection lists which are the same object; therefore, tracking may be the combination of two object lists with different target identifiers (see Table 1). The unique tracking identifier is a way of merging them into one track. The detection status has the same state model as a detection in Table 1.

Additional attributes include the observation time, a list of the historical track points of the detection, a bounding area enclosing a track, a confidence in the track, the count of track points (or detections), the current velocity and acceleration of the target, and the name of the tracking algorithm. Table 4 provides short descriptions for each of the tracking attributes.

Table 4: Target Attributes - Tracking

Attribute	Description
Track Identifier	The track identifier is a unique identifier of the track.
Detection Status	The detection status is the state value in the target lifecycle. Section 7.2 defines the full target lifecycle.
Observation Time	The observation time is the time of the first detection and the time of the latest detection.
Track Points List	The track points are the list of geospatial locations in the track.
Bounding Area	The bounding area is the geospatial bounding box around the detection centroids for all the detections in the track. For every track point added to the tracks points list, the system redefines the geospatial bounding box.
Confidence	The confidence is the level of certainty of the track.
Kinematics	The kinematics is the current velocity and acceleration of the target.
Algorithm	The algorithm is the name of the tracking algorithm.

7.2 Target Lifecycle

The detection system detects targets (such as vehicles) within a frame using various system dependent algorithms. The original purpose of the VMTI standard was to support "Moving Target Indicator (MTI)" detections; however, the concept of a target has evolved to include moving objects (e.g., vehicles) and static objects, such as roads, buildings, and bridges. Each target has a lifecycle, and this standard represents the lifecycle as a state machine with five detection states: Active-Moving, Active-Stopped, Active-Coasting, Inactive, and Dropped. Figure 4 illustrates these states and their interconnectivity and Table 5 provides the definitions. Systems may use different or more detailed states internally, but when reporting a target, the target's state needs to map to the VMTI states.

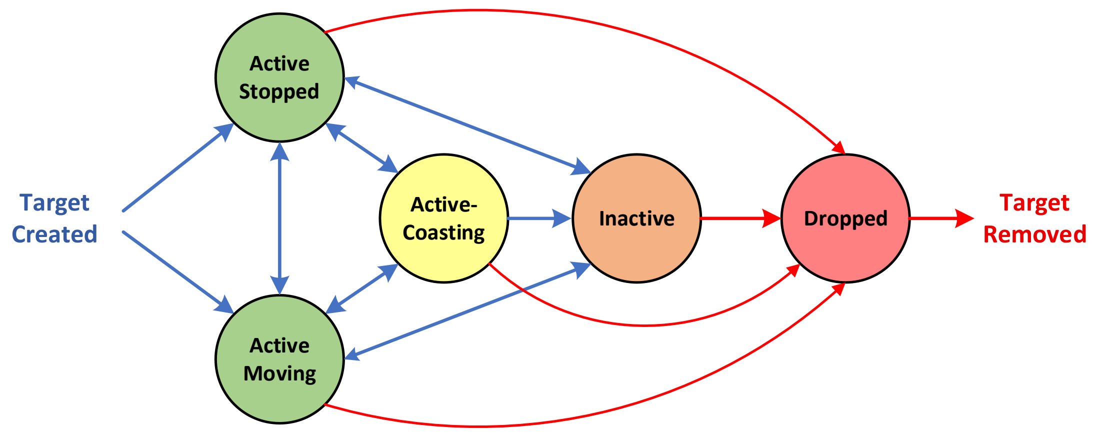


Figure 4: Detection States in Target List

Table 5: Detection Status Values

Value	Status	Description
0	Inactive	The target's coasting time has expired. The target may be reacquired in future detections.
1	Active-Moving	The target is in motion. System maintains visual detection of moving target.
2	Dropped	The system does not detect target and determines it is no longer viable. The system will not reuse the target id.
3	Active-Stopped	The target is stationary. System maintains visual detection of stationary target.
4	Active-Coasting	The system is waiting for the target's coasting time to expire. The target may be reacquired in future detections.

The system manages the target's state data throughout the target's lifecycle. The lifecycle starts when a system detects a target within a frame (i.e., Target Created). The method of detection is system dependent, which may use any number of techniques, including, frame differencing, image segmentation, or AI/ML algorithms. When the system creates a target, the system assigns the target a unique identifier and sets the state to "Active-Moving" or "Active-Stopped". For systems initiating a detection based on target movement, the first state to use is "Active-Moving". For systems performing detection without movement, the system sets the detection's first state to "Active-Stopped". The system only uses the unique identifier for this target and the system does not reuse the identifier for any other target, even if the target is no longer part of the target list (i.e., dropped from the list).

While the system continues to detect the target in the scene, the target remains in the Active-Moving or Active-Stopped state. In cases where a detection may stop momentarily (e.g., car behind a tree), the detection system set the targets state to Active-Coasting. The system governs the duration, called the coasting time, to maintain the target in the Active-Coasting state. For example, if the coasting time is 10 seconds the system waits 10 seconds before switching the targets state to either Inactive or Dropped, in case the target is reacquired (e.g., car reappears from behind a tree). If a system chooses it can set the coasting time to zero, then the target changes from either the Active-Moving or Active-Stopped state to Inactive or Dropped as soon as the system does not detect the target in a frame.

While tracking a moving target (i.e., Active-Moving), if the target comes to a stop in the scene and the target is still visible and detectable, the state changes to the Active-Stopped state. If the target starts moving it returns to the Active-Moving state. If the system loses any Active-Moving or Active-Stopped target, the system may set its state to Active-Coasting, Inactive, or Dropped. When a system no longer detects a target and the coasting time has elapsed, the system may change the state to Inactive, which means there is no detection, but the system may detect the target again. To prevent states from vacillating frequently (e.g., Inactive to Active-Moving back to Inactive), systems may employ a hysteresis delay for any state change. Systems decide on the hysteresis which delays the state change until the system is certain the state needs to change.

When a system decides to discontinue managing a target, the system changes the state to Dropped and the system removes the target from the target list. This standard assumes when a system drops a target the detection system "forgets" the target. For example, if the system reacquires the same target later, the system does not "remember" it and therefore cannot reuse the same identifier; instead, the system assigns a new target identifier. To avoid this behavior, systems may leave a target in the Inactive state indefinitely.

7.3 Object and Feature Labeling

When a VMTI system detects a target, the system may include one or more AI/ML image classifiers which label the target with one or more object-predictions of the target. Each object-prediction includes an **object-label** (i.e., what the type of the object is), an **object-confidence** value (i.e., the certainty the label is correct), and optional **feature-labels** (e.g., antenna, rifle, etc.) with corresponding **feature-confidence** levels. Object-label and feature-labels are ontology references, see Section 7.3.1.

The object-confidence and feature-confidence values are numbers which range from zero to 100%. An object-confidence value of 100% indicates the object-label perfectly reflects what the object is, while a value of 0% means no certainty the assigned object-label is correct. Similarly, a feature-confidence value of 100% indicates the feature-label perfectly reflects what the feature is, while a value of 0% means no certainty the assigned feature-label is correct.

Figure 5 shows an example where a VMTI system assigns three object-predictions in blue to a blob of moving pixels (i.e., the "Target") in green: (1) a Vehicle with object-confidence of 90%, (2) a Car with object-confidence 70%, and (3) a Motorcycle with object-confidence of 50%. If the blob of pixels includes a vertical object, each of these object-predictions may have custom features (each including a feature-label and feature-confidence level) in yellow; for example, the car may include an Antenna with feature-confidence of 80%, while the motorcycle may include an Antenna with feature-confidence of 45% and Rifle with feature-confidence of 60%.

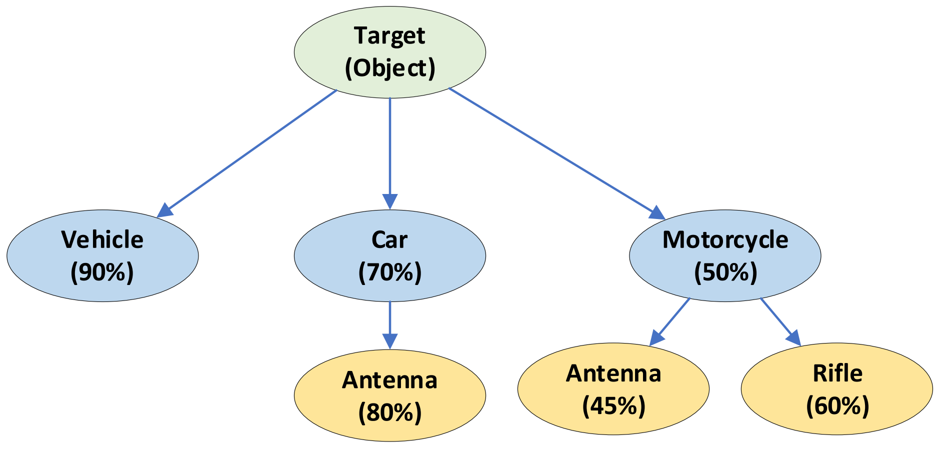


Figure 5: Illustration of Object-Predictions

7.3.1 Ontology References

This standard does not define the target (i.e., object) or feature labels, but the standard does require labels be part of a well-defined ontology using the Web Ontology Language (OWL) [4].

Ontologies provide a consistent means (i.e., common and agreed upon by a body of people and/or organizations) to define terms (e.g., types of objects) and their relationship to other terms, with a coherent set of semantics (e.g., defining terms using other ontology terms). See Appendix A for an exemplar ontology.

There are two parts to an ontology reference: an ontology identifier, and an entity identifier. The ontology identifier is an Internationalized Resource Identifier (IRI) that names the ontology used. The entity identifier is the IRI of the object-label or feature-label defined in the specified ontology. For example, to provide a normative label for "car" using the example ontology from Appendix A:

ontology IRI = http://purl.obolibrary.org/obo/envo.owl
entity IRI = http://purl.obolibrary.org/obo/ENVO_01000605

Optionally, ST 0903 supports adding the label-text from the ontology for the ontology reference:

label = car

VMTI systems may use existing ontologies, i.e., defined in either the commercial, DoD, or NATO organizations, or VMTI systems may build their own ontology by either extending an existing ontology (preferred) or defining a new one. When either extending or defining a new ontology, the resulting ontology must follow the OWL specification [4].

8 Implementation Configurations

Systems may implement object detection and tracking in many different systematic configurations. Multiple source imagery configurations impact the requirements of the VMTI metadata.

8.1 Linkage Considerations

A system configuration where there are N_s imagery sources greater than one, but less than N_s transmitted to a user, may have VMTI "linkage" issues. A linkage issue is where the imagery source for the VMTI processing (i.e., VMTI-MI) and metadata may not be a part of the transmitted imagery to the user (i.e., user-MI). A configuration illustrating a linkage issue is a real-time system with a visual sensor (e.g., RGB) and an IR sensor which transmits only one sensor's Motion Imagery at a time.

Figure 6 illustrates a notional two sensor configuration (Visual and IR) each producing Motion Imagery (blue lines) and metadata (green lines). This standard defines **VMTI-MI** to indicate the Motion Imagery input to the *VMTI Processing*; in this example, the *VMTI Processing* operates on the Visual sensor. The *A/B Switch* determines which sensor's data passes to the *Encoder* for down-stream viewing by users. This standard defines **user-MI** to indicate the Motion Imagery the user will view.

There are two choices for multiplexing the *VMTI Processing* output (red line) with the user-MI: embedded-VMTI (see Section 8.2) or standalone-VMTI (see Section 8.3). Embedded-VMTI (Choice 1) provides bandwidth efficiencies in sending VMTI data to end users. With embedded-VMTI, the VMTI data timing needs to match the sensor's metadata data timing so the VMTI data can properly utilize the parent metadata when computing location information. Implementers should be aware that timing synchronization may introduce latency, and thus, impact real-time delivery. Standalone-VMTI (Choice 2) while less bandwidth efficient reduces potential latency issues for real-time transmittal, although there may be latency between the Motion Imagery and the displayed VMTI information.

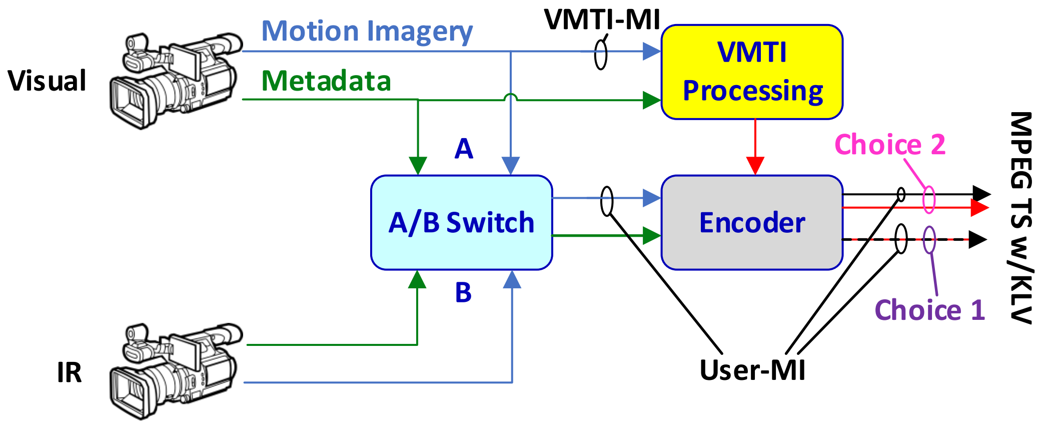


Figure 6: Notional Two Sensor Implementation Configuration

The example illustrates an implementation configuration where the user-MI may be different than the VMTI-MI at any time; therefore, this standard provides requirements to prevent mismatches between VMTI metadata and other system metadata. For example, timing, field of views (FOV), frame sizes, etc. may be different for different sensors metadata.

8.2 Embedded-VMTI

Embedded-VMTI is a VMTI LS which is subordinate to (i.e., a child of) a parent set and relies on metadata items in the parent set. Table 6 lists both Required and Optional parent metadata in embedded-VMTI use cases.

Table 6: Parent Metadata Supporting Items

Metadata Items	Presence
Precision Time Stamp	Required
Frame Center Latitude	Required
Frame Center Longitude	Required
Horizontal Field of View	Optional
Vertical Field of View	Optional
Motion Imagery Identification System (MIIS)	Optional
IR Polarity	Optional

For example, a MISB ST 0601 Datalink LS may include the VMTI LS in the 0601 Item 74. The VMTI LS may then rely on the 0601 LS's Frame Center Latitude/Longitude for specifying VTarget corner offsets. This "embedded" functionality conserves bandwidth by leveraging metadata items in ST 0601. Figure 7 illustrates a single ST 0601 stream of metadata where each Datalink Packet includes the 0601 Item 74. The VMTI Item 74 states offsets to the Datalink Packet's items (blue dots) to complete the VMTI's metadata. See the targetLocationOffsetLat in the VTarget Pack (Section 10.2.2.11) for an example of an offset metadata item in the VMTI LS.

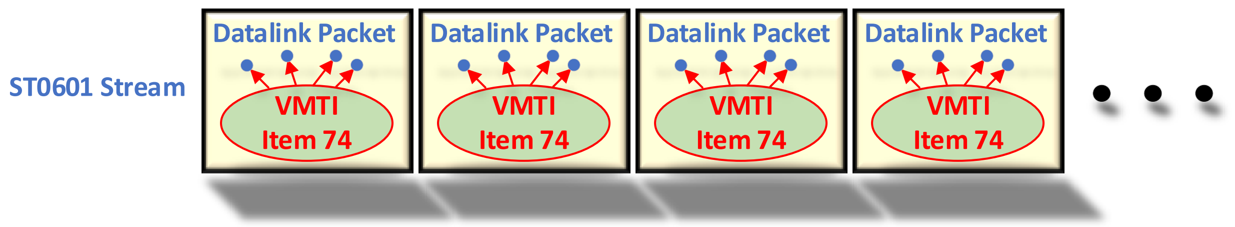


Figure 7: Embedded-VMTI Illustration

The inclusion of some embedded-VMTI metadata items depends on their usage. Table 7 lists conditionally optional LS items based on requirements for the specific item; see each item's full description for the requirements when using embedded-VMTI. Table 8 lists those items which can leverage metadata within a parent Local Set (e.g., MISB ST 0601).

Table 7: Embedded-VMTI conditionally optional items

Local Set	Item #	Item Name
VMTI	2	precisionTimeStamp
VMTI	11	vmtiHorizontalFov
VMTI	12	vmtiVerticalFov
VMTI	13	miisId

Table 8: Embedded-VMTI Items leveraging a parent set

Local Set	Item #	Item Name
VTarget	10	targetLocationOffsetLat
VTarget	11	targetLocationOffsetLon
VTarget	13	boundingBoxTopLeftLatOffset
VTarget	14	boundingBoxTopLeftLonOffset
VTarget	15	boundingBoxBottomRightLatOffset
VTarget	16	boundingBoxBottomRightLonOffset

8.3 Standalone-VMTI

Standalone-VMTI is a self-contained VMTI LS which is not contained within a parent set. In contrast to embedded-VMTI (Section 8.2), standalone-VMTI does not rely on auxiliary metadata (i.e., geospatial) from a parent LS (e.g., the ST 0601 Datalink LS) to support its functionality. An example of standalone-VMTI is transmitting VMTI detections via a separate UDP/IP stream from a Motion Imagery Transport Stream containing the Datalink LS. A second example is encapsulating the VMTI data as a separate private data stream within a MPEG-2 Transport Stream. Figure 8 illustrates two separate streams of metadata where the VMTI Packets do not rely on information from the Datalink Packets Their transmission to end users may use separate communication channels.

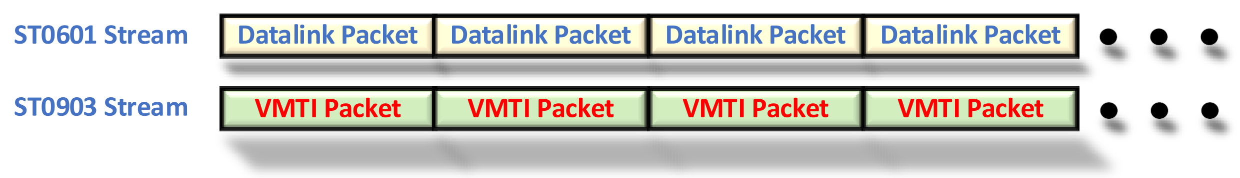


Figure 8: Standalone-VMTI Illustration

Requirement	
ST 0903.6-116	Where using standalone-VMTI, VTarget LS instances shall omit items 10, 11, 13, 14, 15, and 16 (listed in Table 8).

8.4 Timing Considerations

Motion Imagery is a series of Spatio-Temporally Related Images (see [5]), meaning each frame spatially relates to the previous and the time of each frame is known. When performing object detection and tracking the detection results (i.e., VMTI metadata) relate directly to the time of the frame. The combination of implementation configuration, VMTI processing latency, Motion Imagery system latency, and desired output latency/bandwidth, plays a role in the reporting of the VMTI packet time.

For example, referring to the notional implementation in Figure 6, if the VMTI Processing takes one second to complete its object detection and the system operates over a constrained bandwidth channel, the system implementor could choose embedded-VMTI. The encoder then needs to either (1) buffer the Motion Imagery and metadata to match the latency of the VMTI data, or (2) add temporally inconsistent VMTI data to the current Datalink (parent) metadata.

Method (1) offers bandwidth efficiency by leveraging metadata items in the Datalink LS; however, delaying the Motion Imagery may be both impractical and undesirable. Method (2) shares less data with the Datalink LS (hence more bandwidth) but does not affect overall Motion Imagery latency to the end user. Both methods are valid approaches, so careful consideration of the timestamps when embedding the VMTI data is necessary for properly interpreting the results.

