VEHICULAR INTEGRATION FOR C4ISR/EW INTEROPERABILITY (VICTORY) ARCHITECTURE

Version A
April 2, 2019

Released by the VICTORY Standards Support Office
vsso@victory-standards.org

Distribution Statement A

Approved for public release. Distribution is unlimited.

TABLE OF CONTENTS

1 1. Scope 1
2 1.1 Acronyms and Abbreviations 2
3 1.2 Architecture Background and Terminology 5
4 1.3 VICTORY Compliance 8
5 1.3.1 Architecture Compliance and Conformance Concepts 8
6 1.3.2 Component Type Compliance and Conformance Concepts 9
7 1.3.3 VICTORY Compliance Statements 10
8 2. Introduction 11
9 2.1 Motivation for a New Architecture 11
10 2.2 VICTORY Goals 11
11 2.3 Overall Approach 13
12 3. VICTORY Architecture Overview 15
13 3.1 VICTORY Tenets 15
14 3.1.1 Data Bus-Centric Design 15
15 3.1.2 On-the-Wire Network Interfaces 16
16 3.1.3 Shared Hardware Components 17
17 3.1.4 Standard Open Interfaces and APIs 17
18 3.1.5 Shared PNT Services 17
19 3.1.6 Shared Information Assurance Components 18
20 3.2 VICTORY Data Bus Context 19
21 3.3 VICTORY Data Bus Functionality 21
22 3.3.1 Transport Data 21
23 3.3.2 Provide Information Assurance 21
24 3.3.3 Provide Shared Hardware Infrastructure 22
25 3.3.4 Provide Shared PNT 22
26 3.3.5 Enable Integration of Modular Systems 22
27 3.3.6 Enable Interfaces to Non-Modular Systems 22
28 4. VICTORY Component Types and Interfaces 24
29 5. Common Functions 30
30 5.1 Data Transport 30
31 5.2 Auto-Discovery 30

5.3 Health Reporting 31
5.4 Common Management Functions 31
5.5 Activity Marking 32
5.6 Security Logging 33
5.7 Incident Response 33
6. VICTORY Data Bus Architecture 35
6.1 Infrastructure 35
6.1.1 Switch 36
Switch Data Transport Interface 36
Switch Management Interface 36
6.1.2 Low Latency Switch 36
Low Latency Switch Data Transport Interface 36
6.1.3 Generic End Node 36
Generic End Node Data Transport Interface 37
Generic End Node Management Interface 37
6.1.4 Shared Processing Unit 37
Shared Processing Unit Data Transport Interface 38
Shared Processing Unit Management Interface 38
Shared Processing Unit Platform API 38
Direct Attached Storage Encryption Management Interface 39
6.1.5 Network Attached Storage 39
Network Attached Storage Device Interface 39
Network Attached Storage Management Interface 39
Network Attached Storage Encryption Management Interface 39
6.2 Shared PNT 40
6.2.1 Time Synchronization Service 40
Time Synchronization Service Data Interface 40
Time Synchronization Service Management Interface 40
6.2.2 Position, Orientation, and Direction of Travel Services 40
Position, Orientation, and Direction of Travel Service Data Interfaces 41
Position, Orientation, and Direction of Travel Service Management Interfaces 41
6.2.3 GPS Receiver Service 41
GPS Receiver Service Device Interface 41

GPS Receiver Service Management Interface 41
6.3 System Management 41
6.3.1 VDB Management Service 42
VDB System Management Interface 42
6.3.2 VICTORY System Configuration Description 43
VDB Configuration Description 43
Information Assurance Configuration Description 44
C4ISR/EW Systems Configuration Description 44
Platform Systems Configuration Description 44
6.4 Recording and Logging 44
6.4.1 Mission Recorder 44
Mission Recorder Data Interface 45
Mission Recorder Management Interface 45
6.4.2 Data Logger 45
Data Log Retrieval Interface 46
Data Logger Management Interface 46
7. Information Assurance Architecture 47
7.1 Access Control 47
7.1.1 Authentication Service 47
Authentication Management Interface 47
7.1.2 Policy Decision Service 47
Policy Decision Management Interface 48
7.1.3 Policy Enforcement Service 48
Policy Enforcement Management Interface 48
7.1.4 Policy Store Service 48
Policy Store Management Interface 48
7.1.5 Attribute Store Service 48
Attribute Store Management Interface 48
7.2 Data Integrity and Confidentiality 49
7.2.1 Data Signing Service 49
7.2.2 Signed Data Verification Service 49
7.2.3 Data at Rest Encryptor 49
Data at Rest Encryptor Management Interface 49

7.2.4 Inline Network Encryptor 49
Inline Network Encryptor Transport Interface 49
Inline Network Encryptor Management Interface 50
7.3 Host and Network Protection 50
7.3.1 Host-Based Antivirus 50
Host-Based Antivirus Management Interface 50
7.3.2 Host-Based Firewall 50
Host-Based Firewall Management Interface 51
7.3.3 Intrusion Detection and Prevention System 51
Intrusion Detection and Prevention System Management Interface 51
7.3.4 Network Firewall 51
Network Firewall Management Interface 51
8. C4ISR/EW Systems Architecture 53
8.1 Audio Communications 53
8.1.1 Legacy Intercom System 53
Legacy Intercom System Data Interface 53
Legacy Intercom System Management Interface 54
8.1.2 Legacy Radio 54
Legacy Radio Data Interface 54
Legacy Radio Management Interface 54
8.1.3 Digital Audio Source 54
Digital Audio Source Data Interface 54
Digital Audio Source Management Interface 55
8.1.4 Audio Announcement Source 55
Audio Announcement Source Data Interface 55
Audio Announcement Source Management Interface 55
8.2 Video and Imagery Situational Awareness 55
8.2.1 EO/IR Sensor 56
EO/IR Sensor Management Interface 56
8.2.2 EO/IR Encoder 56
EO/IR Encoder Data Interface 56
EO/IR Encoder Management Interface 57
8.2.3 Gimbal 57