A receiving system must know which image the VMTI data associates with; therefore, it is critical for readers to be able to ascertain the correct time of the VMTI data, either by the timestamp existing in the VMTI LS or the parent metadata. MISB standards require every image in a MI stream to contain a Precision Time Stamp. VMTI processing operates on one image of the VMTI-MI which therefore has an image timestamp. When a VMTI system processes an image in VMTI-MI and produces a VMTI LS, the **VMTI-MI-Timestamp** is the VMTI-MI image's timestamp. The VMTI system conditionally includes the VMTI-MI-Timestamp into the VMTI LS.

For standalone-VMTI, this standard requires systems to populate the VMTI LS precisionTimeStamp (Item 2) with its VMTI-MI-Timestamp.

For embedded-VMTI, systems may omit the VMTI LS precisionTimeStamp item if the parent instance's Precision Time Stamp (e.g., ST 0601 Item 2) is equal to the VMTI-MI-Timestamp.

Requirement(s)	
ST 0903.6-117	Where the VMTI LS is standalone-VMTI, the VMTI LS instance shall contain a precisionTimeStamp (Item 2) equal to the VMTI-MI-Timestamp.
ST 0903.6-118	Where the VMTI LS is embedded-VMTI and the parent instance contains a Precision Time Stamp not equal to the VMTI-MI-Timestamp, the VMTI LS instance shall contain the precisionTimeStamp item (Item 2) equal to the VMTI-MI-Timestamp.

For embedded-VMTI which is temporally inconsistent (see method (2) example above), there is an issue when using offset items (i.e., VTarget LS Items 10, 11, 13, 14, 15, and 16 - listed in Table 8). The offset items rely on the parent's Frame Center Latitude/Longitude, and when there is a temporal inconsistency the addition of the offset items and the frame center position will result in position errors. Therefore, when the difference between the VMTI LS precisionTimeStamp and the (current) parent timestamp exceeds the period of two frames, use the absolute metadata items (i.e., VTarget targetLocation (Item 17) and geospatialContourSeries (Item 18)) instead of the offset items.

Requirement	
ST 0903.6-142	Where the VMTI LS is embedded-VMTI and the difference between the parent instance's Precision Time Stamp and the child VMTI LS instance's precisionTimeStamp (Item 2) is greater than the period of two VMTI-MI frames, the VMTI LS instance shall use the VTarget targetLocation (Item 17) and geospatialContourSeries (Item 18) instead of the offset items (i.e., VTarget LS items 10, 11, 13, 14, 15, and 16).

9 KLV Encoding

9.1 Data Types

The Motion Imagery Handbook [5] details the syntax and semantics for KLV items and common complex types[3] found in this standard. MISB ST 0107 [6] provides a set of baseline requirements for using KLV, and defines Tag, Length, and Value encodings for numeric (e.g., floating point) and string KLV items. In this standard many items have a designated fixed length. Some items allow a flexible number of bytes depending on its value specified by the notation "Vmax", where max is the maximum number of bytes allocated. For example, "V4" indicates an encoded value specified with a minimum of one byte to a maximum of four bytes.

Requirement	
ST 0903.4-01	All metadata shall be expressed in accordance with MISB ST 0107.

[3] Common complex types include Local Sets, Variable Length Packs, Defined Length Packs, Truncation Packs, and Floating Length Packs.

9.1.1 IMAP and Special Values

This standard uses the IMAP algorithms of MISB ST 1201 [7] for floating-point values. For specific items, this standard defines special bit patterns to signal out-of-range and User Defined Special Values:

- When an item's measured value exceeds the limits set in this standard the VMTI system sends an out-of-range signal of the measured value. ST 1201 provides two out-of-range signals: IMAP_BELOW_MINIMUM and IMAP_ABOVE_MAXIMUM. If an item's measured value is below this standard's defined range the VMTI system sends the IMAP_BELOW_MINIMUM signal value in place of the measurement. Likewise, if an item's measured value is above this standard's define range the VMTI system sends the IMAP_ABOVE_MAXIMUM in place of the measurement. For example, the VTarget targetHae item defines an IMAP range from -900 to 19,000 meters; if the measured value is 20,000 meters the VMTI system would send the IMAP_ABOVE_MAXIMUM value. See ST 1201 for further details.
- Some measured or computed items in the VMTI structure may result in illegal values. In these cases, this standard defines an "Invalid" signal. For example, a divide by zero or projecting a ray from the image to the ground for a geospatial bounding box corner and the corner does not intersect with the ground, i.e., it lies above the horizon. VMTI systems specify the "Invalid" signal using an IMAP User Defined Special Value of byte 0xC1. For packs requiring fixed length, the software pads the value with zero bytes to accommodate the correct length, e.g., 0xC100 0000, for a pack item requiring 4 bytes.

Requirement	
ST 0903.4-03	Floating point to integer mappings shall comply with MISB ST 1201.

9.1.2 Array Type

This document defines an Array type as a collection of simple data types, such as integers or IMAP floating-point values. Encoded as a variable length pack, each item in an Array is of the same data type and has the same length.

9.1.3 Series Type

This document defines a Series type as a collection of structured data elements, all the same type encoded as a variable length pack. Since all elements are of the same type, element order is not important.

Figure 9 shows the structure of a Series type, where the *Value* is a collection of length-element pairs with the length ("L") specifying the number of bytes of the "Element". The element type is known a priori from the Tag definition (in this document) enabling parsing of the element. The number of elements in the collection is determined by parsing the entire list in the *Value*.

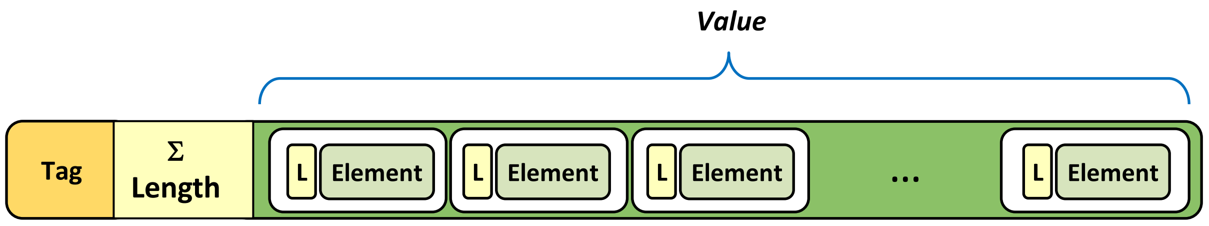


Figure 9: Series Type

9.2 Local Set

9.2.1 *Local Set Uniqueness*

The general definition of a Local Set allows for the repetition of the same item multiple times within a single Local Set. This standard does not allow for repetitive items in all the VMTI LS.

Requirement	
ST 0903.5-98	Within an instance of a Local Set, items within that Local Set shall be used only once.

9.2.2 *Timestamps*

Timestamps specified in this standard utilize the Precision Time Stamp defined in MISB ST 0603 [8] as an eight-byte unsigned integer counter of the number of SI Seconds (in microseconds) which have elapsed since midnight (00:00:00), January 1, 1970 (1970-01-01T00:00:00Z).

Requirement	
ST 0903.4-13	Timestamps specifying a Precision Time Stamp shall comply with MISB ST 0603.

9.2.3 *Target Coordinates*

This standard supports both pixel-based coordinates and geospatial coordinates descriptions for defining items such as the position or location of a target (known as a centroid), and user-defined bounding boxes around a target within an image. The VMTI LS assumes an image with pixel width/height defined by its source sensor.

9.2.3.1 *Pixel Based Representation*

For pixel-based coordinates relative to an image there are two ways to specify a position: 1) as a pixel number computed using row-major addressing (useful in bandwidth sensitive cases), or 2) as a two-dimensional row/column pixel-pair.

Row-major addressing is the count of the pixel number starting at the top left pixel then proceeding from left to right and from top to bottom. Computing the row-major address of a pixel number from the row and column position requires knowing the Motion Imagery frame width, i.e., the number of elements of each row. Both the row and column numbering start with 1, at the top left pixel. The row-major address, or pixel number, given the pixel's row and column is:

$$\text{pixel number} = \text{Column} + ((\text{Row} - 1) \times \text{frame width}))$$

The top left pixel of a frame equates to (Column, Row) = (1, 1) and pixel number 1. Figure 10 illustrates the row-major addressing in a 3-row by 4-column image. Each pixel position in the image contains the row/column address in parenthesis along with the address, or pixel number, in red. Notice the red values increase first across the columns then onto the next row.

	Column 1	Column 2	Column 3	Column 4
Row 1	(1,1) 1	(1,2) 2	(1,3) 3	(1,4) 4
Row 2	(2,1) 5	(2,2) 6	(2,3) 7	(2,4) 8
Row 3	(3,1) 9	(3,2) 10	(3,3) 11	(3,4) 12

Figure 10: Row-major Image addressing

The computation of the pixel number for the yellow pixel at position (2,3) uses a frame width of 4 in the above equation:

$$\text{pixel number} = 3 + \left((2 - 1) \times 4\right) = 7$$

The following two equations convert a pixel number to corresponding row/column coordinates:

$$\text{Row} = \left\lfloor \frac{\text{pixel number}}{\text{frame width}} \right\rfloor + 1$$

$$\text{Column} = (\text{pixel number} - (\text{Row} - 1) \text{ x frame width})$$

Where the $\lfloor \quad \rfloor$ indicates the floor function, e.g., $\lfloor 3.99 \rfloor = 3$ and $\lfloor 3.0 \rfloor = 3$.

For example, using the image in Figure 10, the row and column addresses for the yellow pixel with pixel number 7 is:

$$\text{Row} = \left\lfloor \frac{7}{4} \right\rfloor + 1 = \lfloor 1.75 \rfloor + 1 = 1 + 1 = 2$$

$$\text{Column} = (7 - (2 - 1) * 4) = 3$$

9.2.3.2 Geospatial Based Representation

In this standard, geospatial values have two representations, either full coordinates, or as offset coordinates. The Local Set item description will indicate which method to use. Full coordinates specify the latitude, longitude, and height above ellipsoid with their full ranges, e.g., -90 to +90 for latitude. Offset coordinates leverage values in parent metadata (e.g., a MISB ST 0601 LS parent) for representing geospatial values, thereby saving bandwidth.

When computing geospatial coordinates, it is possible to have situations where the computational results do not intersect the earth surface and therefore are non-existent. For example, projecting a ray from the image to the ground for a geospatial bounding box corner and the corner does not intersect with the ground, i.e., it lies above the horizon. In these cases, either do not report the value or report the value using the "Invalid" value, see Section 9.1.1.

9.2.4 Local Set Table Structure

Local Set tables in this standard follow a common structure with column designations as follows:

- *Tag* indicates the Tag specifying the metadata item
- *Name* is the label associated with the Tag
- *Units* specifies the Value's units
    - Units of "N/A" indicates no units apply
    - All other Units are SI enumerations (e.g., µs is micro-seconds, ° is degrees)
- *Format* indicates the item's KLV format for the Value
- *Len* indicates the nominal length of an item in bytes. See Section 9.1 for nomenclature of variable length assignments.
- *Description* provides a brief description of the metadata item

10 VMTI LS Structure

The VMTI LS encompasses the core information applicable to all reported phenomena within a Motion Imagery frame. Figure 11 is a UML model of the VMTI data construct. The following lists the nomenclature and provides notes regarding the VMTI model:

- Orange color denotes Local Sets; blue color indicates Pack structures; green color is a mix of Pack/Local Sets.
- The class name reflects the collective meaning of the attributes within the class. Class attributes are the informational items defining the class content. The attribute trait name is a single word or phrase starting with a lowercase letter and in lower camel case. The attribute trait type defines the data representation of the attribute value once the class has been instantiated.
    - - <attribute> is for Pack items e.g., -latitude in the Location Pack.
    - - ##_<attribute> lists Local Set items e.g., -02_ontology in the Ontology LS.
    - Multiplicities denote Series Type of Local Sets or Packs.
    - Arrays ("[ ]") denote Series Type of simple types such as integers.

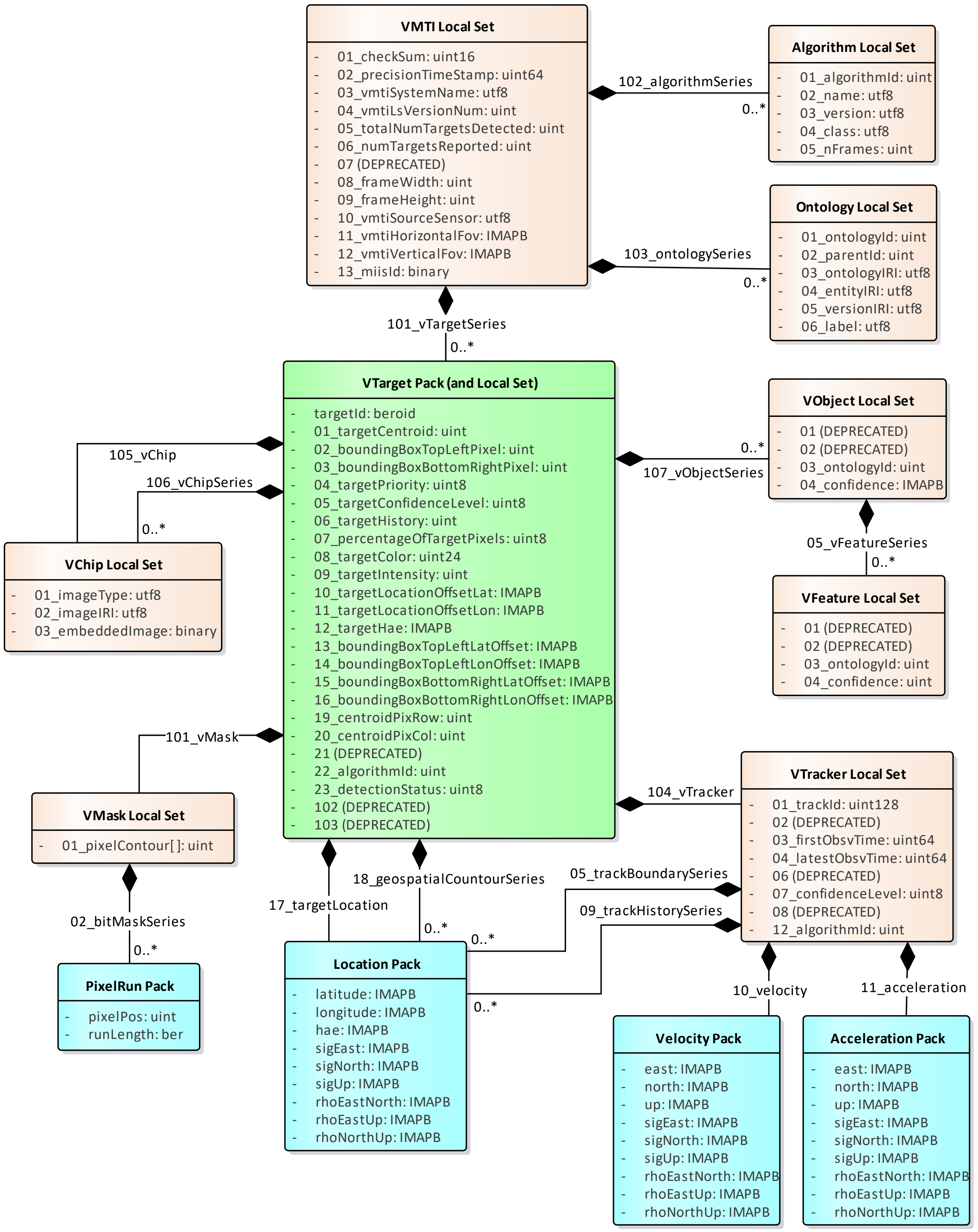


Figure 11: VMTI LS UML Model

The VMTI LS supports information regarding identification and labeling of multiple detections within a Motion Imagery frame. When the VMTI LS is using embedded-VMTI, for example, within MISB ST 0601 (i.e., ST 0601 Item 74), the VMTI LS has a choice to express the frame center geographic coordinate as offsets from the frame center geographic coordinate provided in the parent versus including absolute coordinates. However, when used as a standalone-VMTI, the VMTI LS expresses these items in absolute geographic coordinates.

For example, Figure 12 illustrates the VMTI LS structure referenced subordinate to MISB ST 0601 as Item 74. The BER Length is the number of bytes the *Value* uses. The Value is composed of TLV items from the VMTI LS, and in this example, includes a vTargetSeries variable length pack (VLP) (Item 101 of the VMTI LS).

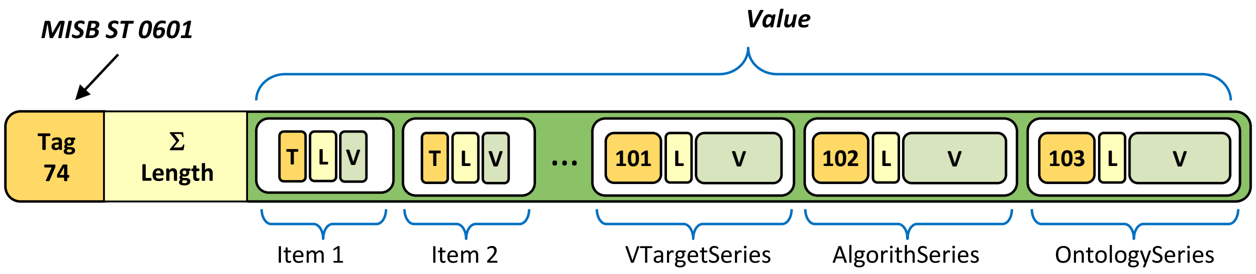


Figure 12: *Value* = Data within the VMTI Structure

Figure 13 illustrates an example VMTI LS subordinate to a MISB ST 0601 LS.

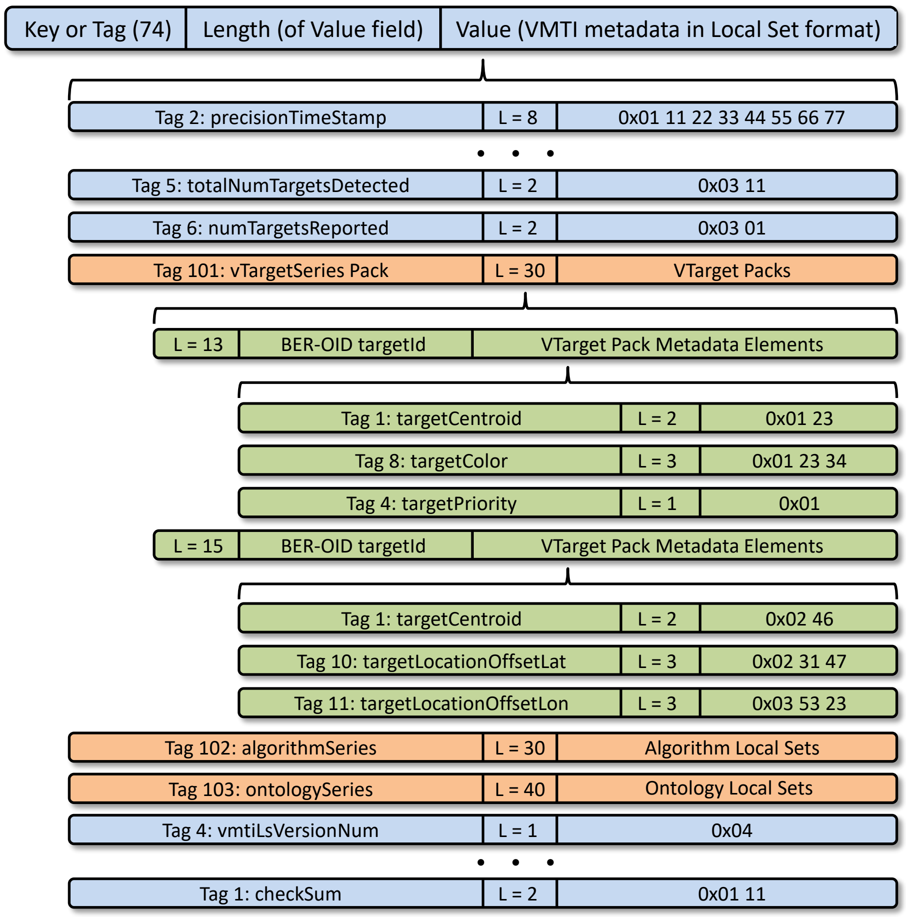

Key or Tag (74)
Length (of Value field)
Value (VMTI metadata in Local Set format)
Tag 2: precisionTimeStamp | L = 8 | 0x01 11 22 33 44 55 66 77
• • •
Tag 5: totalNumTargetsDetected | L = 2 | 0x03 11
Tag 6: numTargetsReported | L = 2 | 0x03 01
Tag 101: vTargetSeries Pack | L = 30 | VTarget Packs
L = 13 | BER-OID targetId | VTarget Pack Metadata Elements
Tag 1: targetCentroid | L = 2 | 0x01 23
Tag 8: targetColor | L = 3 | 0x01 23 34
Tag 4: targetPriority | L = 1 | 0x01
L = 15 | BER-OID targetId | VTarget Pack Metadata Elements
Tag 1: targetCentroid | L = 2 | 0x02 46
Tag 10: targetLocationOffsetLat | L = 3 | 0x02 31 47
Tag 11: targetLocationOffsetLon | L = 3 | 0x03 53 23
Tag 102: algorithmSeries | L = 30 | Algorithm Local Sets
Tag 103: ontologySeries | L = 40 | Ontology Local Sets
Tag 4: vmtiLsVersionNum | L = 1 | 0x04
• • •
Tag 1: checkSum | L = 2 | 0x01 11

Figure 13: VMTI LS Example (subset of available data)

The LS begins with a ST 0601 Tag 74 signaling the presence of a VMTI LS (in blue). Following Tag 74 is a Length value, which is the sum of the lengths of all data in the VMTI LS. In this example, the VMTI LS contains Item 5 (which specifies the totalNumTargetsDetected in the Motion Imagery frame), and Item 101 indicates a vTargetSeries Pack, which contains two VTarget Packs (indicated in green). The Value portion of the first VTarget Pack includes a BER-OID encoded targetId, targetCentroid (Item 1), targetColor (Item 8), and targetPriority (Item 4).

The Value portion of the second VTarget Pack is similar, except it has targetCentroid (Item 1), targetLocationOffsetLat (Item 10), and targetLocationOffsetLon (Item 11). The example assumes each targetId is one byte. Following the vTargetSeries are the algorithmSeries, ontologySeries, vmtiLsVersionNum and checkSum items.

10.1 VMTI LS

The VMTI LS 16-Byte Universal Label (UL) "Key" registered in MISB ST 0807 [9] is:

06.0E.2B.34.02.0B.01.01.0E.01.03.03.06.00.00.00 (CRC 51307)

Table 9 summarizes the VMTI LS. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 9: VMTI LS

VMTI Local Set					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
1	checkSum	None	uint	2	Detects errors within a standalone VMTI LS
2	precisionTimeStamp	µs	uint	8	Microsecond count from Epoch of 1970; See MISP Time System - MISB ST 0603
3	vmtiSystemName	N/A	utf8	V32	Name and/or description of the VMTI system
4	vmtiLsVersionNum	N/A	uint	V2	Version number of the VMTI LS used to generate the VMTI metadata
5	totalNumTargetsDetected	N/A	uint	V3	Total number of targets in VMTI system's processing model's target list
6	numTargetsReported	N/A	uint	V3	Number of targets reported following a culling process
7	Item DEPRECATED	---	---	---	ST 0903.6 deprecates this item
8	frameWidth	pixels	uint	V3	Width of the Motion Imagery frame in pixels
9	frameHeight	pixels	unit	V3	Height of the Motion Imagery frame in pixels
10	vmtiSourceSensor	N/A	utf8	V128	VMTI source sensor (as string). E.g., 'EO Nose', 'EO Zoom (DLTV)'
11	vmtiHorizontalFov	°	IMAPB	2	Horizontal field of view of imaging sensor input to VMTI process
12	vmtiVerticalFov	°	IMAPB	2	Vertical field of view of imaging sensor input to VMTI process

VMTI Local Set					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
13	miisId	N/A	binary	V	A Motion Imagery Identification System (MIIS) Core Identifier conformant with MISB ST 1204 [10]
101	vTargetSeries	N/A	Series	V	VTarget Packs ordered as a Series
102	algorithmSeries	N/A	Series	V	Series of one or more Algorithm LS
103	ontologySeries	N/A	Series	V	Series of one or more Ontology LS

10.1.1 VMTI LS: Item 1 – checkSum

The *checkSum* item aids detecting errors in delivery with standalone-VMTI. Refer to MISB ST 0601 for the checksum algorithm. Performed over the entire LS, the checksum includes the 16-byte UL key and 1-byte checksum length. The Value represents the lower 16-bits of summation.

Example Value	**Example Encoded LS Value**		
0	Tag	Length	Value
	0x01	0x02	0x0000

Requirement(s)	
ST 0903.4-15	Where the VMTI LS is standalone-VMTI, the VMTI LS instance shall contain a checkSum (Item 1).
ST 0903.6-119	Where the VMTI LS is standalone-VMTI, the checkSum shall be a running 16-bit unsigned summation of all bytes in the Local Set (see MISB ST 0601), starting with the first byte of the VMTI LS's UL, up to and including the last byte of the Length of the checksum itself.
ST 0903.4-17	Where the VMTI LS is standalone-VMTI, the VMTI LS instance shall have the checkSum (Item 1) be the last item in the instance.
ST 0903.6-120	Where the VMTI LS is embedded-VMTI, the VMTI LS checkSum (Item 1) shall be omitted.

10.1.2 VMTI LS: Item 2 – precisionTimeStamp

Defined in MISB ST 0603, the Precision Time Stamp is the number of microseconds elapsed since the MISP Time System epoch of midnight (00:00:00), January 1, 1970, and the microsecond count does NOT include leap seconds. The VMTI LS *precisionTimeStamp* (Item 2) is equal to VMTI-MI-Timestamp.

Example Value	**Example Encoded LS Value**		
April 19, 2001, 04:25:21.000000 GMT (987654321000000)	Tag	Length	Value
	0x02	0x08	0x0003 8244 30F6 CE40

When including the precisionTimeStamp item, it must be the first item of the VMTI LS.

Requirement	
ST 0903.4-14	Where a precisionTimeStamp (Item 2) is present in a VMTI LS, it shall be the first item in the VMTI LS.

10.1.3 VMTI LS: Item 3 – vmtiSystemName

The vmtiSystemName item is the name or description of the VMTI system producing the VMTI targets identified as a string of 32 UTF-8 characters. Note that UTF-8 allows up to four bytes per character; thus, this value can expand up to 128 bytes maximum. The vmtiSystemName is free text.

Example Value	**Example Encoded LS Value**		
	Tag	Length	Value
DSTO_ADSS_VMTI	0x03	0x0E	0x4453 544F 5F41 4453 535F 564D 5449

10.1.4 VMTI LS: Item 4 – vmtiLsVersionNum

The *vmtiLsVersionNum* is the version number of the VMTI LS document used to generate the VMTI metadata and notifies downstream clients of the LS version used to encode the VMTI metadata. Values of 1 through 65535 correspond to document revisions 1 through 65535.

Example Value	**Example Encoded LS Value**		
	Tag	Length	Value
6 (i.e., 0903.6)	0x04	0x01	0x06

Requirement	
ST 0903.5-99	All instances of the VMTI LS shall contain vmtiLsVersionNum (Item 4).

10.1.5 VMTI LS: Item 5 – totalNumTargetsDetected

The *totalNumTargetsDetected* item is the total number of targets in the VMTI processing model's target list; this value may be different than the number of elements in the vTargetSeries. To save bandwidth, the VMTI system may only report a subset of the VMTI processing model's target list. Section 6 describes the different scenarios for generating and reporting target lists. A value of zero represents no targets detected in the VMTI processing model's list.

Example Value	**Example Encoded LS Value**		
	Tag	Length	Value
28	0x05	0x01	0x1C

Requirement	
ST 0903.4-18	Where the VMTI LS totalNumTargetsDetected in a frame (Item 5) is different from the VMTI LS numTargetsReported (Item 6), the totalNumTargetsDetected in the frame shall be specified.

10.1.6 VMTI LS: Item 6 – numTargetsReported

The *numTargetsReported* item is the count of a subset of the target list. Reporting only a subset of the target list improves bandwidth efficiency.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
14	0x06	0x01	0x0E

Requirement	
ST 0903.4-19	The VMTI LS numTargetsReported (Item 6) shall always be specified.

10.1.7 VMTI LS: Item 7 (DEPRECATED)

ST 0903.6 deprecates this item; refer to versions prior to ST 0903.6 for information on its use.

10.1.8 VMTI LS: Item 8 – frameWidth

The *frameWidth* item specifies the width of the VMTI-MI frame in pixels, which corresponds to the number of pixels in a row of the image where pixels appear in row-major order. Do not use a value of zero.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
1920	0x08	0x02	0x0780

Requirement	
ST 0903.6-121	Where a VMTI LS instance includes metadata items using the pixel number representation, the VMTI LS instance shall contain the frameWidth item (Item 8) set to the frame width of the VMTI-MI.

10.1.9 VMTI LS: Item 9 – frameHeight

The *frameHeight* item specifies the height of the VMTI-MI frame in pixels, which corresponds to the number of rows of pixels in the image where pixels appear in row-major order. The frameHeight is not a required value. Do not use a value of zero.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
1080	0x09	0x02	0x0438

10.1.10 VMTI LS: Item 10 – vmtiSourceSensor

The *vmtiSourceSensor* item is a free text identifier for the source of the VMTI-MI, e.g., 'EO Nose', 'EO Zoom (DLTV)', 'EO Spotter', 'IR Mitsubishi PtSi Model 500', 'IR InSb Amber Model TBT', 'LYNX SAR Imagery', 'TESAR Imagery', etc. The vmtiSourceSensor identifies the source for systems where there are multiple sensors. Any change to the VMTI-MI requires updating this metadata item. The value is a free text string of 128 UTF-8 characters. UTF-8 allows up to four bytes per character, so this value can expand up to 512 bytes maximum.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
EO Nose	0x0A	0x07	0x454F 204E 6F73 65

10.1.11 VMTI LS: Item 11 – vmtiHorizontalFov

The *vmtiHorizontalFov* item is the VMTI sensor horizontal field of view (HFOV) of the source input. ST 0903 requires Item 11 in two cases: 1) standalone-VMTI, or 2) embedded-VMTI and the VMTI-MI is different from the user-MI. Otherwise, the parent (e.g., ST 0601 LS Item 16) provides the HFOV value.

Valid Values: The set of real numbers from 0 to 180 inclusive.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
12.5 degrees	0x0B	0x02	12.5 → IMAPB(0, 180, 2) = 0x0640

Requirement(s)	
ST 0903.4-26	Where the user-MI is different from the VMTI-MI, the VMTI LS shall contain the vmtiHorizontalFov (Item11).
ST 0903.6-122	Where the VMTI LS is standalone-VMTI, the VMTI LS shall contain the vmtiHorizontalFov (Item 11).

10.1.12 VMTI LS: Item 12 – vmtiVerticalFov

The *vmtiVerticalFov* item is the vertical field of view (VFOV) of the source input. This is a required item in two cases: 1) standalone-VMTI, or 2) embedded-VMTI and the VMTI-MI is different from the user-MI. Otherwise, the parent (e.g., ST 0601 LS Item 17) provides the VFOV value.

Valid Values: The set of real numbers from 0 to 180 inclusive.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
10.0 degrees	0x0C	0x02	10.0 → IMAPB(0, 180, 2) = 0x0500

Requirement(s)	
ST 0903.4-27	Where the user-MI is different from the VMTI-MI, the VMTI LS instance shall contain the vmtiVerticalFov (Item12).
ST 0903.6-123	Where the VMTI LS is standalone-VMTI, the VMTI LS instance shall contain the vmtiVerticalFov (Item12).

10.1.13 VMTI LS: Item 13 – miisId

The *miisId* item provides a unique Motion Imagery Identification System (MIIS) identifier for the VMTI-MI from which the system derives the target list. When using embedded-VMTI and the VMTI-MI is the same as the user-MI, the VMTI system may omit this item in the VMTI LS instance.

Valid Values: A value conformant with MISB ST 1204.

Requirement(s)	
ST 0903.6-124	Where the VMTI-MI is different than the user-MI, the VMTI LS shall contain the VMTI-MI's MIIS in the miisId (Item 13).
ST 0903.6-125	Where the VMTI LS is standalone-VMTI, the VMTI LS shall contain the VMTI-MI's MIIS in the miisId (Item 13).

10.1.14 VMTI LS: Item 101 – vTargetSeries

The *vTargetSeries* item is a Series type which contains VTarget Packs only (see Figure 14). The Series Length is the number of bytes the v*TargetSeries* Value uses. The Value is comprised of one or more VTarget Packs, each of which can be of a different size according to its VTarget Pack Length (L). Each VTarget Pack in the vTargetSeries is for a unique target, i.e., a VTargetSeries cannot contain two VTarget Packs both using the same VTarget Pack targetId.

The example in Figure 14 shows the VMTI LS's first data is a TLV Item 5|1|2 indicating Item 5 "Total Number of Targets Detected in the Frame", a Length of 1 byte, and a Value of 2 indicating two VTarget Packs. Tag 101 indicates a vTargetSeries with the Series Length indicated by ΣL, i.e., the sum of the sizes of each VTargetPack in the Series. The size of each VTargetPack is the number of bytes for the VTarget Pack Length and the number of bytes for the VTarget Pack Value. Two VTarget Packs within the vTargetSeries follow; the VTarget Pack length (L) in each VTarget Pack specifies the respective pack's length for parsing. The number circled in red is the targetId of the VTarget Pack, which must be unique within the vTargetSeries. The remaining information in each VTarget Pack are specifics about the target such as centroid pixel number and target location latitude and longitude.

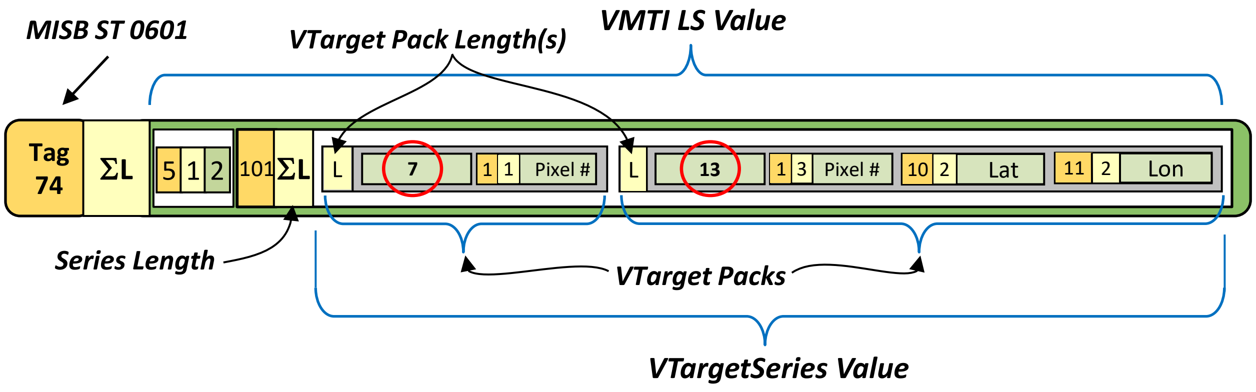


Figure 14: VMTI LS vTargetSeries Example

Requirement	
ST 0903.6-126	Each VTarget Pack element in a VMTI LS vTargetSeries (Item 101) shall have a unique targetId.

Example Value	**Example Encoded LS Value**		
	Tag	Length	Value (Length, VTarget Pack Data)
Three VTarget Packs with Target IDs of 1, 2 and 3	0x65	[Len]	[Len Target 1] 0x01… [Len Target 2] 0x02… [Len Target 3] 0x03…
		[Len] is the BER length of the whole Value [Len Target #] is the BER length of the succeeding VTarget	

In the above example, [Len] is the cumulative length of all VTarget Packs in the VTargetSeries; [Len Target 1], [Length Target 2], and [Length Target 3] are the respective lengths of each VTarget Pack. BER-OID Target IDs (i.e., 0x01, 0x02, 0x03) and the remaining VTarget Pack data follows each length (i.e., [Length Target n] where n = 1, 2, 3) value.

10.1.15 VMTI LS: Item 102 – algorithmSeries

The *algorithmSeries* item is a Series type which contains one or more Algorithm Local Sets. The length for the series is the number of bytes the algorithmSeries value uses. The value is comprised of one or more Algorithm LS, each of which can be of a different size (thereby including different information) parsed according to the length provided for each LS.

The algorithmSeries enables assigning an algorithm LS to a detected target or a generated track, see VTarget Item 22 and VTracker Item 12.

10.1.16 VMTI LS: Item 103 – ontologySeries

The *ontologySeries* item is a Series type which contains one or more Ontology Local Sets. The length for the series is the number of bytes the ontologySeries value uses. The value is comprised of one or more Ontology LS, each of which can be of a different size (thereby including different information) parsed according to the length provided for each LS.

The ontologySeries enables assigning multiple ontologies to each VTarget (see VTarget Item 107) and multiple features to each VObject (see VObject Item 5).

10.2 VTarget Pack

The VTarget Pack contains two items: a BER-OID encoded Value (*Item 1*) which is a targetId, and one or more VTarget TLV items (collectively as *Item 2*) as shown in Figure 15.

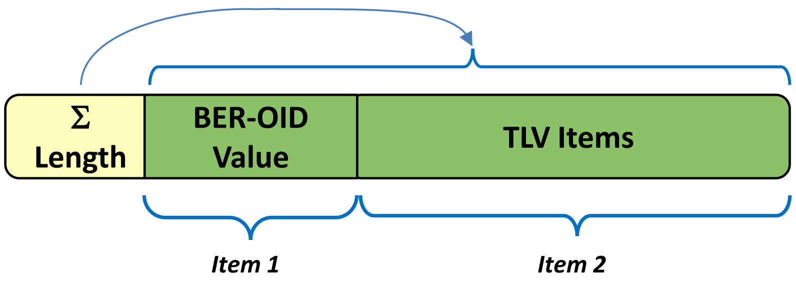


Figure 15: Structure of a VTarget Pack

The VTarget Pack has a defined order. Item 1 may have variable size and Item 2 may have a different number of TLV items. The "∑ Length" of a VTarget Pack is the sum of the length of Item 1 plus the length of Item 2, which can vary at runtime.

Figure 16 shows an example VTarget Pack with *Item 1*, the targetId (BER-OID encoded), and Item 2, a list of VTarget TLV items, each with a Tag (T) and a Length (L). Since each VTarget Pack may be different and will not have the same information, each may be of a different size.