Gimbal Data Interface 57
Gimbal Management Interface 57
8.3 Off-Vehicle Network Interfacing 57
8.3.1 Router 57
Router Data Interface 58
Router Management Interface 58
8.3.2 Network Radio 58
Network Radio Data Transport Interface 58
Network Radio Management Interface 58
8.3.3 Off-Vehicle Network Gateway 59
Off-Vehicle Network Gateway Data Interface 60
Off-Vehicle Network Gateway Management Interface 60
8.4 Modular RF Signal Resources 60
8.4.1 MORA Overview 60
MORA Low Latency Bus 61
MORA Top-Down Perspective 62
MORA Signal Resource 63
MORA Processing Resource 64
MORA Transitional Implementation 64
8.4.2 Modular RF Signal Resources Component Types 65
8.4.3 MORA Device 66
MORA Device Management Interface 67
8.4.4 MORA Signal Resource Manager 67
MORA Signal Resource Management Interface 67
8.4.5 MORA Signal Port Manager 67
8.5 Radio Frequency Processing 67
8.5.1 RF Direction Finding Engine 68
RF Direction Finding Engine Data Interface 69
RF Direction Finding Engine Management Interface 69
8.5.2 RF Geolocation Engine 69
RF Geolocation Engine Data Interface 69
RF Geolocation Engine Management Interface 70
8.5.3 RF Spectrum Sensor 70

RF Spectrum Sensor Data Interface 70
RF Spectrum Sensor Management Interface 70
8.5.4 RF Signal Detector 70
RF Signal Detector Data Interface 71
RF Signal Detector Management Interface 71
8.5.5 RF Signal Generator 71
RF Signal Generator Data Interface 71
RF Signal Generator Management Interface 71
8.5.6 Electronic Warfare System 71
Electronic Warfare System Data Interface 72
Electronic Warfare System Management Interface 72
9. Platform Systems Architecture 73
9.1 Vehicle Systems Interfacing 73
9.1.1 Automotive System 74
Automotive System Data Interface 74
Automotive System Management Interface 74
9.1.2 Power Distribution System 75
Power Distribution System Data Interface 75
Power Distribution System Management Interface 75
9.1.3 Mechanical Articulation Joint 75
Mechanical Articulation Joint Data Interface 76
Mechanical Articulation Joint Management Interface 76
9.2 Lethality Systems Interfacing 76
9.2.1 Weapon System 76
Weapon System Data Interface 77
Weapon System Management Interface 78
9.2.2 Targeting System 78
Targeting System Data Interface 79
Targeting System Management Interface 79
9.2.3 Remote Weapon Station 80
Remote Weapon Station Data Interface 81
Remote Weapon Station Management Interface 81
9.3 Protection Systems Interfacing 81

9.3.1 Threat Detection System 81
Threat Detection System Data Interface 82
Threat Detection System Management Interface 82
9.3.2 Active Protection System 82
Active Protection System Data Interface 82
Active Protection System Management Interface 82
9.4 Logistics Systems Interfacing 83
9.4.1 Vehicle Configuration Service 83
Vehicle Configuration Service Management Interface 83
9.4.2 Vehicle Health and Usage Monitoring Service 84
Vehicle Health and Usage Monitoring Service Data Interface 84
Vehicle Health and Usage Monitoring Service Management Interface 84
9.5 Training Systems Interfacing 84
9.5.1 Live Training Transformation Adapter 84
Live Training Transformation Adapter Data Interface 86
Live Training Transformation Adapter Management Interface 86
9.5.2 Lasing System 86
Lasing System Data Interface 87
Lasing System Management Interface 87
9.5.3 Laser Detection System 87
Laser Detection System Data Interface 88
Laser Detection System Management Interface 88
10. VICTORY Composition Guidelines 89

LIST OF FIGURES

Figure 1. Illustration of Architecture Compliance and Conformance 9
Figure 2. Open Standards Matrix 12
Figure 3. VICTORY Data Bus Context 19
Figure 4. VICTORY Data Bus Architecture Organization 26
Figure 5. Information Assurance Architecture Organization 27
Figure 6. C4ISR/EW Systems Architecture Organization 27
Figure 7. Platform Systems Architecture Organization 28
Figure 8. VICTORY Configuration Description Organization 28
Figure 9. MORA Top-Down Perspective 63
Figure 10. MORA Transitional RF System Top-Down Perspective 65

1. SCOPE

This document describes the Vehicular Integration for C4ISR/EW Interoperability (VICTORY) Architecture, which provides a framework for integrating Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) and Electronic Warfare (EW) systems on military ground vehicles, and interfacing with the vehicle systems. The scope of the VICTORY Architecture is within the vehicle platform.

This document describes the architecture, which consists of a set of system and component types, their interfaces, and architectural patterns through which those system and component types can be composed to create C4ISR/EW system designs on vehicular platforms. The architecture creates a conceptual framework, but does not describe the technical details of the design instantiations, systems, components, or interfaces. The technical details (e.g., specific technologies chosen to implement the types and interfaces) are described in the VICTORY Standard Specifications. In other words, the architecture identifies the conceptual entities (system and component types, interfaces) and how they are integrated (structures and patterns), and the standard specifications identify the technical attributes of some of the entities, namely their interfaces. The standard specifications documentation is developed and matured by a standards body, which consists of government stakeholders, manufacturers of components, systems, vehicles, and system integrators.

1.1 Acronyms and Abbreviations

Table 1. Acronyms and Abbreviations

Acronym	Definition
1PPS	1 Pulse Per Second
2D	Two-Dimensional
3D	Three-Dimensional
ABAC	Attribute Based Access Control
ACERT	Army Computer Emergency Response Team
ADC	Analog-to-Digital Converter
API	Application Programming Interface
APS	Active Protection System
AS	Attribute Store
BIT	Built-In Test
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
CAN	Controller Area Network
CBM	Condition Based Maintenance
CLOE	Common Logistics Operating Environment
CM	Configuration Management
CPU	Central Processing Unit
CREW	Counter Radio-controlled Electronic Warfare -or- Counter RCIED Electronic Warfare
CT	Component Type
DAC	Digital-to-Analog Converter
DAGR	Defense Advanced GPS Receiver
DAR	Data at Rest
DDS	Data Distribution Service
DF	Direction Finding
DFE	Direction Finding Engine
DIFCUE	Direct Fire/Indirect Fire CUE
DoD	Department of Defense
DSP	Digital Signal Processor
EO	Electro-Optical

Acronym	Definition
EO/IR	Electro-Optical and Infrared
EPLRS	Enhanced Position Location Reporting System
EW	Electronic Warfare
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
GFE	Government Furnished Equipment
GLE	Geolocation Engine
GPP	General Purpose Processor
GPU	Graphics Processing Unit
GPS	Global Positioning System
IA	Information Assurance
IDPS	Intrusion Detection and Prevention System
IMU	Inertial Measurement Unit
INE	Inline Network Encryptor
INS	Inertial Navigation System
IP	Internet Protocol
IR	Infrared
ITAS	Improved Target Acquisition System
IVN	In-Vehicle Network
LAN	Local Area Network
LCOP	Logistics Common Operating Picture
LOB	Line of Bearing
LRAS3	Long-Range Advanced Scout Surveillance System
LT2	Live Training Transformation
MAC	Media Access Control
MAPS	Modular Active Protection System
MDM	MORA Data Message
MILES	Multiple Integrated Laser Engagement System
ML2B	MORA Low Latency Bus
MLRF	Micro-Laser Rangefinder
MORA	Modular Open RF Architecture
NAS	Network Attached Storage

Acronym	Definition
NDA	Non-Disclosure Agreement
NTP	Network Time Protocol
OS	Operating System
OSA	Open Systems Architecture
OSI	Open System Interconnection
PAN	Personal Area Network
PNT	Position, Navigation, and Timing
PDS	Policy Decision Service
PDU	Power Distribution Unit
PES	Policy Enforcement Service
PHY	Physical Layer
PS	Policy Store
PSD	Power Spectral Density
PTP	Precision Time Protocol
QoS	Quality of Service
RCD	RF Conditioning and Distribution
RCIED	Radio-Controlled Improvised Explosive Device
RCM	Reliability Centered Maintenance
RF	Radio Frequency
RHD	Radiohead
RPG	Rocket-Propelled Grenade
RWS	Remote Weapon Station
SA	Situational Awareness
SDR	Software-Defined Radio
SoSE	System of Systems Engineering
SPU	Shared Processing Unit
STORM	Small Tactical Optical Rifle Mounted
SWaP	Size, Weight, and Power
SWaP-C	Size, Weight, and Power and Cost
TES	Tactical Engagement Simulation
TOW	Tube-launched Optically-tracked Wire-guided
TPE	Theater Provided Equipment

Acronym	Definition
TTL	Transistor-to-Transistor Logic -or- Transistor-Transistor Logic
UI	User Interface
USB	Universal Serial Bus
VDB	VICTORY Data Bus
VCL	VICTORY Configuration Language
VCL ID	VCL Instance Document
VHUMS	Vehicle Health and Usage Monitoring Service
VICTORY	Vehicular Integration for C4ISR/EW Interoperability
VISA	Video and Imagery Situational Awareness
VoIP	Voice over Internet Protocol
VRCU	Vehicular Remote Control Unit (VRCU)
VSSO	VICTORY Standards Support Office
WAN	Wide Area Network
WESS	Weapons Effects Simulation System
WMI	Warfighter Machine Interface
WPAN	Wireless Personal Area Network

1.2 Architecture Background and Terminology

A*rchitecture*, within the context of this document, is a conceptual framework consisting of a common terminology, a set of modular functional entities that interoperate through interfaces, and common structures, patterns, and rules to follow when instantiating and organizing the modular entities to form system designs.

More concisely, an architecture defines a *common glossary* (well-defined terminology), a *type system* (modules and interfaces), and *system composition guidelines* (common structures, patterns, and constraints).

The *VICTORY Architecture* has been developed specifically for integrating electronics systems in U.S. Army ground vehicles, and is therefore specific to the domain of electronics systems on military ground vehicles. The VICTORY Architecture type system consists of a set of *component types* and *interfaces* that are relevant to the kinds of electronic systems that are part of or are integrated with these vehicles.

Although some of the component types may provide functionality that is commonly implemented in either hardware or software, they should be thought of as conceptual as opposed to tangible items. Their groupings in the architecture description are not meant to imply the same breakdown and allocation of functions to physical and logical elements of a design. In other words, architectural components A and B, with interfaces a, b, and c, may be implemented in firmware on a single physical device in one design, and in software on multiple physical devices in another design. Those systems could be considered functionally equivalent as long as the interfaces are open and correctly implemented.

The VICTORY *Architecture* is not the same as a vehicle *design* or a vehicle system implementation. A *design* is an instantiation of the architecture. In a design, technical choices have been made to address the operational, functional, performance, and other types of constraints and requirements for the system being developed.

A design is a plan that describes how a system will be constructed to implement a specific set of functional and non-functional requirements. A design includes a set of design elements (physical and logical components with physical and logical interfaces), a system mapping (allocation of functionality to design elements), and a system organization (interrelation and interconnection of design elements). *Logical components* are usually realized in software or firmware, and implement a set of *features* defined by the architecture. Note that the features of the VICTORY Architecture correspond to component types, which organize functionality with sets of supporting interfaces. Logical components have logical interfaces to other logical components, and implement a set of architectural interfaces. These support communication and management-related functions such as configuration, control, status reporting, or fault management. *Physical components* and interfaces are hardware devices that can host the logical components and their physical interconnections. In a design, the architectural components and interfaces have been allocated to logical components and interfaces; and the logical components and interfaces have been allocated to physical components and interfaces. In this document, the term *component* is used to describe either a logical or physical component in the context of a system design or implementation. The term *artifact* is used to describe a tangible entity implementing one or more logical and/or physical components.

A *physical design* (or *hardware design*) is the physical view of a design. It includes a set of physical components with their physical interfaces connected. A physical design is analogous to a wiring diagram. In a physical design, the hardware components have been identified, choices have been made about technologies, the interfaces have been chosen, and the connections between interfaces have been defined.

A *logical design* (or *software design*) is the logical view of a design. It includes a set of logical components (software) with their logical interfaces (messaging paths and application programming interfaces [APIs]) connected. The logical components form applications, which provide the functionality of the system. In a logical design, software components have been identified, functionalities have been decided, and technology choices for implementation of the software components and interfaces have been made.

The mapping of logical components to physical components and logical interfaces (messages and APIs) to physical interfaces (wires, connections, and platform resources) is part of the design process, and is called the system *allocation*.

As stated above, within the context of this document, an architecture defines a *common glossary*, a *type system* (modules and interfaces), and *system composition guidelines* (common structures, patterns, and constraints). The modules, which are called component types in VICTORY, describe a set of functionality or behavior, but only in general terms, along with interfaces necessary to support that functionality. The architecture does not define how the functionality is to be implemented, identify the technical details of the interfaces, completely detail the behavior of the components, or constrain how functionality is to be mapped to physical devices or software components. The technical details of the component behaviors and interfaces are defined in part by *standard specifications*, which leave some options open to be chosen during the design process. The choices of hardware components, software components, performance, and overall system design are defined in the *system design specifications*. Some technical details are prescribed by the standard specifications, and others are left as choices for the designer of a particular system. The line demarking where the architecture and specifications stop and where the vehicle-specific design starts is important to the success of the architecture. If the component and interface standard specifications are too rigid (over constrained), there is less flexibility in designing systems. If they are too loose (under constrained) subsystems, and components developed independently against the specifications will not interoperate when integrated. The goal is for the standard specifications to be detailed and constraining enough that interoperability between systems and components on the vehicle is achieved in most cases, while allowing for flexibility for designers to support a wide range of system requirements.