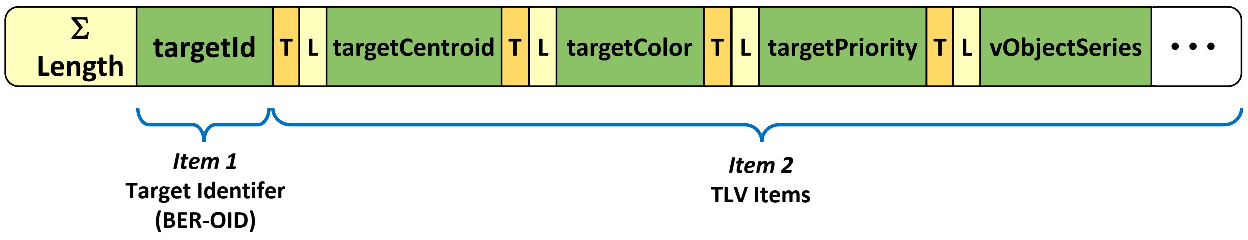


Figure 16: VTarget Pack Example

The targetId in a VTarget Pack is mandatory; while other TLV items are optional, at least one needs to be present. This construct saves bandwidth. For example, in the simplest case, a VTarget Pack might consist of only a targetId and a targetCentroid.

Requirement	
ST 0903.4-10	Where using a VTarget Pack, at least one Tag-Length-Value item shall follow the targetId.

Table 10 summarizes the VTarget Pack. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 10: VTarget Pack (and Local Sets)

VTarget Pack					
Tag	Name	Units	Format	Len	Description
N/A	targetId	N/A	BER-OID	V9	Mandatory BER-OID encoded target id and first value in a VTarget Pack

VTarget Pack					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
1	targetCentroid	Pixel Number	uint	V6	Defines the position of the target within the Motion Imagery frame as a pixel number
2	boundingBoxTopLeft	Pixel Number	uint	V6	Position of the top left corner of the target's bounding box within the Motion Imagery frame as a pixel number
3	boundingBoxBottomRight	Pixel Number	uint	V6	Position of the bottom right corner of the target's bounding box within the Motion Imagery frame as a pixel number
4	targetPriority	N/A	uint	1	Priority or validity of target based on criteria within the VMTI system
5	targetConfidenceLevel	N/A	uint	1	Confidence level of target based on criteria within the VMTI system
6	targetHistory	N/A	uint	V2	Number of previous times the same target detected
7	percentageOfTargetPixels	N/A	uint	1	Ratio of the target's pixels to the number of pixels in the target's pixel bounding box (multiplied by 100)
8	targetColor	N/A	uint	3	Dominant color of the target
9	targetIntensity	N/A	uint	V3	Dominant intensity of the target
10	targetLocationOffsetLat	°	IMAPB	3	Latitude offset for target from frame center latitude (used with embedded-VMTI)
11	targetLocationOffsetLon	°	IMAPB	3	Longitude offset for target from frame center longitude (used with embedded-VMTI)
12	targetHae	m	IMAPB	2	Height of target in meters above WGS84 Ellipsoid
13	boundingBoxTopLeftLatOffset	°	IMAPB	3	Latitude offset for top left corner of target's geospatial bounding box
14	boundingBoxTopLeftLonOffset	°	IMAPB	3	Longitude offset for top left corner of target's geospatial bounding box
15	boundingBoxBottomRightLatOffset	°	IMAPB	3	Latitude offset for bottom right corner of target's geospatial bounding box
16	boundingBoxBottomRightLonOffset	°	IMAPB	3	Longitude offset for bottom right corner of target's geospatial bounding box

VTarget Pack					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
17	targetLocation	N/A	Location	V	Location of the target (latitude, longitude, & height above WGS84 Ellipsoid), with sigma and rho values
18	geospatialContourSeries	N/A	Series	V	Geospatial boundary encompassing the target
19	centroidPixRow	N/A	uint	V4	Specifies the row in pixels of the target centroid within the Motion Imagery frame
20	centroidPixCol	N/A	uint	V4	Specifies the column in pixels of the target centroid within the Motion Imagery frame
21	Item DEPRECATED	---	---	---	ST 0903.6 deprecates this item
22	algorithmId	N/A	uint	V3	Identifier indicating which algorithm in Algorithm Series detected this target
23	detectionStatus	N/A	uint	1	Enumeration indicating the current state of VMTI detections for a given entity (Inactive, Active-Moving, Dropped, Active-Stopped, Active-Coasting)
101	vMask	N/A	Local Set	V	Local Set to include a mask for delineating the perimeter of the target
102	Item DEPRECATED	---	---	---	ST 0903.6 deprecates this item
103	Item DEPRECATED	---	---	---	ST 0903.6 deprecates this item
104	vTracker	N/A	Local Set	V	Local Set to include track information about the target
105	vChip	N/A	Local Set	V	Local Set to include underlying pixel values for the target
106	vChipSeries	N/A	Series	V	Series of one or more VChip LS
107	vObjectSeries	N/A	Series	V	Series of one or more VObject LS

10.2.1 Centroid Representation

This standard supports multiple methods to describe the centroid of a target. A target in an Active-Moving or Active-Stopped state should have a computable centroid.

Requirement(s)	
ST 0903.6-127	Where the VTarget LS's instance includes detectionStatus (Item 23) set to Active-Moving, the instance shall contain at least one of the following representations: VTarget Pack targetCentroid (Item 1); VTarget Pack centroidPixRow (Item 19) and centroidPixCol (Item 20); or VTarget Pack boundingBoxTopLeft (Item 2) and boundingBoxBottomRight (Item 3).
ST 0903.6-128	Where the VTarget LS's instance includes detectionStatus (Item 23) set to Active-Stopped, the VTarget LS shall contain at least one of the following representations: VTarget Pack targetCentroid (Item 1); or VTarget Pack centroidPixRow (Item 19) and centroidPixCol (Item 20); or VTarget Pack boundingBoxTopLeft (Item 2) and boundingBoxBottomRight (Item 3).

10.2.2 VTarget Items

10.2.2.1 VTarget Pack: targetId

The *targetId* uniquely identifies a detected target and does not change during the lifecycle of the target, see Section 7.1.1 for background on the VMTI processing model. The targetId is BER-OID encoded to convey the length and has no Tag.

Example Value	Example Encoded Value		
	Tag	Length	Value (BER-OID)
1234	N/A	N/A	0x8952

Requirement	
ST 0903.6-129	When a VMTI system creates a target, the targetId shall be unique and remain the same throughout the lifecycle of the target.

10.2.2.2 VTarget Pack: Item 1 – targetCentroid

The *targetCentroid* item specifies the position of a target centroid within a frame as a pixel number. See Section 9.2.3.1 for pixel number computation. See Section 10.2.1 for usage requirements.

Range: All integer values from 1 to 0xFFFFFFFFFFFF (281,474,976,710,655).

Example Value	Example Encoded LS Value		
	Tag	Length	Value
409,600	0x01	0x03	0x0640 00

Equations for calculating the coordinates of the centroid (geometric center) (x_c, y_c) of a non-intersecting closed polygon with n vertices:

$$x_c = \frac{1}{6A}\sum_{i=0}^{n-1}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i), \quad y_c = \frac{1}{6A}\sum_{i=0}^{n-1}(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

where $$A = \frac{1}{2}\sum_{i=0}^{n-1}\left(x_i y_{i+1} - x_{i+1} y_i\right)$$

As a practical matter calculation of the exact centroid is probably not necessary. The centroid of a pixel bounding box might be adequate.

10.2.2.3 VTarget Pack: Item 2 – boundingBoxTopLeft

VTarget Pack Items 2 and 3 define a target's pixel bounding box with two numbers. The *boundingBoxTopLeft* item is the position of the top left corner of a target's pixel bounding box using the pixel number representation; see Section 9.2.3.1 for pixel number computation.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
409,600	0x02	0x03	0x0640 00

10.2.2.4 VTarget Pack: Item 3 – boundingBoxBottomRight

The *boundingBoxBottomRight* item specifies the position of the bottom right corner of the target's pixel bounding box within the frame using the pixel number representation; see Section 9.2.3.1 for pixel number computation.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
409,600	0x03	0x03	0x0640 00

10.2.2.5 VTarget Pack: Item 4 – targetPriority

The *targetPriority* item provides systems downstream a means to intelligently cull targets for a given frame as VMTI processors may generate thousands of hits.

Valid Values: 1 to 255, where 1 is the highest priority.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
27	0x04	0x01	0x1B

10.2.2.6 VTarget Pack: Item 5 – targetConfidenceLevel

The *targetConfidenceLevel* item expresses the confidence level as a percentage based on criteria within the VMTI system. Target(s) with the highest confidence may not have the highest priority value. The potential is to send the highest confidence targets in limited bandwidth scenarios. Multiple targets may have the same confidence level. The range is 0 to 100, where 100 percent is the highest confidence. A confidence level of 0 percent indicates no confidence that a detection is a potential target. A target detected with a high confidence may be a low priority target.

Valid Values: The set of integer values from 0 to 100 inclusive.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
80%	0x05	0x01	0x50

10.2.2.7 VTarget Pack: Item 6 – targetHistory

The *targetHistory* is the number of times (i.e., frames) the system detects the same target with the same targetId. The targetHistory can indicate target persistence i.e., the number of previous detections of the same target and may provide useful context when a target reappears after no detection for a significant time. There is no requirement that detections be in consecutive frames.

Valid Values: 0 to 65535 frames, where a value of 0 denotes the target as a new detection.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
2765	0x06	0x02	0x0ACD

10.2.2.8 VTarget Pack: Item 7 – percentageOfTargetPixels

The *percentageOfTargetPixels* item specifies the ratio of the target pixels to the size of the bounding box, multiplied by 100. The range is 1 to 100, where 100 signifies the target completely fills the bounding box.

Valid Values: The set of integer numbers from 1 to 100 inclusive.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
50%	0x07	0x01	0x32

10.2.2.9 VTarget Pack: Item 8 – targetColor

The *targetColor* item is the dominant color of the target expressed using RGB color values, with general mapping of any multispectral dataset to an RGB value. VMTI systems may compute the dominant color by any desired method, for example averaging all the pixels, by bands, in the bounding box. The targetColor's primary use is when transmitting metadata in the absence of the underlying Motion Imagery. Represent the RGB color value as: first byte = Red, second byte = Green, third byte = Blue.

Valid Values: All integer values from 0 to 0xFF (255) for each of three one-byte values.

Example Value	Example Encoded LS Value		
(Red, Green, Blue) =	Tag	Length	Value
(218, 165, 32) (i.e., "Goldenrod")	0x08	0x03	0xDAA5 20

10.2.2.10 VTarget Pack: Item 9 – targetIntensity

The *targetIntensity* item is the dominant intensity of the target with dynamic range up to 24 bits. The targetIntensity provides a relative measure of how the different targets compare with each other. The intensity value comes directly from the source imagery and knowledge of the specific bit-range or status (e.g., gain adjusted) is unknown. VMTI systems may compute the dominant intensity of a target by any desired method, for example using the maximum intensity in the target bounding box or averaging all the intensities in the bounding box.

The primary use of the targetIntensity is for infrared (IR) systems; for non-IR systems use the targetColor item (Item 8). The intensity value meaning (i.e., White-Hot, or Black-Hot) is consistent with IR Polarity specified in the parent set (e.g., MISB ST 0601), if present; if the IR Polarity is unknown, assume White Hot.

Primarily, for use when transmitting metadata in the absence of the underlying Motion Imagery.

Example Value	**Example Encoded LS Value**		
	Tag	Length	Value
13140	0x09	0x02	0x3354

10.2.2.11 VTarget Pack: Item 10 – targetLocationOffsetLat

The *targetLocationOffsetLat* item is the latitude offset for the target from the parent's Frame Center Latitude (e.g., MISB ST 0601 Item 23) based on the WGS84 ellipsoid. This item has meaning only when embedding the VMTI LS in ST 0601 LS. The targetLocationOffsetLat adds to the Frame Center Latitude to determine the latitude of the target. Both data items need to be in decimal representation prior to their addition to determine the actual measured or calculated Motion Imagery target location.

The targetLocationOffsetLat has a real earth coordinate represented by a latitude-longitude pair. See Section 9.2.3.2 on reporting geographic values.

Conversion: IMAPB(-19.2, 19.2, 3)

<u>Valid Values</u>: The set of real numbers from -19.2 to 19.2 inclusive.

Example Value	**Example Encoded LS Value**		
	Tag	Length	Value
10.00 Degrees	0x0A	0x03	10.00 → IMAPB(-19.2, 19.2, 3) = 0x3A66 67

10.2.2.12 VTarget Pack: Item 11 – targetLocationOffsetLon

The *targetLocationOffsetLon* item is the longitude offset for the target from parent's Frame Center Longitude (e.g., MISB ST 0601 - Item 24) based on the WGS84 ellipsoid. This item has meaning only when embedding the VMTI LS in ST 0601 LS. The targetLocationOffsetLon adds to the Frame Center Longitude to determine the longitude of the target. Both data items need to be in decimal representation prior to their addition to determine the actual measured or calculated Motion Imagery target location.

The targetLocationOffsetLon has a real earth coordinate represented by a latitude-longitude pair. See Section 9.2.3.2 on reporting geographic values.

Conversion: IMAPB(-19.2, 19.2, 3).

<u>Valid Values</u>: The set of real numbers from -19.2 to 19.2 inclusive.

Example Value	**Example Encoded LS Value**		
12.00 Degrees	Tag	Length	Value
	0x0B	0x03	12.00 → IMAPB(-19.2, 19.2, 3) = 0x3E66 67

10.2.2.13 VTarget Pack: Item 12 – targetHae

The *targetHae* item is the height of the target expressed as height in meters above the WGS84 ellipsoid (HAE). Conversion: IMAPB(-900, 19000, 2).

<u>Valid Values:</u> The set of real numbers from -900 to 19,000 inclusive.

Example Value	**Example Encoded LS Value**		
10,000 meters	Tag	Length	Value
	0x0C	0x02	10000 → IMAPB(-900, 19000, 2) = 0x2A94

10.2.2.14 VTarget Pack: Item 13 – boundingBoxTopLeftLatOffset

The *boundingBoxTopLeftLatOffset* item is the latitude offset for the top left corner of target's geospatial bounding box from the parent's Frame Center Latitude (e.g., MISB ST 0601 - Item 23) based on the WGS84 ellipsoid. The boundingBoxTopLeftLatOffset adds to the Frame Center Latitude to determine the latitude of the top left corner of the target's geospatial bounding box. Both data items need to be in decimal representation prior to their addition to determine the actual measured or calculated Motion Imagery target location. See Section 9.2.3.2 on reporting geographic values.

Conversion: IMAPB (-19.2, 19.2, 3).

<u>Valid Values:</u> The set of real numbers from -19.2 to 19.2 inclusive.

Example Value	**Example Encoded LS Value**		
10.00 Degrees	Tag	Length	Value
	0x0D	0x03	10.00 → IMAPB(-19.2, 19.2, 3) = 0x3A66 67

10.2.2.15 VTarget Pack: Item 14 – boundingBoxTopLeftLonOffset

The *boundingBoxTopLeftLonOffset* item is the longitude offset for the top left corner of target's geospatial bounding box from the parent's Frame Center Longitude (e.g., MISB ST 0601 - Item 24) based on the WGS84 ellipsoid. Added to the Frame Center Longitude from the parent ST 0601 to determine the longitude of the top left corner of the target's geospatial bounding box. Both data items need to be in decimal representation prior to their addition to determine the actual measured or calculated Motion Imagery target location. See Section 9.2.3.2 on reporting geographic values

Conversion: IMAPB(-19.2, 19.2, 3).

<u>Valid Values:</u> The set of real numbers from -19.2 to 19.2 inclusive.

Example Value	Example Encoded LS Value		
10.00 Degrees	Tag	Length	Value
	0x0E	0x03	10.00 → IMAPB(-19.2, 19.2, 3) = 0x3A66 67

10.2.2.16 VTarget Pack: Item 15 – boundingBoxBottomRightLatOffset

The *boundingBoxBottomRightLatOffset* item is the latitude offset for the bottom right corner of target's geospatial bounding box from the parent's Frame Center Latitude (e.g., MISB ST 0601 - Item 23) based on the WGS84 ellipsoid. The boundingBoxBottomRightLatOffset adds to the Frame Center Latitude to determine the latitude of the bottom right corner of the target's geospatial bounding box. Both data items need to be in decimal representation prior to their addition to determine the actual measured or calculated Motion Imagery target location. See Section 9.2.3.2 on reporting geographic values

Conversion: IMAPB(-19.2, 19.2, 3).

Valid Values: The set of real numbers from -19.2 to 19.2 inclusive.

Example Value	Example Encoded LS Value		
10.00 Degrees	Tag	Length	Value
	0x0F	0x03	10.00 → IMAPB(-19.2, 19.2, 3) = 0x3A66 67

10.2.2.17 VTarget Pack: Item 16 – boundingBoxBottomRightLonOffset

The *boundingBoxBottomRightLonOffset* item is the longitude offset for the bottom right corner of target's geospatial bounding box from the parent's Frame Center Longitude based on the WGS84 ellipsoid. The boundingBoxBottomRightLonOffset adds to the Frame Center Longitude to determine the longitude of the bottom right corner of the target's geospatial bounding box. Both data items need to be in decimal representation prior to their addition to determine the actual measured or calculated Motion Imagery target location. See Section 9.2.3.2 on reporting geographic values.

Conversion: IMAPB(-19.2, 19.2, 3).

Valid Values: The set of real numbers from -19.2 to 19.2 inclusive.

Example Value	Example Encoded LS Value		
10.00 Degrees	Tag	Length	Value
	0x10	0x03	10.00 → IMAPB(-19.2, 19.2, 3) = 0x3A66 67

Requirement	
ST 0903.4-35	VTarget Pack boundaryBottomRightLonOffset (Item 16) shall only be present when using embedded-VMTI.

10.2.2.18 VTarget Pack: Item 17 – targetLocation

The *targetLocation* item provides detailed geo-positioning information for a target, optionally including the standard deviation and correlation coefficients. This item is of type Location which is a Defined Length Truncation Pack. To specify the geographic coordinates for a target with standalone-VMTI, use targetLocation in lieu of VTarget Pack - Item 10 targetLocationOffsetLat and Item 11 targetLocationOffsetLat. However, even when using embedded-VMTI, targetLocation is preferred vice offset calculations.

Requirement	
ST 0903.6-130	When using Standalone-VMTI a VTarget LS instance shall contain targetLocation (Item 17).

10.2.2.19 VTarget Pack: Item 18 – geospatialContourSeries

The *geospatialContourSeries* item is of type BoundarySeries, which provides detailed geo-positioning information for the contour around the target. An arbitrary number of vertices defines the contour. Each vertex is an element of type Location. The Location type captures geo-positioning data about a specific location on or near the surface of the Earth. Typical geospatial contours include boxes defined by two or four vertices, although other contours are possible.

Use a geospatialContourSeries instead of a target's geospatial bounding box (Items 13 through 16) when accuracy and correlation information is available and needed. Such information aids fusion with other moving object indicators, such as, radar based GMTI, to support track identification and tracking.

10.2.2.20 VTarget Pack: Item 19 – centroidPixRow

The *centroidPixRow* item specifies the row of the target centroid within the Motion Imagery frame in pixels. Numbering commences from 1, denoting the top row. The centroidPixRow may be used with VTarget Pack centroidPixCol - Item 20 to provide an alternate method to specify VTarget Pack targetCentroid – Item 1, the pixel location of the target centroid. If present, the centroidPixCol - Item 20 must also be present.