Note: The VICTORY Architecture document concentrates on the first two elements of an architecture; the common glossary, and the type system. It defines terms and a common lexicon, and defines modular entities called component types, and their interfaces through which they interoperate. Although this architecture document does identify a commonly used architectural structure (the VICTORY Data Bus), and patterns that may be used when developing designs (e.g., the adapter pattern), it does not completely provide the third element of an architecture; the composition guidelines. Properly explaining implementation patterns, and in particular constraints (rules), necessarily requires providing detailed design examples, which the authors of this document determined should be put into a separate document. The VICTORY Implementation Guide document will be a companion to the VICTORY Architecture document. It will explain in detail the VICTORY composition guidelines (the third leg of an architecture), and provide general guidance to designers and implementers for how to use VICTORY Component Types to design and implement systems to meet different types of system requirements.

The architecture provides an abstract framework onto which standard specifications can be developed. The goals are for the architecture and standard specifications to be applicable for a

large variety of systems, and to reduce the difficulty of integration, reduce the amount of hardware, and to increase interoperability between systems.

In order to achieve these goals, it is necessary to have both an architecture and standard specifications. This document provides the definition of the VICTORY Architecture, whereas the standard specifications are provided in "VICTORY Standard Specifications."

1.3 VICTORY Compliance

There is no concept of overall *VICTORY compliance* or *conformance*. VICTORY defines sets of component types, each of which defines requirements for a set of interfaces. No component or system will implement all of the component types, as not all of them are relevant for a single system. For that reason, a system, design, or architecture cannot be said to be fully *compliant* or *conformant* with the VICTORY Architecture or Specifications.

Compliance is designed, claimed, tested, and documented at the component type level, based on the component types implemented in a particular system. VICTORY does not define a concept of conformance. The reasoning that VICTORY defines compliance at the component type level and does not define conformance is explained in this section.

1.3.1 Architecture Compliance and Conformance Concepts

Compliant and conformant are similar concepts, but there are important differences. There are also differences in the meaning when compliant and conformant are applied at the system and component type level.

Referring to Figure 1, *Architecture Compliance* implies that a system implements some, but not necessarily all of the features of an architecture, and that no features defined by the architecture are implemented in a different way. *Architecture Conformance* implies that a system implements all of the features of an architecture. *Architecture Full Conformance* means that not only are all features implemented, but no features are implemented that are not defined by the architecture.

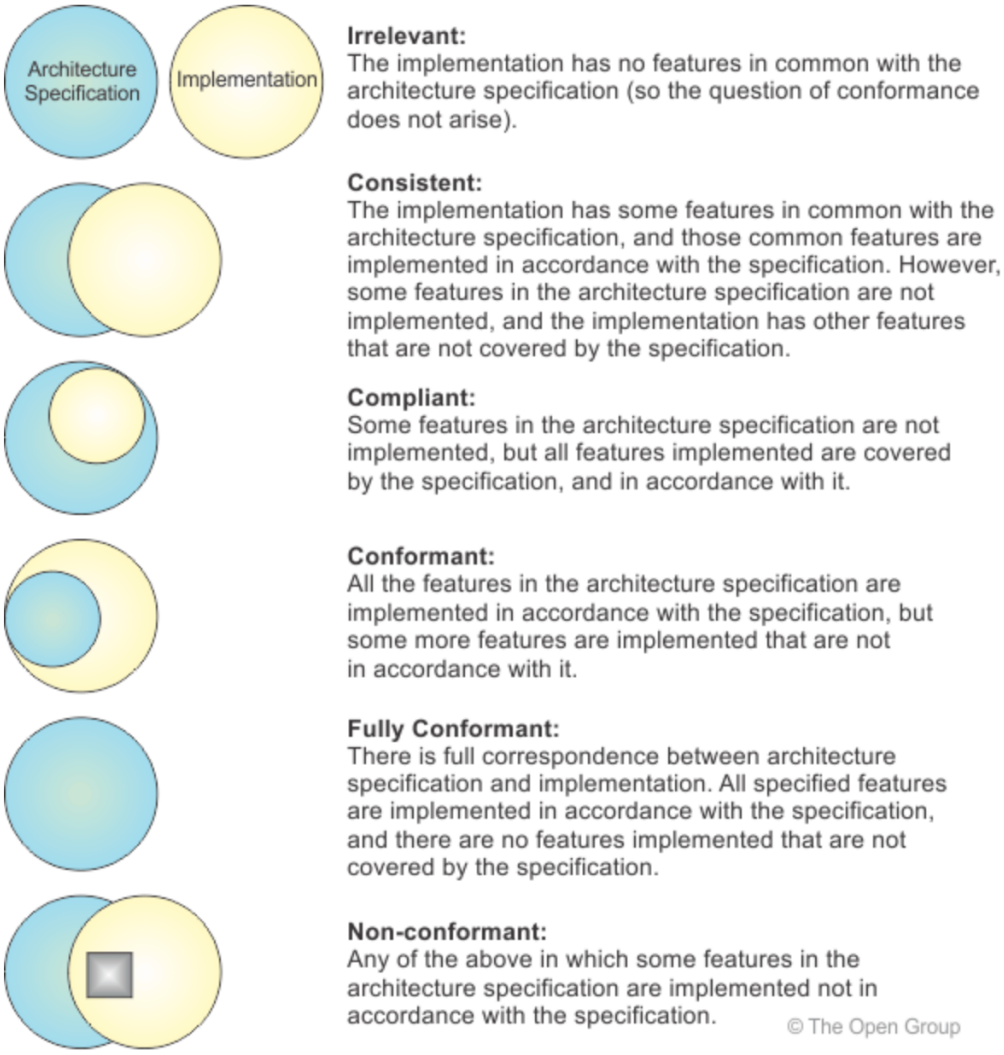

Figure 1. Illustration of Architecture Compliance and Conformance
Note: Used with permission, ©2011 The Open Group.

As mentioned above, it is highly unlikely that a system will be implemented to include all of the VICTORY Component Types (CTs). That is why VICTORY does not define a concept of system or architecture level compliance or conformance.

1.3.2 Component Type Compliance and Conformance Concepts

When the compliance and conformance concepts are applied at a component type level, as opposed to a full system architecture or design level, the differentiation is similar.

The VICTORY concept of compliance relates to component type specifications, analogous to an *architectural feature*, which is the term used in the text of Figure 1. In the VICTORY context, the terms *component* or *artifact* mean hardware and/or software that implements the functionality and interfaces of one or more component types.