Valid Values: Integer values in the range 1 to 2^{32}-1.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
872	0x13	0x02	0x0368

Requirement	
ST 0903.4-37	Where VTarget Pack centroidPixCol (Item 20) is present, VTarget Pack centroidPixRow (Item 19) shall be present.

10.2.2.21 VTarget Pack: Item 20 – centroidPixCol

The *centroidPixCol* item specifies the column of the target centroid within the Motion Imagery frame in pixels. Numbering commences from 1, denoting the left column. May be used with VTarget Pack centroidPixRow - Item 19 to provide an alternate method to specify VTarget Pack targetCentroid – Item 1, the pixel location of the target centroid. If present, the centroidPixRow - Item 19 must also be present.

Valid Values: Integer values in the range 1 to 2^{32}-1.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
1137	0x14	0x02	0x0471

Requirement	
ST 0903.4-38	Where VTarget Pack centroidPixRow (Item 19) is present, VTarget Pack centroidPixCol (Item 20) shall be present.

10.2.2.22 VTarget Pack: Item 21 (DEPRECATED)

ST 0903.6 deprecates this item; refer to versions prior to ST 0903.6 for information on its use.

10.2.2.23 VTarget Pack: Item 22 – algorithmId

The *algorithmId* item refers to one of the algorithm ids in the VMTI LS algorithmSeries Item 102, which lists all the algorithms a VMTI LS uses. Each algorithm in the series includes an identifier (algorithmId). The algorithmId value equals one of the Id values in the algorithmSeries.

10.2.2.24 VTarget Pack: Item 23 – detectionStatus

The *detectionStatus* item allows assigning a target a status in detection. Section 7.2 describes the lifecycle states for a detected target.

10.2.2.25 VTarget Pack: Item 101 – vMask

The vMask item is a VMask LS which defines the outline of a detected target. Downstream clients can redraw the outline or “chip” the target from the Motion Imagery. Specifying the shape of the outline is by (1) three or more points representing the vertices of a polygon within a Motion Imagery frame, or (2) a bit mask identifying the pixels within the Motion Imagery frame subsumed by the target. There is no restriction in specifying both a polygon and a bit mask simultaneously.

10.2.2.26 VTarget Pack: Item 102 (DEPRECATED)

ST 0903.6 deprecates this item; refer to versions prior to ST 0903.6 for information on its use.

10.2.2.27 VTarget Pack: Item 103 (DEPRECATED)

ST 0903.6 deprecates this item; refer to versions prior to ST 0903.6 for information on its use.

10.2.2.28 VTarget Pack: Item 104 – vTracker

The vTracker item is a VTracker LS which contains spatial and temporal information ancillary to the VChip LS, VObject LS, and VFeature LS to assist in tracking the target. Such information allows Motion Imagery tracking algorithms to produce better tracks from the VMTI target information.

10.2.2.29 VTarget Pack: Item 105 – vChip

The vChip item is a VChip LS which enables embedding an image "chip" of the target into the VMTI metadata. The system creates the image chip from the same sensor and time as the detection. The VChip LS finds use in bandwidth constrained environments where the operator does not have access to the underlying Motion Imagery stream. Alternatively, the VChip LS includes using an Internationalized Resource Identifier (IRI), which can be in the form of a Uniform Resource Locator (URL), for linking to an externally stored image chip. An externally referred chip should have the same chip representation and data as within the stream. Alternatively, use the vChipSeries where there are more than one VChip.

10.2.2.30 VTarget Pack: Item 106 – vChipSeries

The *vChipSeries* item is a Series (see Figure 17) of one or more VChip LS associated with a specific target. The system creates the first image chip in the vChipSeries from the same sensor and time as the detection. The system adds supplemental images from alternative sensors but at the same time as the detection. When a system uses Item 106 vChipSeries, there is no need for Item 105.

Multiple image chips could be useful, for example, where an image chip from a source sensor and another image chip from a different sensor are of different resolutions or different modalities (e.g., IR).

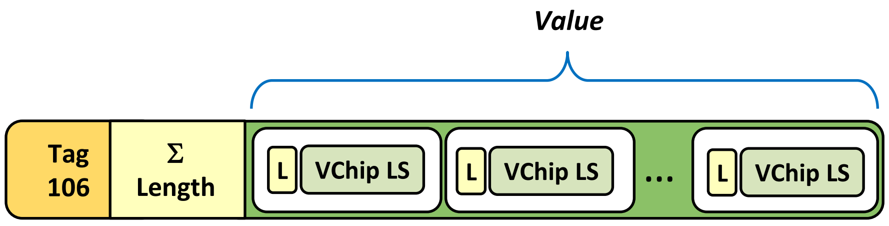


Figure 17: vChipSeries Structure

Requirement	
ST 0903.6-143	The image chip within the first VChip LS in a vChipSeries shall be from the VMTI-MI frame where the target was detected.

10.2.2.31 VTarget Pack: Item 107 – vObjectSeries

The *vObjectSeries* item is a Series (see Figure 18) of one or more VObject LS associated with a specific target.

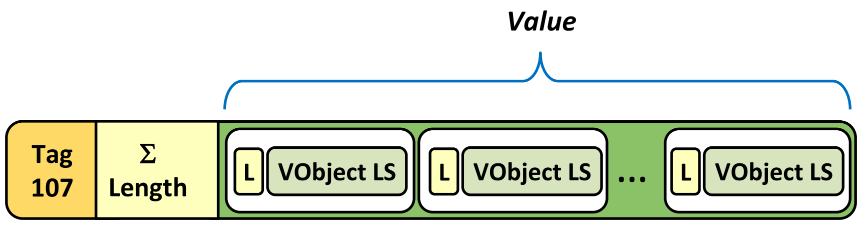


Figure 18: VObject Series Structure

10.3 VMask LS

Table 11 summarizes the VMask LS. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 11: VMask LS

VMask Local Set					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
1	pixelContour	N/A	Array	V	At least three unsigned integer numbers specifying the vertices of a polygon representing the outline of a target
2	bitMaskSeries	N/A	Series	V	Describes the area of the frame occupied by a target using a run-length encoded bit mask with 1 to indicate that a pixel includes a part of the target and 0 to indicate otherwise

10.3.1 VMask LS: Item 1 – pixelContour

A *pixelContour* item is an Array type of three or more points representing the vertices of a polygon within a Motion Imagery frame listed in clockwise order. Close the polygon by connecting the last point to the first point. Each point is a pixel number with numbering commencing with 1, at the top left pixel, proceeding from left to right, top to bottom, then encoded using the Length-Value construct of a Variable Length Pack. Note: in the UML of the VMTI LS, the closed brackets [ ] indicates an array.

The calculation of the pixel number is pixel number = Column + ((Row-1) x frame width)). The top left pixel of the frame equates to (Column, Row) = (1, 1) and pixel number = 1. For example, for frame width = 1920 and pixel location (1, 1) the pixel number = 1; for pixel location (2, 1) the pixel number = 2; for pixel location (1, 2) the pixel number = 1921.

Example Value	Example Encoded LS Value		
	Tag	Length	Value (Len, pixel number)
14762, 14783, 15115	0x01	0x09	0x02 39AA 0x02 39BF 0x02 3B0B

Requirement(s)	
ST 0903.4-40	The VMask LS pixelContour (Item 1) shall list the contour points in clockwise order.
ST 0903.5-100	A pixelContour pixel number shall be no greater than the number of pixels in a VMTI-MI frame.

10.3.2 VMask LS: Item 2 – bitMaskSeries

The *bitMaskSeries* item is a Series type defining a run length encoding of a bit mask describing the pixels which include the target within the Motion Imagery frame. As shown in Figure 19, the VMask LS *Value* consists of multiple pixel-runs. Each pixel-run specifies the starting pixel number (i.e., pixel) and the number of pixels in a run (i.e., Run BER).

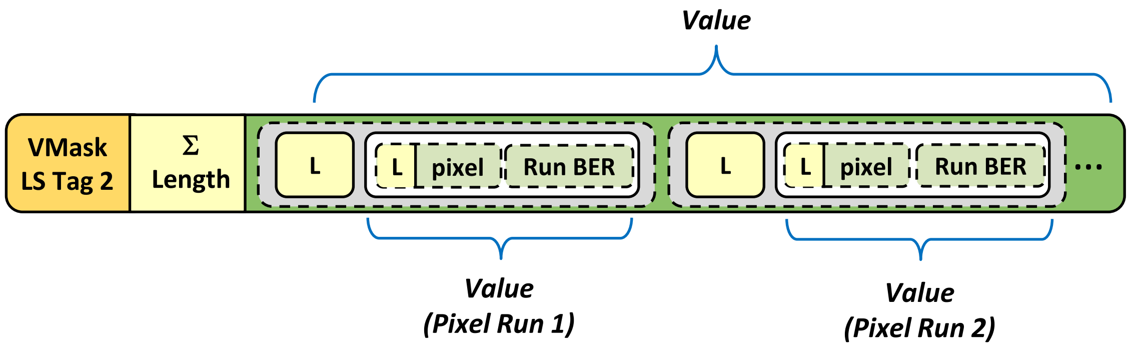


Figure 19: bitMaskSeries structure for encoding pixel-runs

Pixel numbering commences with 1, at the top left pixel, proceeding from left to right, top to bottom. Encode pixel numbers using the Length-Value (i.e., "L | pixel" in Figure 19) construct of a Variable Length Pack. Encode the length of each run using BER-Length encoding. The criterion to decide whether a pixel "covers" a portion of the target is arbitrary and left to the implementer. The implementer is free to decide whether overlap with all, a majority, or a fraction of pixels constitutes "covering" the target.

The calculation of the pixel number is pixel number Column + ((Row-1) x frame width)). The top left pixel of the frame equates to (Column, Row) = (1, 1) with pixel number = 1, then encoded using the Length-Value construct of a Variable Length Pack.

For example, in Figure 20 the pairs of pixel numbers and run lengths (pixel, run) are:

(74, 2) = [0x01 4A] [0x02]   (89, 4) = [0x01 59] [0x04]   (106, 2) = [0x01 6A] [0x02]

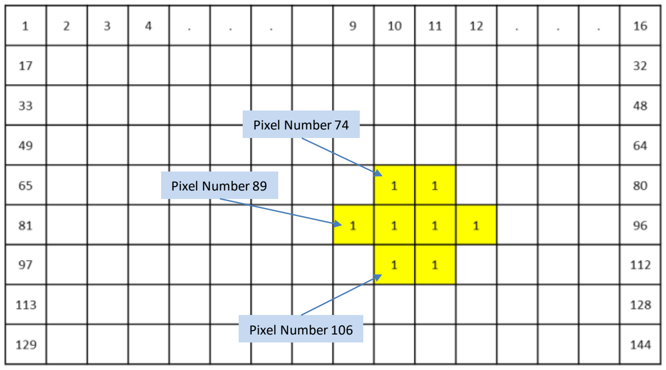


Figure 20: Example Bit Mask

In this example, each run length is a single byte using the short form of BER-Length encoding. The long form of BER-Length encoding is for run lengths exceeding 127 pixels. Run lengths typically (but not necessarily) will be "small" leveraging the bit efficiency of the short form.

Example Value	Example Encoded LS Value		
	Tag	Length	Value = (Len, (Len, Pixel Num), RunLen); (…) ; …
bit mask series = (74, 2); (89, 4); (106, 2)	0x02	0x0C	0x03, (0x01, 0x4A), 0x02; 0x03, (0x01, 0x59), 0x04; 0x03, (0x01, 0x6A), 0x02

Requirement	
ST 0903.5-101	A bitMaskSeries pixel-run starting pixel number plus run-length shall be less than the line length of the VMTI-MI image.

10.4 VObject LS

Table 12 summarizes the VObject LS. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 12: VObject LS

VObject Local Set					
Tag	Name	Units	Format	Len	Description
1	Item DEPRECATED	---	---	---	ST 0903.6 deprecates this item
2	Item DEPRECATED	---	---	---	ST 0903.6 deprecates this item
3	ontologyId	N/A	uint	V8	Identifier indicating which Ontology in the VMTI's Ontology Series represents this object
4	confidence	N/A	IMAP	V3	The amount of confidence in the label of this object
5	vFeatureSeries	N/A	Series	V	One or more VFeature LS associated with a specific VObject

10.4.1 VObject: Item 1 (DEPRECATED)

ST 0903.6 deprecates this item; refer to versions prior to ST 0903.6 for information on its use.

10.4.2 VObject: Item 2 (DEPRECATED)

ST 0903.6 deprecates this item; refer to versions prior to ST 0903.6 for information on its use.

10.4.3 VObject: Item 3 – ontologyId

The *ontologyId* is a reference to one of the ontologies in the VMTI LS ontologySeries. Each item in the ontologySeries defines either an object label or feature label, see Section 7.3. Each ontology in the ontologySeries includes an identifier (ontologyId) so the value of the VObject LS ontologyId is equal to one of the ontologyId values in the ontologySeries. Using the ontologyId saves bandwidth by not duplicating the same information for different VObjects.

Requirement	
ST 0903.6-131	The value of a VObjects LS ontologyId (Item 3) shall be equal to one of the Ontology LS's ids in the VMTI LS's ontologySeries.

10.4.4 VObject: Item 4 – confidence

The *confidence* item is the measure of "trust" in the labeling of this vObject, ranging from 0.0% to 100.0%. For example, an object-classifier analyzes a blob of pixels and labels the blob as representing a vehicle. If the blob closely matches the classifiers criteria for a vehicle the confidence in the labeling is high (towards 100.0%); alternatively, if the classifier is less sure of its label the confidence is low (towards 0%). The confidence value is IMAPB(0.0, 100.0, length) with the length defined by the tag's length value. Increasing the length provides more accuracy. The minimum length is 1.

10.4.5 VObject: Item 5 - vFeatureSeries

The *vFeatureSeries* item is a Series (see Figure 21) of one or more VFeature LS associated with a specific VObject.

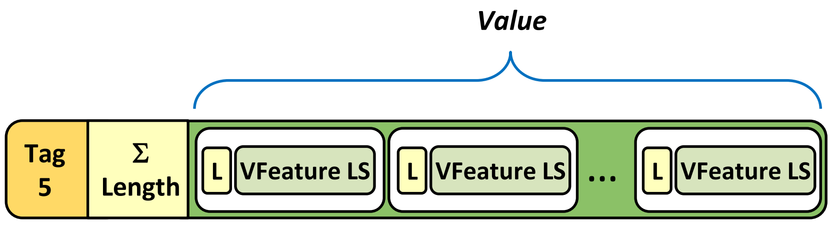


Figure 21: VFeature Series Structure

10.5 VFeature LS

Table 13 summarizes the VFeature LS. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 13: VFeature LS

VFeature Local Set					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
1	Item DEPRECATED	---	---	---	ST 0903.6 deprecates this item
2	Item DEPRECATED	---	---	---	ST 0903.6 deprecates this item
3	ontologyId	N/A	uint	V8	Identifier indicating which ontology in the VMTI LS's ontologySeries represents this feature
4	confidence	N/A	IMAP	V3	The amount of confidence in the feature's label.

10.5.1 VFeature LS: Item 1 (DEPRECATED)

ST 0903.6 deprecates this item; refer to versions prior to ST 0903.6 for information on its use.

10.5.2 VFeature LS Item 2 (DEPRECATED)

ST 0903.6 deprecates this item; refer to versions prior to ST 0903.6 for information on its use.

10.5.3 VFeature LS Item 3 – ontologyId

The *ontologyId* is a reference to one of the ontologies in the VMTI LS ontologySeries. Each ontology in the ontologySeries includes an identifier (Id) so the value of the VFeature LS ontologyId is equal to one of the id values in the ontologySeries. Using the ontologyId saves bandwidth by not duplicating the same information for different VFeatures.

Requirement	
ST 0903.6-132	The value of VFeature LS's ontologyId shall be equal to one of the Ontology LS's ids in the VMTI LS's ontologySeries.

10.5.4 VFeature LS Item 4 – confidence

The *confidence* item is the measure of "trust" in the feature-classification of this vFeature from 0.0% to 100.0%. For example, a feature-classifier analyzes an object and "classifies" a subset of the object's pixels as an antenna. If the pixels closely match the classifier's criteria for an antenna the confidence in the feature-classification is high (towards 100.0%); alternatively, if the classifier is less sure of its feature-classification the confidence is low (towards 0%). The confidence value is IMAPB(0.0, 100.0, length) with the length defined by the item's length value. Increasing the length provides more accuracy. The minimum length is 1.

10.6 VTracker LS

Table 14 summarizes the VTracker LS. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 14: VTracker LS

VTracker Local Set					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
1	trackId	N/A	UUID	16	A unique identifier (UUID) for the track
2	Item DEPRECATED	---	---	---	ST 0903.6 deprecates this item
3	firstObsvTime	µs	uint	8	Time of the first observation of the entity
4	latestObsvTime	µs	uint	8	Time for the latest (most recent) observation of the entity
5	trackBoundarySeries	N/A	Series	V	Set of vertices of type Location that specify a bounding area, which encloses the full extent of VMTI detections for the entity
6	Item DEPRECATED	---	---	---	ST 0903.6 deprecates this item
7	confidenceLevel	N/A	uint	1	An estimation of the certainty or correctness of VMTI movement detections. Larger values indicate greater confidence.
8	Item DEPRECATED	---	---	---	ST 0903.6 deprecates this item
9	trackHistorySeries	N/A	Series	V	Points of type Location that represent the locations of VMTI detections
10	velocity	N/A	Velocity	V	Velocity of the entity at the time of latest (most recent) observation
11	acceleration	N/A	Acceleration	V	Acceleration of the entity at the time of latest (most recent) observation
12	algorithmId	N/A	uint	V	Identifier indicating which algorithm in the Algorithm Series tracked this target

10.6.1 *VTracker LS: Item 1 – trackId*

Uniquely identifies a track, using a 128-bit (16-byte) Universal Unique Identification (UUID) as standardized by the Open Software Foundation according to ISO/IEC 9834-8 [11].

Example Value	Example Encoded LS Value		
	Tag	Length	Value
f81d4fae-7dec-11d0-a765-00a0c91e6bf6	0x01	0x10	0xF81D 4FAE 7DEC 11D0 A765 00A0 C91E 6BF6

10.6.2 *VTracker LS: Item 2 (DEPRECATED)*

ST 0903.6 deprecates this item; refer to versions prior to ST 0903.6 for information on its use.

10.6.3 VTracker LS: Item 3 – firstObsvTime

The *firstObsvTime* item captures the time of the first observation of the entity using the MISP Precision Time Stamp, see section 9.2.2.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
April 19, 2001, 04:25:21 GMT (987654321000000)	0x03	0x08	0x0003 8244 30F6 CE40

10.6.4 VTracker LS: Item 4 – latestObsvTime

The *latestObsvTime* item captures the time of the most recent or latest observation of the entity using the MISP Precision Time Stamp, see section 9.2.2.

10.6.5 VTracker LS: Item 5 – trackBoundarySeries

The *trackboundarySeries* item is of type BoundarySeries which specifies a 2D bounding area that encloses the full extent of VMTI detections for the entity. For a simple, geospatial bounding box, the area will generally lie on the surface of the Earth (although not necessarily, depending upon the Height values provided) defining the “footprint” of the track.

10.6.6 VTracker LS: Item 6 (DEPRECATED)

ST 0903.6 deprecates this item; refer to versions prior to ST 0903.6 for information on its use.

10.6.7 VTracker LS: Item 7 – confidenceLevel

The *confidenceLevel* item, expressed as a percentage from 0 to 100, is an estimation of the certainty or correctness that the track described by the sequence of VMTI movement detections corresponds to the same object. A value = 0 indicates no confidence; a value = 100 indicates absolute certainty. Confidence is an estimation of the certainty or correctness that the track described by the sequence of VMTI movement detections corresponds to the same object. For example, detections derived from many unambiguous target reports, such as, for a single vehicle on a road in a desert environment might signal high confidence. Reports associated with several overlapping or nearby tracks in a partially obscured environment, such as, for dismounts (people) in an urban setting might signal low confidence.

Valid values: The set of all integers from 0 to 100 inclusive.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
50%	0x07	0x01	0x32

10.6.8 VTracker LS: Item 8 (DEPRECATED)

ST 0903.6 deprecates this item; refer to versions prior to ST 0903.6 for information on its use.

10.6.9 VTracker LS: Item 9 – trackHistorySeries

The trackHistorySeries item is a Series of points that represent the locations of entity VMTI detections. Each point is an element of type Location. Points are in chronological order from the first to the most recent VMTI detection.

Requirement	
ST 0903.4-56	The VTracker LS trackHistorySeries item (Item 9) shall contain the points in chronological order from start to end of the VMTI detections.

10.6.10 VTracker LS: Item 10 – velocity

The *velocity* item is the velocity of the entity at the time of latest (most recent) detection.

Valid Values: See Velocity type

10.6.11 VTracker LS: Item 11 – acceleration

The *acceleration* item is the acceleration of the entity at the time of latest (most recent) detection.

Valid Values: See Acceleration type

10.6.12 VTracker LS: Item 12 – algorithmId

The *algorithmId* item specifies the identifier number assigned to the detection algorithm used.

Example Value	**Example Encoded LS Value**		
	Tag	Length	Value
3	0x0B	0x01	0x03

Requirement	
ST 0903.6-133	The VTracker LS algorithmId (Item 12) value shall refer to the algorithmId of one of the elements in the VMTI LS algorithmSeries (Item 102).

10.7 VChip LS

The purpose of the VChip LS is to provide a basic visual target representation.

Table 15 summarizes the VChip LS. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 15: VChip LS

VChip Local Set					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
1	imageType	N/A	utf8	V	Internet Assigned Numbers Authority (IANA) image media subtype specifying the VChip image type (limited to "jpeg" and "png")
2	imageIRI	N/A	utf8	V	Internationalized Resource Identifier referring to an image stored on a server
3	embeddedImage	N/A	binary	V	An image "chip" of the image type specified by Item 1

10.7.1 VChip LS: Item 1 – imageType

The *imageType* item is a UTF-8 string which corresponds to an IANA media image subtype. The only allowed subtypes are "jpeg" and "png" which are common formats for compressing a still image. JPEG is a lossy compression method, but quality is adjustable. PNG is lossless preserving "raw" pixel values providing RGB bit depths up to 48 bits per pixel (16 bits per color component).

Item 1 - imageType is a required item in the VChip LS.

Valid Values: A string of UTF-8 characters that correspond to an IANA media image subtype.

Example Value	**Example Encoded LS Value**		
	Tag	Length	Value
jpeg	0x01	0x04	0x6A70 6567

Requirement(s)	
ST 0903.6-134	Every VChip LS instance shall include the imageType (Item 1).
ST 0903.4-59	The VChip LS imageType (Item 1) shall state the type of image referred to by VChip LS imageIRI (Item 2).
ST 0903.6-135	The VChip LS imageType (Item 1) shall state the type of VChip LS embeddedImage (Item 3).
ST 0903.6-136	The VChip LS imageType (Item 1) shall be either "jpeg" or "png".

10.7.2 VChip LS: Item 2 – imageIRI

The *imageIRI* item is an Internationalized Resource Identifier (usually, a Uniform Resource Locator) that refers to an image of the type specified by VChip LS imageType - Item 1, stored on a network or a file system. In some situations, likely downstream from the collection source, using a referred image eliminates embedding the image chip into the stream.

Valid Values: A string of UTF-8 characters that comply with the rules for building a valid IRI.

10.7.3 VChip LS: Item 3 – embeddedImage

The *embeddedImage* item is an image chip of the type specified by VChip LS imageType - Item 1 embedded in the VMTI stream.

Valid Values: Any image implemented in compliance with the IANA media image subtype specified in Item 1.

10.8 Algorithm LS

The Algorithm LS documents attributes of the algorithm used for detection and tracking of targets. The VMTI LS algorithmSeries – Item 102 conveys one or more instances of the LS allowing for documenting different algorithms in use within one VMTI LS.

Table 16 summarizes the Algorithm LS. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 16: Algorithm LS

Algorithm Local Set					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
1	algorithmId	N/A	uint	V8	Identifier for the algorithm used
2	name	N/A	utf8	V	Name of algorithm
3	version	N/A	utf8	V	Version of algorithm
4	class	N/A	utf8	V	Type of algorithm e.g., detector classifier
5	nFrames	N/A	uint	V8	Number of frames the algorithm operates over

10.8.1 Algorithm LS: Item 1 – algorithmId

When the VMTI LS contains a Series of Algorithm LS, each element in the Series requires a unique identifier (*algorithmId*). The algorithmId is an integer which identifies a single Algorithm LS in the Series and is unique among all other elements in the list. Systems may reuse algorithmIds from VMTI LS to VMTI LS (i.e., two sequential VMTI packets) so receivers should not assume an identifier value is static for an entire VMTI stream. The algorithmId does not need to start with a value of one (1) nor do the algorithmIds need to be in any specific order in an algorithmSeries.

Requirement(s)	
ST 0903.5-102	All instances of the Algorithm LS shall contain an algorithmId (Item 1).
ST 0903.6-137	All Algorithm LSs in an Algorithm LS series shall have unique algorithmIds (Item 1).

Valid Values: An unsigned integer.

Example Value	**Example Encoded LS Value**		
	Tag	Length	Value
9	0x01	0x01	0x09

10.8.2 Algorithm LS: Item 2 – name

The *name* item is the name assigned to the algorithm by the data producer.

Requirement	
ST 0903.5-103	All instances of the Algorithm LS shall contain a name (Item 2).

Valid Values: Any alphanumeric value in UTF8.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
k6_yolo_9000_tracker	0x02	0x14	0x6B36 5F79 6F6C 6F5F 3930 3030 5F74 7261 636B 6572

10.8.3 Algorithm LS: Item 3 – version

The *version* item is the version of the algorithm.

Requirement	
ST 0903.5-104	All instances of the Algorithm LS shall contain version (Item 3).

Valid Values: Any alphanumeric value in UTF8.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
2.6a	0x03	0x04	0x322E 3661

10.8.4 Algorithm LS: Item 4 – class

The *class* item is the type of algorithm.

Valid Values: Any alphanumeric value in UTF8.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
kalmann	0x04	0x07	0x6B61 6C6D 616E 6E

10.8.5 Algorithm LS: Item 5 – nFrames

The *nFrames* item is the number of frames the algorithm processes when detecting or tracking the object.

Example Value	Example Encoded LS Value		
	Tag	Length	Value
10	0x05	0x01	0x0A

10.9 Ontology LS

The Ontology LS describes the class or type of a target (aircraft, watercraft, car, truck, train, dismount, etc.) to an arbitrary level of detail. This standard mandates the use of the Web Ontology Language (OWL) [4] to define the ontology, see Section 7.3.

The Ontology LS requires, ontologyId – Item 1, ontologyIRI – Item 3, and entityIRI - Item 4 to be present.

Table 17 summarizes the Ontology LS. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 17: Ontology LS

Ontology Local Set					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
1	ontologyId	N/A	uint	V8	Identifier for the ontology used
2	parentId	N/A	uint	V8	Defines the link when an OntologySeries has two related LS in the Series
3	ontologyIRI	N/A	utf8	V	Internationalized Resource Identifier (IRI) which identifies the OWL ontology for the entityIRI.
4	entityIRI	N/A	utf8	V	Internationalized Resource Identifier (IRI) specifying an entity from the ontology.
5	versionIRI	N/A	utf8	V	Internationalized Resource Identifier (IRI) specifying the version of the ontology.
6	label	N/A	utf8	V	Entity label defined in the ontology

10.9.1 Ontology LS: Item 1 – ontologyId

When the VMTI LS contains a Series of Ontology LS, each element in the Series gets a unique identifier. The *ontologyId* item is an integer which identifies a single Ontology LS in the Series and is unique in the list. Systems may reuse ontologyIds for different ontologies/entities from VMTI LS to VMTI LS (i.e., two sequential VMTI packets) so receivers should not assume an ontologyId value is static for a whole VMTI stream. The ontologyId does not need to start at a value of one nor do the ontologyIds need to be in any specific order in the Ontology Series.

Requirement(s)	
ST 0903.5-105	All instances of the Ontology LS shall contain an ontologyId (Item 1).
ST 0903.6-138	All Ontology LSs in an Ontology LS series shall have unique ontologyIds (Item 1).

10.9.2 Ontology LS: Item 2 – parentId

The parentId enables relating one ontology reference to another. When detecting or tracking objects, there may be several related object-labels made for an object. From the example in Section 7.3, vehicle is a generalization of both car and motorcycle. Systems may use the *parentId* item to indicate the parent-child relationships between car and vehicle, motorcycle, and vehicle.

However, this information is duplicative to the ontology hierarchy; therefore starting with ST0903.6 the MISP suggests not using this item.

When an ontologySeries has two related Ontology LS in the Series, the parentId defines the link by the child defining its parentId equal to the parent ontology object's ontologyId. For example, consider an ontologySeries having three elements: vehicle with ontologyId 10, car with ontologyId 17, and motorcycle with ontologyId 3. Since car and motorcycle are both "children" of vehicle, those two LS define their parentId as equal to 10.

10.9.3 Ontology LS: Item 3 – ontologyIRI

The *ontologyIRI* identifies the ontology which provides the definition of the entityIRI. See Section 7.3.

Requirement(s)	
ST 0903.5-106	The ontology referred to by the IRI of the Ontology LS - ontologyIRI (Item 3) item shall reference an OWL ontology.
ST 0903.5-107	All instances of the Ontology LS shall contain an ontologyIRI (Item 3).

10.9.4 Ontology LS: Item 4 – entityIRI

The *entityIRI* identifies an entity within the ontology. See Section 7.3.

Requirement(s)	
ST 0903.5-108	All instances of the Ontology LS shall contain entityIRI (Item 4).
ST 0903.6-139	All instances of entityIRI (Item 4) shall have a value taken from the ontologyIRI (Item 3) specified in the Ontology LS.

10.9.5 Ontology LS: Item 5 – versionIRI

The *versionIRI* identifies the ontology version. See Section 7.3.

10.9.6 Ontology LS: Item 6 – label

The *label* is the name of the entity, defined in the entityIRI, as defined by the ontology. See Section 7.3. The label text is either the value of the (OWL defined) rdfs:label[4] or skos:prefLabel[5] property of the entity

[4] See [15] (specifically https://www.w3.org/TR/rdf-schema/#ch_label)

[5] See [15] (specifically https://www.w3.org/2009/08/skos-reference/skos.html#prefLabel)

Requirement	
ST 0903.6-140	Where the Ontology LS includes a label (Item 6), the label shall exactly match the rdfs:label or skos:prefLabel in the defining ontologyIRI (Item 3).

10.10 Location, Velocity, Acceleration DLPs

Location, Velocity, and Acceleration are DLP truncation structures. The Location, Velocity and Acceleration DLPs each contain three groups of information as discussed below. Truncation structures allow lower priority elements (in this case a group) to be "truncated" or not included in the DLP, thus saving bandwidth. Truncation begins with the trailing information within the pack.

Requirement(s)	
ST 0903.4-62	Truncation of Location, Velocity, and Acceleration Defined-Length Truncation Packs shall be allowed only as a group.
ST 0903.4-63	Within a Location, Velocity, or Acceleration Defined-Length Truncation Pack, no filler values shall be used for (unknown) higher priority elements.

10.10.1 Location DLP

The Location DLP captures geo-positioning data about a specific location on or near the surface of the Earth. Three groups of information form a Location DLP shown in Figure 22.

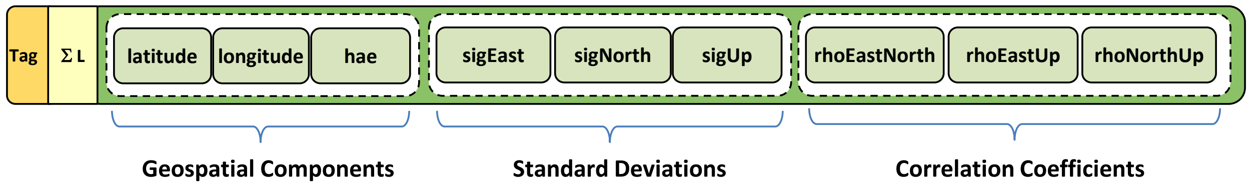


Figure 22: Location DLP Structure

The first, and highest priority group, includes latitude, longitude, and height above ellipsoid (hae). These values define the origin of a local tangential East-North-Up (ENU) coordinate system. The second group, Standard Deviations, and the third group, Correlation Coefficients, express uncertainty with respect to the ENU coordinate axes. Standard Deviations is a medium priority group providing standard deviations for the location in meters (with respect to the ENU coordinate axes). Correlation Coefficients is the lowest priority group providing correlation coefficients for the location (with respect to the ENU coordinate axes). The correlation coefficients provide a measure of systematic behavior, such as whether variation in the values of pairs of variables are random[6] or "coupled".

Standard deviations and correlation coefficients express confidence in the geo-coordinates. For example, if standard deviations are small and correlation coefficients are near unity, then

[6] Correlation is dimensionless with a fixed range of values from -1.0 to +1.0, inclusive. Covariance, while it is a similar measure of "relatedness," is in units obtained by multiplying the units of two variables, and thus has values less well constrained.

“confidence” in the accuracy of the coordinate values is high. Conversely, if standard deviations are large and correlation coefficients are near zero potentially large, random errors are likely.

The Location DLP is a truncation pack where the standard deviations and correlation coefficients groups are optional; thus, when not included truncate the correlation coefficients group first followed by the standard deviations group.

Table 18 summarizes the Location DLP. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 18: Location DLP Truncation Pack

Location DLP Truncation Pack					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
N/A	latitude	°	IMAPB(-90, 90, 4)	4	Latitude in degrees of a point with respect to the WGS84 datum
N/A	longitude	°	IMAPB(-180, 180, 4)	4	Longitude in degrees of a point with respect to the WGS84 datum
N/A	hae	m	IMAPB(-900, 19000, 2)	2	Height of a point in meters above the WGS84 Ellipsoid (HAE)
N/A	sigEast	m	IMAPB(0, 650, 2)	2	Standard deviation of the location of the point with respect to the ENU coordinate system East axis
N/A	sigNorth	m	IMAPB(0, 650, 2)	2	Standard deviation of the location of the point with respect to the ENU coordinate system North axis
N/A	sigUp	m	IMAPB(0, 650, 2)	2	Standard deviation of the location of the point with respect to the ENU coordinate system Up axis
N/A	rhoEastNorth	N/A	IMAPB(-1, 1, 2)	2	Correlation coefficient between the East and North components of error
N/A	rhoEastUp	N/A	IMAPB(-1, 1, 2)	2	Correlation coefficient between East and Up components of error
N/A	rhoNorthUp	N/A	IMAPB(-1, 1, 2)	2	Correlation coefficient between North and Up components of error

Requirement	
ST 0903.4-66	The Geospatial Coordinates triplet of a Location Pack shall always be present.

10.10.2 Velocity DLP

The Velocity DLP captures data about the velocity of a moving object. Three groups of information form a Velocity DLP as shown in Figure 23.

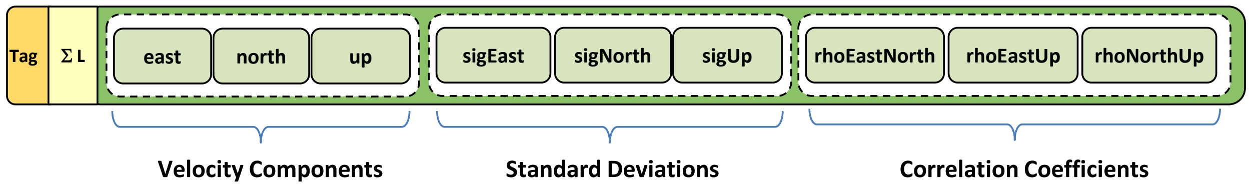


Figure 23: Velocity DLP Structure

The first, and highest priority group includes East, North, and Up velocity components; these provide the measurements of velocity along the coordinate axes of the East-North-Up coordinate system specified by the Location Truncation Pack for the location of the moving object. The second, and medium priority group, includes Standard Deviations for the first group measurements. The third, and lowest priority group includes Correlation Coefficients for values in the first group.

The Velocity DLP is a truncation pack where the standard deviations and correlation coefficients groups are optional; thus, when not included truncate the correlation coefficients group first followed by the standard deviations group.

Table 19 summarizes the Velocity DLP. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 19: Velocity DLP Truncation Pack

Velocity DLP Truncation Pack					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
N/A	east	m/s	IMAPB(-900, 900, 2)	2	Velocity along the East axis of the East-North-Up coordinate system
N/A	north	m/s	IMAPB(-900, 900, 2)	2	Velocity along the North axis of the East-North-Up coordinate system
N/A	up	m/s	IMAPB(-900, 900, 2)	2	Velocity along the Up axis of the East-North-Up coordinate system
N/A	sigEast	m/s	IMAPB(0, 650, 2)	2	Standard deviation along East axis
N/A	sigNorth	m/s	IMAPB(0, 650, 2)	2	Standard deviation along North axis
N/A	sigUp	m/s	IMAPB(0, 650, 2)	2	Standard deviation along Up axis
N/A	rhoEastNorth	m/s	IMAPB(-1, 1, 2)	2	Correlation coefficient between East and North
N/A	rhoEastUp	m/s	IMAPB(-1, 1, 2)	2	Correlation coefficient between East and Up
N/A	rhoNorthUp	m/s	IMAPB(-1, 1, 2)	2	Correlation coefficient between North and Up

Requirement(s)	
ST 0903.4-75	The Velocity Components triplet of the Velocity pack shall be present.
ST 0903.4-82	The Velocity pack shall be present only if there is an associated Location pack.

10.10.3 Acceleration DLP

The Acceleration DLP captures data about the acceleration of a moving object. Three groups of information form an Acceleration DLP as shown in Figure 24.

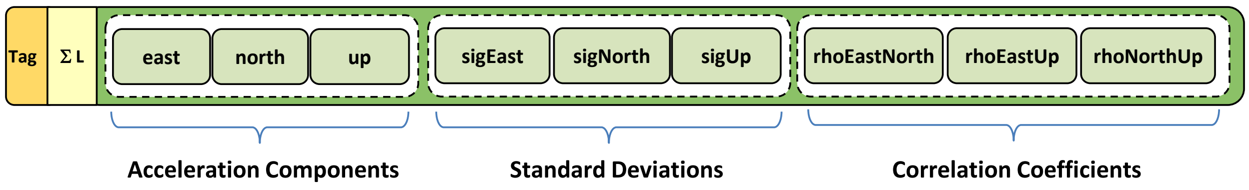


Figure 24: Acceleration Structure

The first, and highest priority, group includes East, North, and Up components; these measurements provide acceleration along the coordinate axes of the East-North-Up coordinate system specified by the Location pack for the location of the moving object. The second, and medium priority, group includes Standard Deviations for the first group measurements. The third, and lowest priority group includes Correlation Coefficients for values in the first group.

The Acceleration DLP is a truncation pack where the standard deviations and correlation coefficients groups are optional; thus, when not included truncate the correlation coefficients group first followed by the standard deviations group.