An implemented artifact can be said to be *compliant* with a VICTORY CT if and only if all of the interfaces and features that are indicated as *required* in the CT specification are implemented, and pass a compliance test. VICTORY CT interfaces can be *optional* or *recommended*. An artifact can be compliant with the VICTORY CT specification if it does not implement the optional or recommended interfaces. However, if the artifact does implement those interfaces, they must also pass a compliance test for the implementation to be compliant with the CT specification. This is similarly true for features that are marked as optional or recommended within the interface specifications. If an artifact implements optional or recommended features, it must pass a compliance test for those features for it to be compliant with the CT.

VICTORY does not define CT-level conformance. If it did, saying an artifact is conformant to a CT specification would imply that all options and features were implemented and passed a conformance test, and that no features were implemented that are not defined by the specifications. That concept would be far too inflexible for the ground vehicle domain.

1.3.3 VICTORY Compliance Statements

To reiterate the most important concept, compliance with VICTORY specifications is stated, tested, and documented at the component type level only. Instead of saying a system is "VICTORY Compliant," the correct statement is that a system is "compliant with the following VICTORY Component Types: Position Service, etc." Whenever a compliance claim is made, it must be supported by a compliance test document for each component type for which the claim is made. The compliance test documents describe the tests that were performed and the results generated.

The objective is for the U.S. Army to benefit from the interoperability created by adopting the VICTORY Component Type specifications. It is understood that this will not happen overnight, but will require a gradual phasing in of open specifications into vehicle and product programs over time. The architecture has been developed with that type of phased-in approach in mind, and will provide increasing benefits as more of the component types are adopted.

2. INTRODUCTION

2.1 Motivation for a New Architecture

In most cases, current C4ISR/EW equipment on U.S. Army ground vehicles is developed as a set of stand-alone "Bolt On" systems that, at best, share only a common power source. These systems are developed separately and do not usually interoperate or share components. Instead, each new capability generally requires the installation of new hardware – sensors, processing units, displays, and user input devices. The objective of the VICTORY initiative is to define a network-based data bus architecture for integrating the electronics systems, enabling C4ISR/EW systems to share information, computational resources, and user interfaces (displays, controls, etc.) as a means of reducing overall vehicle size, weight, and power (SWaP) requirements.

Over the past decades, C4ISR/EW systems have primarily been integrated on military platforms using the "Bolt On" integration approach. Vehicles are procured with a set of base systems, and additional systems are procured in whole and fitted to the vehicles. Each system comes with its own processing resources, displays, and user interface devices, often creating duplication of common functions and a proliferation of hardware devices. This integration approach has become problematic and system performance has been adversely affected as the equipment on platforms has increased. As C4ISR/EW equipment has been added, crew compartments have become more cramped, available platform power has been exceeded, and maintenance requirements have increased. This has been caused by the duplication of hardware components (e.g., displays for each system, multiple input devices, and dedicated processing units). The "Bolt On" approach adds unnecessary weight, complexity, heat, volume, and power draw to the platform. The addition of multiple C4ISR/EW systems increases operator workload, and adds to both training and maintenance requirements. Warfighters must learn to use the interfaces of each C4ISR/EW system, and there is a significant increase in both recurring and non-recurring maintenance costs. The practice of "Bolt On" C4ISR/EW integration leads to sub-optimal platform integration solutions that limit the operational effectiveness of C4ISR/EW systems and their host platforms, increases operator and maintainer training and workloads, requires more SWaP, drives high integration cost, and results in dense vehicle crew compartments.

Furthermore, because most C4ISR/EW systems are procured separately, they are not designed to interoperate. They tend to operate independently, and data is not exchanged between the systems as a rule. This lack of interoperability results in unrealized capabilities that could be possible.

2.2 VICTORY Goals

The goals of developing the VICTORY Architecture and standard specifications are to:

- Improve upon current practices by eliminating, where possible, the practice of "Bolt On" systems.
- Reduce the SWaP and system cost impact of adding electronic systems to vehicles by utilizing shared hardware computing resources and using shared displays, as opposed to duplicating resources by dedicating displays to specific C4ISR/EW systems.

- Create a flexible framework for specifying the required functionality and interfaces of electronic components to simplify integration, enhance interoperability, create increased capabilities for the warfighter, and reduce life cycle costs.
- Maximize C4ISR/EW portability by defining 1) open standard interfaces, 2) open data formats, and 3) open protocols that can be used by vehicle communities.
- Support current and future IA requirements in U.S. Army ground vehicles. Document a set of shared IA hardware and software components, interfaces, and architectural patterns that enable a system integrator to build a "defense in depth" security design appropriate to a wide array of requirements and security levels.
- Provide an evolutionary approach towards network-centric C4ISR/EW, starting with interoperability with current systems, and providing a pathway for insertion of new capabilities and technologies.

The term "open standard" is often used loosely. Here, we define the meaning clearly to communicate the intent of the VICTORY Architecture. An open standard is a standard specification that 1) is not controlled by a single vendor (vendor-neutral), 2) is not tied to a specific vendor's platform (platform-independent), 3) is available for implementation by multiple sources, 4) is published, 5) may be used without royalties or the need for non-disclosure agreements (NDAs), and 6) is managed by a standards body.

Figure 2 provides a pictorial representation of what constitutes an open standard. At a minimum, for a given standard to be considered "open," it must be published (within the appropriate security context), royalty free, and with no contractual obligation to the developer or publisher of the standard.

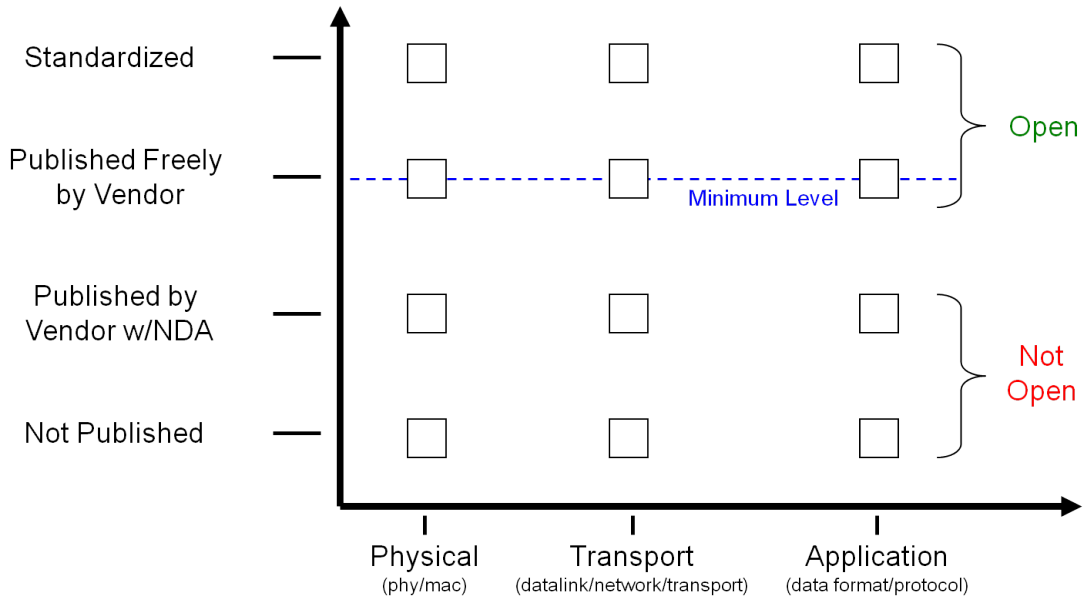

Figure 2. Open Standards Matrix

2.3 Overall Approach

The path forward that has been chosen for reducing the SWaP and cost impact of the electronics systems as capabilities are added is to combine functionality in shared computing resources. Earlier efforts have shared displays and computing functionality on vehicle platforms. In order to generalize this approach, and to define a framework for integrating C4ISR/EW systems in U.S. Army ground vehicles, an overall systems engineering effort was executed. The result is the VICTORY Architecture, which defines a network-based architecture that supports the addition of new capabilities to a vehicle without requiring as much additional hardware. A key architectural component is the shared processing unit, which is intended to host multiple C4ISR/EW applications and to provide common warfighter machine interfaces (WMIs). The shared processing unit is an enabler for reducing SWaP, as it allows systems to provide software components instead of always providing system-specific processing hardware.

The approach for increasing overall capability and to integrate between systems in a cost-effective manner is developing standard network-based system and component interfaces, and specifying that future systems utilize those standard interfaces. The main architectural structure that promotes interoperability is the VICTORY Data Bus (VDB), which provides network transport and shared data services. To make the transition from the current methods to a network and standards-based approach tenable and cost-effective, the architecture and standard specifications should take into account existing and emerging C4ISR/EW systems. For example, current systems without network interfaces can be integrated with VICTORY systems using "network adapters," which translate between the native and VICTORY Standard Interfaces. As existing systems are upgraded or evolved, the standard network interfaces can be implemented as part of the systems, reducing the need for network adapters.

Instead of integrating individual systems together one by one, or engineering an integrated solution for a single vehicle type (point solutions), it is advantageous to employ a System of Systems Engineering (SoSE) approach to identify concepts, systems, and components across a large set of vehicle types. The resulting core concepts can then be analyzed, and the common concepts used to form an architecture. This approach was taken in the development of the VICTORY Architecture.

The VICTORY architecture identifies the building block types and the interfaces that can be used to fit them together to create a vehicle network to integrate the electronic systems. The VICTORY Architecture does not specify or require specific hardware or software components, or specific ways of grouping or implementing the building blocks. The architecture also does not call out the technical details of the building blocks or their interfaces. That is the role of the VICTORY Standard Specifications document.

The role of the VICTORY Standard Specifications document is to provide enough technical detail for the building blocks and their interfaces to create interoperability, while not over-specifying, so that system integrators have the flexibility to meet varying functional and performance requirements.

Another way of looking at it is that the VICTORY Architecture identifies the conceptual building blocks, and the VICTORY Standard Specifications document provides enough technical details of those building blocks to ensure that they fit and work together.

3. VICTORY ARCHITECTURE OVERVIEW

The VICTORY Architecture has been developed as a solution to the problems discussed in the previous section. The development process follows a systems engineering analysis process to identify components and systems that are deployed across multiple vehicle types, and to identify functionality that is commonly implemented multiple times in a single vehicle design. These common types and functions provide the basis of the VICTORY Architecture.

3.1 VICTORY Tenets

Several fundamental principles, defined as *tenets*, form the core of the VICTORY Architecture. These tenets assert that electronics system designs on vehicle platforms should:

- Utilize a "data bus-centric" design.
- Leverage "on-the-wire" network interfaces.
- Provide hardware components that can be shared so that software components can be deployed without having to add hardware.
- Use only open standard physical and logical interfaces between systems and between C4ISR/EW components.
- Utilize a set of shared data services.
- Provide shared hardware and software IA component types to enable system integrators to build "defense in depth" security designs that adequately protect platform resources.

Without following these tenets, a design is not in line with the VICTORY Architecture.

3.1.1 Data Bus-Centric Design

The foundation of the VICTORY Architecture is the VDB. The VDB serves as the core integration mechanism, and the intent is for systems to utilize the VDB for integration with other systems on the platform. Each system that is compliant with VICTORY Component Types must provide VDB interfaces for data, configuration, control, status reporting, and fault management of those component types. The details of these interfaces are defined in the VICTORY Standard Specifications document.

Data bus-centric design implies that the most important interfaces in the architecture are "network interfaces."

The VDB is nominally based upon Internet Protocol (IP), and IP is considered the "convergence layer" between hardware and software components. Given the ubiquity of Ethernet, the VDB uses Ethernet for the physical layer (PHY), as well as the media access control (MAC), and the data link layer. The VICTORY Architecture does not preclude other technologies from being employed when the performance requirements cannot be met with Ethernet.

The following is a list of attributes that represent a "data bus-centric" design:

- Data from sensors, the platform, and systems is published on the VDB. In some cases, the publication on the VDB may be secondary to high-performance control or processing loops,

but the data should be accessible from nodes attached to the VDB (in accordance with security requirements).

- Tight coupling of sensors and their processing functionality should be avoided unless there is a performance requirement that would otherwise be unattainable.
- Data from other types of buses (e.g., controller area network [CAN]) should be published onto and accessed via the VDB.
- Configuration, control, status reporting, and fault management interfaces must be accessible via the VDB. This includes interfaces to the VDB components, C4ISR/EW systems, and platform systems.
- Message formats and protocols for all data, control, configuration, and status interfaces must be open (as defined earlier).
- Applications should access data, control, configuration, status reporting, and fault management interfaces via the VDB interfaces.

The purpose of a "data bus-centric" design is to ensure that the core architecture does not need to be modified based on the number of data sources, data sinks, or applications that are implemented on it. The architecture should scale up and down for high and low performance implementations and with the number of data sources and sinks. This approach avoids the geometric scaling problem (N*M) when the number of sources (N) and sinks (M) scales.

3.1.2 On-the-Wire Network Interfaces

VICTORY promotes interoperability between components through the definition of network interfaces that have "on-the-wire" specifications. The key concept driving interoperability between components implementing the VICTORY Component Type specifications is that the message formats and protocols that they use to communicate are standardized. The messaging protocols are held constant, allowing components that were developed using different technologies (varying hardware, firmware, software, programming language, etc.) and by independent organizations to be integrated more easily. On-the-wire network specifications drive inter-component interoperability at the system level, which addresses system-level integration cost and component reuse concerns.