Table 20 summarizes the Acceleration DLP. See Section Local Set Table Structure Section 9.2.4 for column designations.

Table 20: Acceleration DLP Truncation Pack

Acceleration DLP Truncation Pack					
Tag	**Name**	**Units**	**Format**	**Len**	**Description**
N/A	east	m/s^2	IMAPB(-900, 900, 2)	2	Acceleration along the East axis of the East-North-Up coordinate system
N/A	north	m/s^2	IMAPB(-900, 900, 2)	2	Acceleration along the North axis of the East-North-Up coordinate system
N/A	up	m/s^2	IMAPB(-900, 900, 2)	2	Acceleration along the Up axis of the East-North-Up coordinate system
N/A	sigEast	m/s^2	IMAPB(0, 650, 2)	2	Standard deviation along East axis
N/A	sigNorth	m/s^2	IMAPB(0, 650, 2)	2	Standard deviation along North axis
N/A	sigUp	m/s^2	IMAPB(0, 650, 2)	2	Standard deviation along Up axis
N/A	rhoEastNorth	N/A	IMAPB(-1, 1, 2)	2	Correlation coefficient between East and North
N/A	rhoEastUp	N/A	IMAPB(-1, 1, 2)	2	Correlation coefficient between East and Up
N/A	rhoNorthUp	N/A	IMAPB(-1, 1, 2)	2	Correlation coefficient between North and Up

Requirement(s)	
ST 0903.4-85	The Acceleration Components triplet of the Acceleration pack shall always be present.

ST 0903.4-92	The Acceleration pack shall be present only if there is an associated Location pack.

10.11 BoundarySeries DLP

The *BoundarySeries* is a DLP comprised of a Series of Location Packs which define the vertices of the outer boundary, or contour, of a drape-surface. A drape-surface is the "top" surface of an area of interest or object. For example, using a metaphorical cloth to cover a target (e.g., a tank on a hill), defines the drape-surface. The contour of the drape-surface, e.g., the part of the cloth covering the truck touching the ground, is a closed non-planar polygon with multiple vertices. The non-planar polygon means the polygon's vertices are 3D coordinates and the vertices do not need to be on the same mathematical plane, i.e., each vertex may have different HAE values to match the terrain or object's contour.

Figure 25 illustrates the Cross-Section (side view) and Top-View of a drape-surface (blue) over a tank which is on a hill with a ditch behind the tank. The cross-section view illustrates the drape-surface intersecting the ground at the red points. The top-view shows the complete contour (red) around the tank, with black points showing a list of points to describe the tank contour.

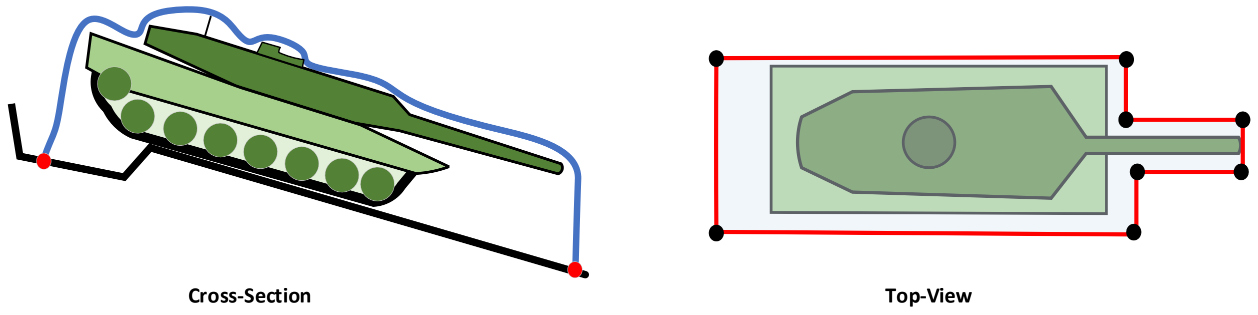


Figure 25: Example Drape-Surface with Contour

A "normalized" contour is an ellipsoid polygon (i.e., a polygon on the surface of the zero-HAE ellipsoid) formed by projecting a drape-surface's contour points to the zero-HAE ellipsoid. A BoundarySeries requires the normalized contour to have only one interior area. A normalized contour may have co-incident points or stranded points. Co-incident points are two (or more) adjacent contour points with the same latitude and longitude but different HAE; when projected to the ellipsoid those points become the same point in the normalized contour. Stranded points are one or more points lying outside the normalized contour's interior area, and do not make their own interior area. If the contour meets the single-area requirement (i.e., one enclosed area) co-incident and stranded points present no issue.

Figure 26 illustrates different normalized contours. Normalized contour (a) is a simple 8-point contour with a single interior area. Normalized contour (b) shows points 4 (red) and 5 as co-incident; since there is not an additional interior area, the contour is valid. Normalized contour (c) shows points 4 and 5 "stranded" or outside the interior area; since there is not an additional interior area, the contour is valid. Normalized contour (d) shows point 6 intersecting the segment between points 3 and 4 creating a second interior area. This is an invalid normalized contour; therefore, the original 3-D contour is invalid.

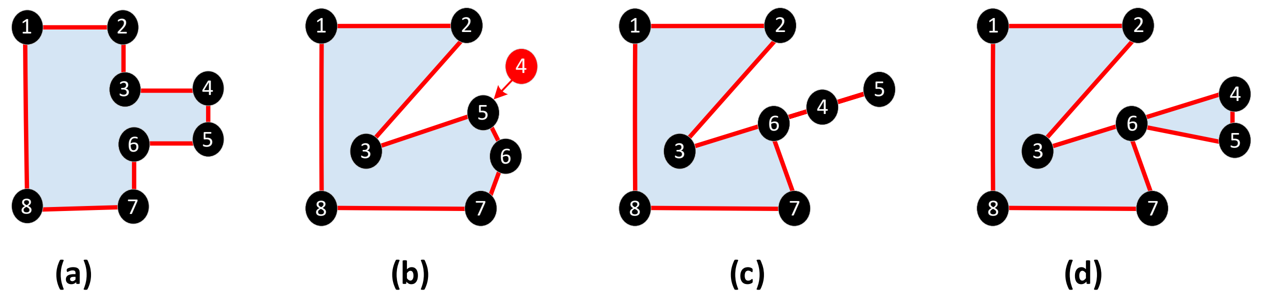


Figure 26: Normalized Contour Examples

The drape-surface description only provides a means to define the contour for this standard and it is not the intention to be "the" method for constructing a contour. The construction of a contour (or BoundarySeries) is implementation dependent; however, when normalizing the contour, it must only contain one area. The BoundarySeries only defines the outer edge of the drape-surface, therefore, features extending up or down within the drape-surface are not part of the BoundarySeries.

Requirement	
ST 0903.6-141	A BoundarySeries' normalized contour shall contain only one interior area.

Figure 27 is an example of a seven-point boundary of a drape-surface on the roof sections of a sports arena. Points 1-3 and 5-7 have similar HAE values; however, Point 4 has a higher HAE than the rest of the boundary. The inner part of the drape-surface has a much lower HAE (i.e., the sports field), but the BoundarySeries does not include this information.

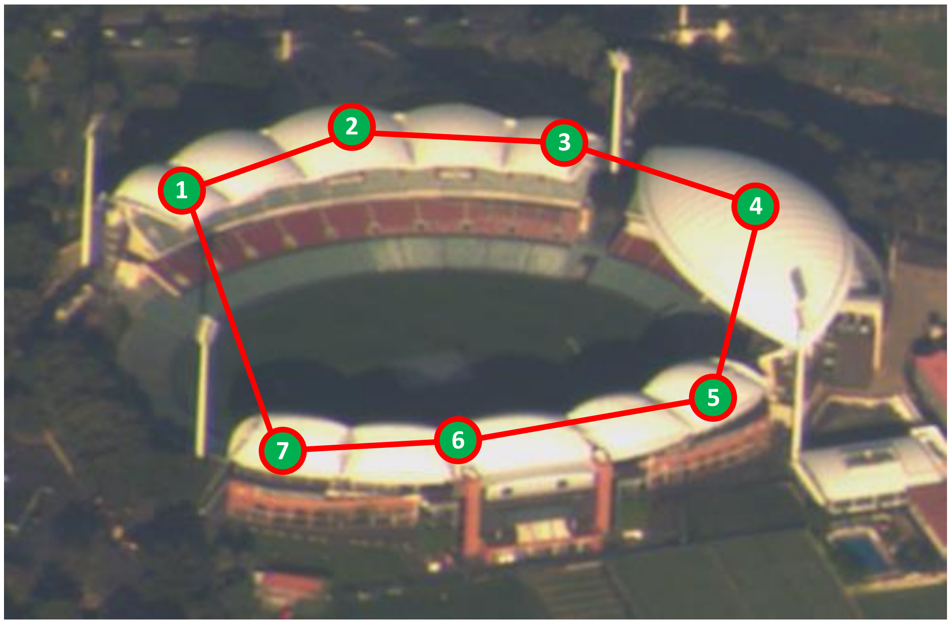


Figure 27: Example of a Boundary

There is no limit on the number of vertices in a BoundarySeries, although at least two are the minimum required. Specifying two vertices imply opposite corners of a simple, geospatial bounding box, aligned with the Latitude-Longitude grid, which is the simplest approximation of a drape-surface.

The BoundarySeries structure includes the BER-encoded Length for each value. Each vertex is of type Location.

Requirement(s)	
ST 0903.4-93	A BoundarySeries shall contain at least two vertices.
ST 0903.4-94	Where a BoundarySeries contains only two vertices, the vertices shall be opposite corners of a simple, geospatial bounding box, aligned with the Latitude-Longitude grid.

11 Deprecated Requirements

ST 0903.4-02 (Deprecated)	All data encoded using the Key-Length-Value (KLV) encoding protocol shall comply with SMPTE ST 336. [defined in MISB ST 0107]
ST 0903.4-04 (Deprecated)	The number of bytes used to encode a variable length unsigned integer value shall be less than or equal to the specified maximum length.
ST 0903.4-05 (Deprecated)	The number of bytes used to encode the value zero for a variable length unsigned integer value shall be one (1). [replaced by req ST 0107.3-10]
ST 0903.4-06 (Deprecated)	The Series type shall be a one-dimensional array of data elements, all of the same type, encoded as a Variable Length Pack. [definition]
ST 0903.4-07 (Deprecated)	VTargetSeries shall be a Series of VTarget Packs. [definition]
ST 0903.4-08 (Deprecated)	Each VTarget Pack in a VTargetSeries shall contain metadata for a single target. [definition]
ST 0903.4-09 (Deprecated)	The first data in the value field of a VTarget Pack shall be a BER-OID-encoded value that represents the Target ID of a target. [definition]
ST 0903.4-11 (Deprecated)	All data within a VTarget Pack, excluding the Target ID Number, shall be items encoded as Tag-Length-Value items. [definition]
ST 0903.4-12 (Deprecated)	The Length field of a VTarget Pack shall be BER-encoded. [definition]
ST 0903.4-16 (Deprecated)	The Checksum shall be a 16-bit sum of all bytes in the Local Set, starting with the first byte of the 16-byte Local Set Key, up to and including the last byte of the Length field of the Checksum itself. [not the intended meaning; replaced with -116]
ST 0903.4-20 (Deprecated)	Where an associated Motion Imagery stream width is different from the Motion Imagery on which the VMTI process operates, VMTI LS Frame Width (Tag 8) shall be specified.
ST 0903.4-21 (Deprecated)	Where there is no associated Motion Imagery stream, VMTI LS Frame Width (Tag 8) shall be specified.
ST 0903.4-22	Where the VMTI LS Frame Width (Tag 8) is present, the Frame Width shall be the value of the VMTI LS frameWidth.
ST 0903.4-23 (Deprecated)	Where the VMTI LS Frame Width (Tag 8) is not present, the Frame width shall be taken from the Motion Imagery Frame width.

ST 0903.4-24 (Deprecated)	Where the VMTI LS Source Sensor (Tag 10) is specified, it shall appear from time to time in the KLV stream as an item with Periodic Volatility.
ST 0903.4-25 (Deprecated)	Where the VMTI's Source Sensor (Tag 10) is specified, it shall be updated at the first opportunity following a detected change.
ST 0903.4-28 (Deprecated)	To the extent possible a VTarget Pack Target ID Number shall uniquely identify a given target. [not testable]
ST 0903.4-29 (Deprecated)	At least one representation of VTarget Pack Target Centroid Pixel Number (Tag 1) shall be present. The Target Centroid Pixel Number can be specified using either the data element Target Centroid Pixel Number (Tag 1) or the pair of data elements Target Centroid Pixel Row (Tag 19) and Target Centroid Pixel Column (Tag 20). [replaced by -117 & -118]
ST 0903.4-30 (Deprecated)	VTarget Pack Target Location Latitude Offset (Tag 10) shall only be present when the VMTI LS is embedded within a MISB ST 0601 LS.
ST 0903.4-31 (Deprecated)	VTarget Pack targetLocationOffsetLon (Item 11) shall only be present when using embedded-VMTI.
ST 0903.4-32 (Deprecated)	VTarget Pack boundingBoxTopLeftLatOffset (Item 13) shall only be present when using embedded-VMTI.
ST 0903.4-33 (Deprecated)	VTarget Pack boundingBoxTopLeftLonOffset (Item 14) shall only be present when using embedded-VMTI.
ST 0903.4-34 (Deprecated)	VTarget Pack boundingBoxBottomRightLatOffset (Item 15) shall only be present when using embedded-VMTI.
ST 0903.4-36 (Deprecated)	When the VMTI LS is not embedded within a MISB ST 0601 LS, VTarget Pack Target Location (Tag 17) shall be used instead of VTarget Pack Target Location Latitude Offset (Tag 10) and VTarget Pack Target Location Longitude Offset (Tag 11). [replaced with revision]
ST 0903.4-39 (Deprecated)	Coordinates used for VMask LS Polygon (Tag 1) and VMask LS Bit Mask (Tag 2) shall be specified using pixel numbers calculated with the equation Column + ((Row-1) x frame width)), where numbering commences at 1 from the left for Column and from the top for Row, and where frame width is the number of columns in the image. [replaced with req's 100 & 101]
ST 0903.4-41 (Deprecated)	Each pixel number specified in VMask LS Polygon (Tag 1) shall be encoded using the Length-Value construct of a Variable-Length Pack. [definition]
ST 0903.4-42 (Deprecated)	The polygon specified in VMask LS Polygon (Tag 1) shall be closed by connecting the last point to the first point. [definition]
ST 0903.4-43 (Deprecated)	Each run of pixels in VMask LS bitMaskSeries (Tag 2) that subtends a part of the target shall be encoded by specifying the number of the pixel at the start of the run and the number of pixels in the run. [definition]
ST 0903.4-44 (Deprecated)	The run length in VMask LS Bit Mask (Tag 2) shall be encoded using BER-Length encoding. [definition]
ST 0903.4-45 (Deprecated)	The ontology referred to by the URI of the VObject LS Ontology (Tag 1) item shall be expressed using the OWL Web Ontology Language.
ST 0903.4-46 (Deprecated)	The VObject LS Ontology (Tag 1) item shall appear prior to any appearance of the OntologyClass (Tag 2).
ST 0903.4-47 (Deprecated)	The VObject Ontology (Tag 1) item shall appear from time to time in the KLV stream as an item with Periodic Volatility.
ST 0903.4-48 (Deprecated)	The VObject LS ontologyClass (Tag 2) shall have a value taken from the ontology specified by VObject LS Ontology (Tag 1) that precedes it in the KLV stream.
ST 0903.4-49 (Deprecated)	The VFeature LS shall conform to ISO 19156 and related schemas.

ST 0903.4-50 (Deprecated)	The VFeature LS Schema (Tag 1) item shall be present prior to the appearance of a VFeature LS Feature (Tag 2) item.
ST 0903.4-51 (Deprecated)	The VFeature Schema (Tag 1) item shall appear from time to time in the KLV stream as an item with Periodic Volatility.
ST 0903.4-52 (Deprecated)	The VFeature LS SchemaFeature (Tag 2) shall conform to the schema specified by VFeature LS Schema (Tag 1).
ST 0903.4-53 (Deprecated)	The VTracker LS Track ID (Tag 1) shall be a 16-byte Universal Unique Identification (UUID) in accordance with ISO/IEC 9834-8. [definition]
ST 0903.4-54 (Deprecated)	VTracker LS BoundarySeries (Tag 5) vertices shall be ordered so that looking toward Earth center they spiral in a clockwise direction from lowest elevation to highest.
ST 0903.4-55 (Deprecated)	Where an instance of the VTracker LS is present, the number of track points specified by VTracker LS Number of Track Points (Tag 8) shall be at least one (1).
ST 0903.4-57 (Deprecated)	The VChip LS Image Type (Tag 1) item shall appear in the KLV stream prior to a VChip LS Image URI (Tag 2) or a VChip LS Embedded Image (Tag 3) item.
ST 0903.4-58 (Deprecated)	The VChip LS Image Type (Tag 1) item shall appear in the KLV stream with Periodic Volatility.
ST 0903.4-60	The type of a VChip LS Embedded Image (Tag 3) shall be that specified by the preceding VChip LS Image Type (Tag 1).
ST 0903.4-61 (Deprecated)	Defined-Length Truncation Packs shall be defined in accordance with MISB RP 0701. [definition]
ST 0903.4-64 (Deprecated)	The Location Structure shall be encoded as a Defined-Length Truncation Pack. [defined in document]
ST 0903.4-65 (Deprecated)	The Location Pack shall consist of up to three groups of information that include a Geospatial Coordinates triplet, a Standard Deviations triplet, and a Correlation Coefficients triplet, in that order. [definition]
ST 0903.4-67 (Deprecated)	The Geospatial Coordinates triplet of a Location Pack shall consist of values for Latitude, Longitude, and Height, in that order. [definition]
ST 0903.4-68 (Deprecated)	The Latitude, Longitude, and HAE values of the Geospatial Coordinates triplet of the Location Pack all shall use the WGS84 Ellipsoid as reference. [definition]
ST 0903.4-69 (Deprecated)	The HAE value in the Geospatial Coordinates triplet of the Location Pack shall be expressed as Height Above the Ellipsoid (HAE) in meters with respect to the WGS84 ellipsoid. [definition]
ST 0903.4-70 (Deprecated)	Where the Correlation Coefficients triplet of a Location data type is present, the Standard Deviations triplet shall also be present. [definition]
ST 0903.4-71 (Deprecated)	The Standard Deviations triplet of the Location Pack shall consist of values for the standard deviations of the values in the Geospatial Coordinates triplet with respect to the East-North-Up local coordinate system, specifically East, North, and Up, in that order. [definition]
ST 0903.4-72 (Deprecated)	The Correlation Coefficients triplet of the Location Pack shall consist of values for the pairwise correlation coefficients of the values in the Geospatial Coordinates triplet with respect to the East-North-Up local coordinate system, specifically East-to-North, East-to-Up, and North-to-Up, in that order. [definition]
ST 0903.4-73 (Deprecated)	The Velocity Structure shall be encoded as a Defined-Length Truncation Pack. [definition]
ST 0903.4-74 (Deprecated)	The Velocity pack shall consist of a Velocity Components triplet, a Standard Deviations triplet, and a Correlation Coefficients triplet, in that order. [definition]
ST 0903.4-76 (Deprecated)	The Velocity Components triplet of the Velocity pack shall consist of values for East, North, and Up, in that order.