Architectures that primarily define APIs, middleware, or transport independent data specifications are conceptually different than those that primarily define on-the-wire specifications. For reference, the Generic Vehicle Architecture defined by the United Kingdom Ministry of Defence is based primarily on a middleware approach. In middleware specifications, the primary concern driving interoperability is to reduce the cost of software through reuse. Middleware APIs can reduce the dependence of software on the underlying hardware architecture and operating environment. When the processing and communication architecture vary, the software components have a constant API to the operating environment, so are not required to change. API specifications tend to promote platform independent software, which addresses software reuse and cost concerns.

The key differentiator between on-the-wire network and middleware architectures are the primary concerns they address. Note that it is possible to address software reuse and system-level integration and component reuse concerns simultaneously by leveraging both approaches. That said, the primary concerns of VICTORY are promoting interoperability between system-level components, and reducing system integration effort.

The primary goal of VICTORY is to promote interoperability between components and systems to allow the government the choice in who provides systems and components, while achieving interoperability between them. A secondary goal of VICTORY is to reduce the technical risk (cost and schedule) of moving software components between computing platforms to allow the government to procure software components instead of pre-integrated "boxes." The on-the-wire network interfaces are key to VICTORY while the API specifications are important, but secondary.

3.1.3 Shared Hardware Components

Another core tenet of the VICTORY Architecture is the concept of shared hardware components. The purpose of shared hardware components is to reduce the number of hardware components on a platform. The VICTORY Architecture promotes the idea that processing, display, and user input hardware should be shared across multiple systems. To realize this goal, the VICTORY Architecture promotes the use of processing hardware that is able to host third party software applications that are developed to the appropriate application programming interface (within performance limits of the platform). The architecture also promotes the use of display devices that provide for third party WMIs and input devices that work with third party applications running on shared processing units (SPUs). The VICTORY Standard Specifications define standards to be used for sharing computing, display, and user input devices with multiple applications.

3.1.4 Standard Open Interfaces and APIs

VICTORY Architecture promotes the idea that not only the data and management interfaces be implemented on the VDB, but that those interfaces utilize standard open interfaces and APIs. Implementing network-based data and management interfaces that utilize the VDB lays a pathway toward interoperability between systems and components, and helps reach the VICTORY goals.

The technical details of the interfaces are developed by the VICTORY Standards Body. This body identifies and develops specifications for the VICTORY interfaces. These specifications are then matured using a defined maturation process, and transitioned into use as standards. It is a goal of the overall initiative to adopt existing standards if they already exist, and only develop new ones if that is not feasible. Once the specifications have been documented and matured into standards, new components and systems that are developed need to be compliant with the VICTORY standard specifications.

3.1.5 Shared PNT Services

A systems engineering analysis showed that many C4ISR/EW systems require access to Position, Navigation, and Timing (PNT) information. In some cases, the systems use a Global

Positioning System (GPS) receiver in order to obtain an accurate time source. Others require the position of the vehicle or its orientation or heading. For this reason, many systems include their own GPS receivers. The architecture identifies shared PNT services that allow systems, subsystems, and components attached to the VDB to be designed with the assumption that PNT data is available from the platform VDB. This reduces the need for duplicated processing hardware. The architecture also identifies services used for configuring, controlling, reporting status of, and managing faults in systems.

The shared data services include, but are not limited to:

- Time synchronization
- Position (i.e., location on Earth)
- Direction of Travel (also known as heading)
- Orientation (also known as attitude)

Time synchronization is important to the overall fidelity and accuracy of the integrated VICTORY approach. The system clock of each hardware component attached to the VDB should be synchronized so that applications running on the distributed processing units have the same notion of time.

Many C4ISR/EW and other systems on ground vehicles require knowledge of the position and direction of travel of the vehicle. Future applications may also require knowledge of the vehicle's orientation. Providing services on the VDB that implement a standard interface for obtaining position, direction of travel, and orientation information on the data bus allow a reduction in the number of devices (e.g., GPS, etc.) that are needed on a platform. A standard position interface provides vehicle latitude, longitude, and altitude. Similarly, a standard direction of travel interface provides the vehicle heading, and a standard orientation interface provides the vehicle's attitude (roll, pitch, and yaw). These interfaces are part of the data bus, making the data available to any system connected to the VDB.

To reduce the amount of time and effort necessary to configure, control, acquire and report status for, and manage faults on the data bus and its components, it is necessary to have a standard set of interfaces for data bus management. The data bus management interface provides standard network APIs for configuring, controlling, monitoring status, and managing faults of the VDB as an aggregate subsystem. Useful functions that would be supported by this interface include querying the current configuration of the data bus and individual components, updating the data bus configuration, controlling the state of the system and individual components, querying the top-level status, and receiving notifications of fault conditions of the data bus and individual components.

3.1.6 Shared Information Assurance Components

Since IA solutions are very difficult to implement retroactively, a goal of VICTORY is to treat IA as a "first class citizen" by identifying the components and architectural patterns that are necessary for implementing a wide array of IA solutions. This enables system designers to include IA solutions during the design phase. The IA solution to be applied in a particular system design is driven by the mission IA requirements.

3.2 VICTORY Data Bus Context

The VDB is the central structure of the VICTORY Architecture. An instance of a VDB provides a data transport infrastructure, IA facilities, shared hardware devices that can host software applications, a set of shared data services that include network time synchronization, as well as platform position, orientation, and direction of travel information, and management interfaces that can be used to configure, control, report status, and manage faults on the VDB and the attached systems.

Figure 3 provides the context of the VDB, and illustrates its interactions with various groups of C4ISR/EW and platform systems.