ST 0903.4-77 (Deprecated)	The East, North, and Up values of the Velocity Components triplet of the Velocity pack shall be expressed using the East-North-Up coordinate system specified by a Location pack for the location of the moving object.
ST 0903.4-78 (Deprecated)	The East, North, and Up values in the Velocity Components triplet of the Velocity pack shall be expressed in meters per second.
ST 0903.4-79 (Deprecated)	When the Velocity pack Correlation Coefficients triplet is present, the Velocity pack Standard Deviations triplet shall also be present.
ST 0903.4-80 (Deprecated)	The Standard Deviations triplet of the Velocity Pack shall consist of values for the standard deviations of the East, North, and Up values in the Velocity Components triplet, in that order.
ST 0903.4-81 (Deprecated)	The Correlation Coefficients triplet of the Velocity pack shall consist of values for the pairwise correlation coefficients of the values in the Velocity Components triplet, specifically East-to-North, East-to-Up, and North-to-Up, in that order.
ST 0903.4-83 (Deprecated)	The Acceleration Structure shall be encoded as a Defined-Length Truncation Pack. [definition]
ST 0903.4-84 (Deprecated)	The Acceleration pack shall consist of up to three groups of information that include an Acceleration Components triplet, a Standard Deviations triplet, and a Correlation Coefficients triplet, in that order.
ST 0903.4-86 (Deprecated)	The Acceleration Components triplet of the Acceleration pack shall consist of values for East, North, and Up, in that order.
ST 0903.4-87 (Deprecated)	The East, North, and Up values in the Acceleration Components triplet of the Acceleration pack shall be expressed in meters per second squared.
ST 0903.4-88 (Deprecated)	The East, North, and Up values of the Acceleration Components triplet of the Acceleration pack shall be expressed using the East-North-Up coordinate system specified by a Location pack for the location of the moving object.
ST 0903.4-89 (Deprecated)	When the Acceleration pack Correlation Coefficients triplet is present, the Acceleration pack Standard Deviation triplet shall also be present.
ST 0903.4-90 (Deprecated)	The Standard Deviations triplet of the Acceleration pack shall consist of values for the standard deviations of the East, North, and Up values in the Acceleration Components triplet, in that order.
ST 0903.4-91 (Deprecated)	The Correlation Coefficients triplet of the Acceleration pack shall consist of values for the pairwise correlation coefficients of the values in the Acceleration Components triplet, specifically East-to-North, East-to-Up, and North-to-Up, in that order.
ST 0903.4-95	VTrackItem Motion Imagery ID (Tag 19) shall be a Motion Imagery Identification System Core Identifier that conforms to MISB ST 1204. [VTrack related]
ST 0903.4-96	VTrackItem Motion Imagery URL (Tag 24) shall be a Uniform Resource Locator (URL) that conforms to IETF RFC 3986. [VTrack related]
ST 0903.4-97 (Deprecated)	The VTrack LS and VTrackItem Pack metadata listed in MISB ST 0903.4 shall be provided every time a track is reported or updated, unless the VTrack LS Track Status (Tag 4) indicates a status of “Dropped”, in which case the VTrackItem metadata elements may be omitted.
ST 0903.5-109 (Deprecated)	The Velocity Pack shall consist of up to three groups of information that include a Velocity Components triplet, a Standard Deviations triplet, and a Correlation Coefficients triplet, in that order.
ST 0903.5-110 (Deprecated)	The VTrack LS Checksum Tag 1 shall be present.
ST 0903.5-111 (Deprecated)	The VTrack LS Track Timestamp Tag 2 shall be present.

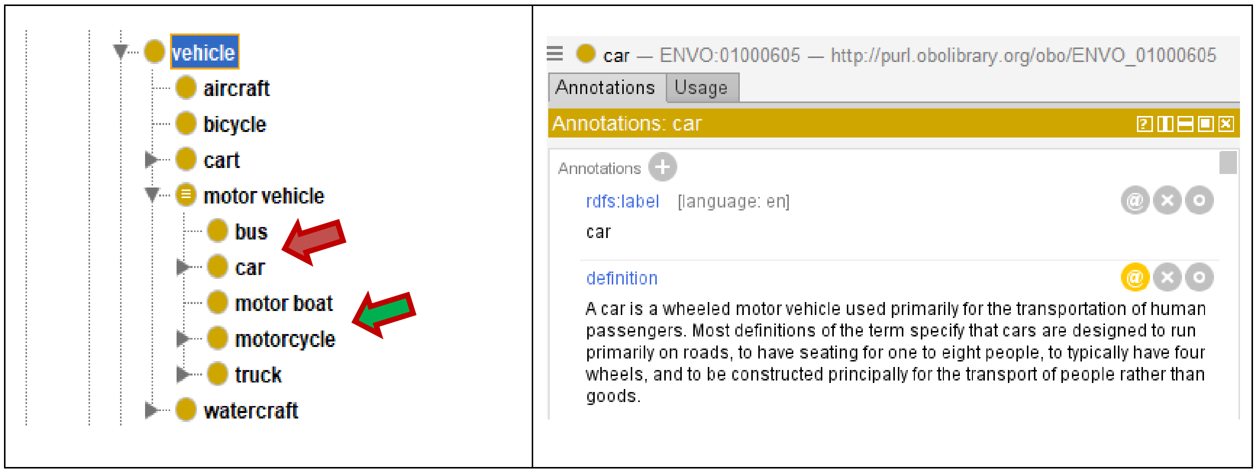
ST 0903.5-112 (Deprecated)	The VTrack LS Track ID Tag 3 shall be present.
ST 0903.5-113 (Deprecated)	Where VTrack Track Status Tag 4 is not “Dropped”, VtrackItem Target ID Number shall be present.
ST 0903.5-114 (Deprecated)	Where VTrack Track Status Tag 4 is not “Dropped”, VtrackItem Target Timestamp Tag 1 shall be present.
ST 0903.5-115 (Deprecated)	Where VTrack Track Status Tag 4 is not “Dropped”, VtrackItem Target Location Tag 13 shall be present.

Appendix A Ontology Exemplar [Informative]

The biological/biomedical field has developed the Open Biological and Biomedical Ontology (OBO) Foundry which is a list of ontologies. These ontologies enable use of common terms, labels of those terms, and other relationships between the terms. By doing so, ontologies enable easier data integration and prevent misinterpretation and associated errors. Ontologies are more than a “dictionary” or “thesaurus” and they have many benefits which are beyond the scope of this document. See [12] and [13] for more information.

This standard uses ontology references to associate items within an ontology to target objects. There are two parts in an ontology reference: the ontology identifier, and the entity identifier. The ontology identifier is an Internationalized Resource Identifier (IRI) which names the ontology where the definition of the entity resides. The entity identifier, also an IRI, identifies the desired label within the ontology.

For example, the OBO Foundry includes an Environmental Ontology (envo), with an identifier of http://purl.obolibrary.org/obo/envo.owl. The envo ontology includes entities for vehicle, car, and motorcycle (see example in Section 7.3). The left side of Figure 28 shows a subset of the envo ontology (using visualization tool) which illustrates the relationship of the three items. Every car (red arrow) is a type of motor vehicle, which is a type of vehicle. Likewise, every motorcycle (green arrow) is a type of motor vehicle. The right side of Figure 28 shows the definition for the entity car, along with its identifier (http://purl.obolibrary.org/obo/ENVO_01000605).



A car is a wheeled motor vehicle used primarily for the transportation of human passengers. Most definitions of the term specify that cars are designed to run primarily on roads, to have seating for one to eight people, to typically have four wheels, and to be constructed principally for the transport of people rather than goods.

Figure 28: Ontology Example

The two parts to a classification label are the ontology IRI and the entity IRI. For the car example:

ontology IRI = http://purl.obolibrary.org/obo/envo.owl
entity IRI = http://purl.obolibrary.org/obo/ENVO_01000605

Recognizing the entity IRI is not necessarily human readable/understandable, VMTI systems may additionally include the entity name. For the above example:

label = car

Optionally, ontologies may have versioning information that VMTI systems may include as another IRI. In this example ontology, the version IRI is:

version IRI = http://purl.obolibrary.org/obo/envo/releases/2021-05-14/envo.owl

Appendix B Operational Considerations [Informative]

B.1 Bandwidth

Bandwidth management is important in VMTI (and tracking) systems since they can produce a large quantity of dynamic data at frame rate. In its simplest implementation the VMTI LS may send thousands of targets to downstream systems by providing a targetId and a targetCentroid within a Motion Imagery frame for each target. At the other end of the scale, the VMTI LS has scope to include multiple features about each target, image chips of the target, tracking information about the target, and numerous descriptive elements. The bandwidth overhead required to include all this information is very large – especially at 60 frames per second (FPS) or higher.

To reduce bandwidth, data that changes infrequently does not need updating every frame but only often enough to ensure it is available in any clip extracted from the Motion Imagery; that is, deliver dynamic data at a rate appropriate to the granularity of the intelligence provided. For example, for a target moving at 3 meters per second, a rate of 60 updates a second provides very little value over a rate of 20 updates per second. In that case, while the Motion Imagery frame rate might be 60 FPS, a VMTI update rate of 20 FPS suffices.

The VMTI LS includes data that may not be available from onboard VMTI processors; thus, downstream processes will contribute “value add” elements to a basic VMTI stream. Calculations for required bandwidth need to consider the additional data of VMTI versus the available bandwidth at a given stage in the workflow.

Bandwidth implications and bandwidth management is important in the design of systems generating VMTI data. Reducing metadata to a minimal effective configuration is good engineering. The VMTI LS can leverage metadata available in other sets, such as MISB ST 0601. Some metadata (e.g., vmtiSourceSensor, vmtiHorizontalFov and vmtiVerticalFov) in the VMTI LS are comparable to those in the MISB ST 0601 LS. They are necessary, however, when VMTI operates on different Motion Imagery from that described by and/or included with the ST 0601 data. Consider, for example, two bore-sighted sensors, where ST 0601 metadata describes Motion Imagery from one sensor, but ST 0903 metadata describes VMTI detections from the Motion Imagery of a different sensor. (Note that ST 0601 metadata allows only one VMTI LS instance, precluding the carriage within it of detections from multiple Motion Imagery essences.)

Each VMTI process operating for a given sensor requires its own individual VMTI LS. That is, a VMTI LS should never contain a mixture of moving targets detected by different sensors. (The vChipSeries for a given detection may, however, contain image chips from multiple sensors.)

B.2 Co-located Bore-sighted Sensors

A system with multiple bore-sighted imagers within a single turret may send a Motion Imagery stream from one camera with a given field of view synchronously with VMTI hits from other sensors with different fields of view. For example, a system containing a narrow field of view visible light sensor (EON), a wide field of view visible light sensor (EOW), and an infrared (IR) sensor may be transmitting a stream from the EON sensor and simultaneously include VMTI data from both the EOW sensor and the IR sensor. This scenario necessitates a separate VMTI stream consisting of distinct VMTI metadata for the IR and EOW cameras carried along with the EON Motion Imagery. VMTI streams should be distinguishable by the vmtiSourceSensor and the vmtiHorizontalFov items.

Sophisticated VMTI systems may use the same targetId to identify a common target detected by different sensors and retain the use of same targetId temporally (that is, from one detection to another). Also, downstream processes (e.g., trackers and fusion systems) may reassign the targetId to identify a common target.

B.3 Independent Sensors

Each independent sensor system requires a separate MISB ST 0601 LS. The extra elements required in the VMTI LS to support multiple non-bore-sighted sensors would disproportionally increase bandwidth requirements for inclusion within a single ST 0601 LS packet. Given that ST 0601 does not support such cases anyway, the most appropriate solution is to generate independent VMTI LS and ST 0601 metadata for these sensors.

B.4 Large Volume Motion Imagery

Large Volume Motion Imagery (LVMI) systems present a problem not normally encountered with “traditional” airborne Motion Imagery sensors. LVMI systems cover large geographic extents and can detect thousands of simultaneous moving objects over an area of several square kilometers. A standalone stream of VMTI information (independent of Motion Imagery essence) could describe the detected moving objects and cue analysts as to which objects to monitor actively and which to leave to automated processes. This information can task “spotlights” which specify regions of interest to send accompanying Motion Imagery, which reduces data bandwidth. These spotlights in turn can carry VMTI data for the moving objects within the scope of the spotlight imagery.

VMTI data for spotlight streams is akin to the single sensor paradigm in their packaging and processing. However, VMTI data for cueing takes a different approach. In this case there may be no associated Motion Imagery present. Cueing based on VMTI data is therefore self-contained, with no dependencies on other data streams.

Appendix C Backward Compatibility

MISB ST 0903.6 underwent considerable modifications to improve usability. This required tradeoffs between usability verses backward compatibility. Although the operational nature of the VMTI model is the same, the changes in the functionality of some elements in the model causes a disruption in backwards compatibility with prior versions. Backwards compatibility means a ST 0903.6 data reader/processor may not fully understand the data written by VMTI systems prior to 0903.6 (e.g., ST0903.5).

There are two main types of changes affecting backwards compatibility: Deprecation of LS items and the formalization of the target life cycle (see Section 6 and 7). Systems built to ST 0903.6 and later processing VMTI from pre 0903.6 systems will need to accommodate conversion to account for these changes.

Table 21 summarizes the backwards compatibility issues between ST 0903.5 and ST 0903.6 by examining the usage of certain items in ST 0903.5. If a VMTI system producing ST 0903.5 does not use any of the items in the first column, there will not be a backwards compatibility issue.

Table 21: ST 0903.5 and ST 0903.6 backwards compatibility issues

ST 0903.5 Usage	ST 0903.6 Usage
VTarget Item 6 (targetHistory)	The ST 0903.6 target lifecycle does not support reusing target identifiers. A ST 0903.5 system might utilize the VTarget Pack's targetHistory to indicate the reuse of a targetId (i.e., after using a targetId for a while, the system resets the targetHistory to zero to indicate a new target with the same identifier). When reading ST 0903.5 data with a ST 0903.6 reader, if VTarget targetHistory (Item 6) is set to zero create a new unused ST0903.6 targetId (Note: to avoid conflicts with future targets this value may have to be large).
VTarget Item 21 (fpaIndex)	ST 0903.6 does not support targets from different Focal Plane Arrays (FPAs). Use a separate VMTI metadata stream for each FPA.
VTarget Item 102 (vObject)	ST 0903.6 does not support single vObjects attached to each VTarget. When reading ST 0903.5 data with a ST 0903.6 reader, move or append the ST0903.5 single VObject to the ST0903.6 vObjectSeries (VTarget Item 107). Note, the VObject may require other changes noted in Table 21.
VTarget Item 103 (vFeature)	ST 0903.6 does not support single vFeatures assigned to each VTarget. When reading ST 0903.5 data with a ST 0903.6 reader, move or append the ST 0903.5 vFeature (Item 103) to a ST 0903.6 VObject vFeatureSeries (Item 5).
VObject Item 1 and Item 2 (ontology and ontologyClass)	ST 0903.6 supports defining object information only through references to the ontologySeries (VMTI Item 103). When reading ST 0903.5 data with a ST 0903.6 reader, create an Ontology LS (see Section 10.9) from the two ST 0903.5 items ontology and ontologyClass as follows: • The Ontology LS requires an ontologyId (Item 1) unique to all other Ontology LSs within the ontologySeries. • Set the ontologyIRI (Item 3) (ST 0903.6) to be the same as the VObject ontology (Item 1) (ST 0903.5). • Use the VObject ontologyClass (Item 2) (ST0903.5) value to search within the ontology to find the entity's IRI. Set the entityIRI (Item 4) (ST 0903.6) to the entity's IRI in the ontology.

	• Optionally, set the Ontology LS label (Item 6) to the ontology's label value for the given entityIRI. Note: the label is DIFFERENT from an ontologyClass full value; see Section 10.9.6 for details and requirements. • Populate the versionIRI (Item 5) (ST 0903.6) if the information is available. • After creating the new Ontology LS append it to the ontologySeries. • Set the VObject ontologyId (Item 3) to the new Ontology LS's ontologyId. If the Ontology LS already exists (i.e., for a different VObject), reuse the existing ontologyId.
VFeature Item 1 and Item 2 (schema and schemaFeature)	ST 0903.6 only supports defining feature information by using references to the ontologySeries (VMTI Item 103). VFeature Item 1 and Item 2 in ST 0903.5 are not well constrained, so they may be difficult to convert into ST0903.6. If a mapping between a schemaFeature and an ontology exists, use the following method for conversion. When reading ST 0903.5 data with a ST 0903.6 reader, create an Ontology LS (see Section 10.9) from the two items (schema and schemaFeature). • The Ontology LS requires an ontologyId unique to other Ontology LSs within the ontologySeries. • Set the ontologyIRI (Item 3) (ST 0903.6) to an ontology that provides the equivalent schemaFeature. • Set the entityIRI (Item 4) (ST 0903.6) to the entity's IRI for the VObject schemaFeature (Item 2) (ST 0903.5) value in the ontology. • Optionally, set the Ontology LS label (Item 6) to the ontology's label value for the given entityIRI. • Populate the versionIRI (Item 5) (ST 0903.6) if the information is available. • After creating the new Ontology LS append it to the ontologySeries. • Set the VObject ontologyId (Item 3) from the VFeature (Item 3) to the new Ontology LS's ontologyId. If the Ontology LS already exists (i.e., for a different VFeature), reuse the existing ontologyId.
VTracker Item 2 (detectionStatus)	ST 0903.6 does not support VTracker detectionStatus. The VTarget detectionStatus provides a combined detection and tracking status, but with different meanings (see Section 7.2). A recommended mapping is:

ST 0903.5 VTracker Status	ST 0903.6 VTarget Status
Inactive	Inactive
Active	Active-Moving
Stopped	Active-Stopped
Dropped	Dropped (see note below)

	Note: ST 0903.5 allows a target's track to be Dropped then restarted; the ST 0903.6 lifecycle does not allow Dropped targets to be reused. See Section 7.2 for discussion on a target's lifecycle.
VTracker Item 6 (algorithm)	ST 0903.6 only supports specifying algorithms by using references to an algorithmSeries (VMTI Item 102). When reading ST 0903.5 data with a ST 0903.6 reader, create an Algorithm LS (see Section 10.8) from VTracker algorithm (Item 6) and append it to an algorithmSeries. Set the VTracker algorithmId (Item 12) to the new Algorithm LS's algorithmId (Item 1). If the algorithm LS already exists (i.e., for a different VTracker), reuse the existing algorithmId.
VTracker Item 8 (numTrackPoints)	With ST 0903.6, use the number of elements in the trackHistorySeries (Item 9) to replace the VTracker numTrackPoints (Item 8) (ST0903.5).
VMTI Item 7 (motionImageryFrameNumber)	ST 0903.6 only supports the Precision Time Stamp for referencing frames. When reading ST 0903.5 data with a ST 0903.6 reader, if a VMTI precisionTimeStamp exists, ignore the motionImageryFrameNumber. If a ST 0903.5 VMTI precisionTimeStamp does NOT exist, copy the motionImageryFrameNumber into the VMTI precisionTimeStamp (ST 0903.6). The fabricated timestamp, although not an accurate timestamp, will appear in proximity to the MISP Time System's epoch (see Section 10.1.2) providing a Level 1 Timing System Capability (see the Motion Imagery Handbook), i.e., total ordering using an increasing timestamp for sequential packets. In effect, the timestamp acts as a counter.

Additionally, ST 0903.6 now invokes MISB ST 0107.5 for MISB's common requirements when encoding numeric values. This change means LS values must use the minimum number of bytes, i.e., the value does not include zero or sign extension padding bytes. Previous versions of ST 0903 allowed for padding up to the maximum size specified in the values definition. This change means encoders writing ST 0903.6 metadata must trim any padding off each value. To support backwards capability, ST 0903 readers need to read both values with padding and no padding.