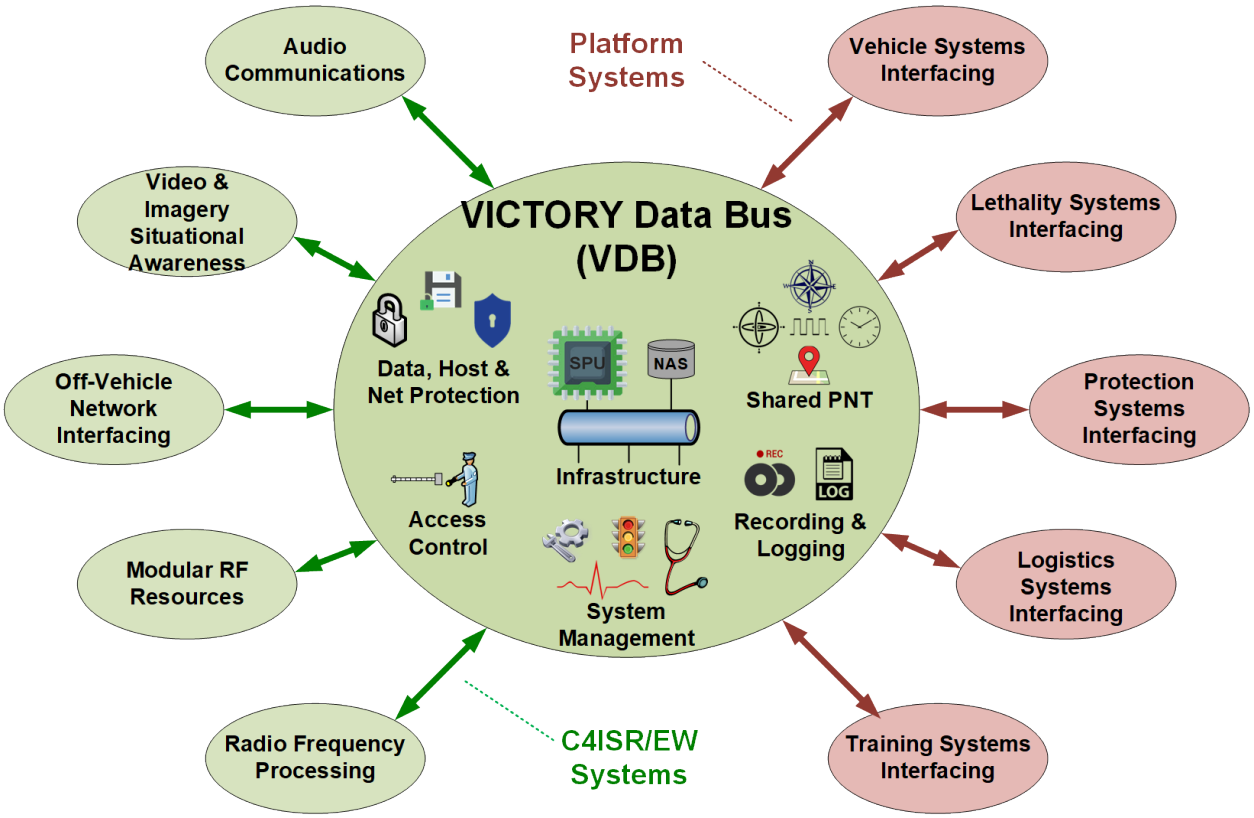

Figure 3. VICTORY Data Bus Context

The VDB *enables integration* of some systems on the vehicle. *Enables integration* means that the VDB transports data between the C4ISR/EW systems and also between the components of which the C4ISR/EW systems are composed. VICTORY defines interfaces at the system level and to the components of which these systems are composed. These can be identified in Figure 3 as they are in groups without the word "Systems" in the title.

The systems integrated by the VDB are made up of components that are modular, and interoperate through the VDB. The system components implement data and management (configuration, control, status reporting, and fault management) interfaces that are compliant

with VICTORY Standard Specifications. Those systems and their components utilize the services provided by the VDB including data transport, data protection, access control, shared data, and management. The VDB serves as the main transport mechanism for data shared among these systems, and also supports configuration, control, status reporting, and fault management of the systems. In cases in which performance requirements are difficult to support with the VDB, such as when sensing and processing must be tightly integrated, the data may be "published" on the VDB secondarily, outside of the tight performance loop, but data and management interfaces must be accessible from the VDB.

The VDB *enables system-level interfaces* to other kinds of systems on the ground vehicle platform, but does not define interfaces between the components of which these systems are composed. Instead it defines system-level interfaces with the VDB. The system-level interfaces can be identified in Figure 3, as they are connected to groups that have the word "Systems" in the title. VICTORY defines at least one interface to each of these systems, but their components are not necessarily integrated by the VDB. The VDB provides standard data and management interfaces at the system level so that these systems can be made to interoperate at least at a high level. The individual components of these systems do not utilize the VDB for primary data transport, nor are they managed from the VDB. As an example, the automotive control systems on ground vehicles are often built around CAN buses. The components of the automotive system include components such as sensors and engine control computers. The VDB does not define interfaces to the individual components of the vehicle powertrain control system. However, it does provide top-level data interfaces to the vehicle control system that can be used to publish data on the VDB related to the health and status of the vehicle systems. This data interface is secondary to the vehicle control system, and publishes data on the VDB that may be useful to other systems. However, there may be use cases where individual components of platform systems do have VDB interfaces. This can be beneficial, as some platform system components can be used for different needs during the system life-cycle. As an example, if a laser detector that is part of an active protection system has a VDB interface, an embedded training solution may be able to take advantage of these sensors as Multiple Integrated Laser Engagement System (MILES) detectors. For that reason, some platform system sensors have component-level interfaces in addition to system-level interfaces.

Figure 3 illustrates the relationship between the systems that are integrated by the VDB and those that interface with the VDB. The C4ISR/EW systems depicted on the left utilize the VDB as the data transport mechanism for implementing system functionality with individual components. The systems on the right merely interface with the VDB, and may have data, configuration, control, status reporting, and/or fault management interfaces so that the C4ISR/EW systems can interact with them. All systems and components that attach to the VDB systems may utilize the VDB shared data services, including time, position, orientation, and direction of travel services.

The following sections describe the VDB, the C4ISR/EW systems that it integrates, and the other systems with which the VDB interfaces. Several examples are provided showing how these

systems and components could be composed to create designs, taking into account requirements of the specific vehicle mission.

3.3 VICTORY Data Bus Functionality

The primary functions of the VDB are transporting data between the systems and components that are attached to the network, providing shared IA services, providing shared hardware resources, providing shared data services that can be accessed by any systems or components attached to the VDB, and providing services for managing the VDB and its components.

These functions can be used to integrate C4ISR/EW systems with the network, and thus increase the capability of these systems in support of the Warfighter. Integration allows new capabilities to be implemented that are more difficult to implement with stovepipe solutions. The standards-based architecture approach of the VDB allows increased capability to come with less of an increase in SWaP.

3.3.1 Transport Data

The main purpose of the VDB is to serve as the primary data transport mechanism for C4ISR/EW systems and components, as well as providing physical and application layer interfaces with other vehicular systems. The VDB delivers data from sources to sinks, while supporting various styles of transport, such as single and multiple destination addressing, dependable and non-dependable transport, and differentiated quality of service (QoS). Application data delivery, system management, and access control functions are implemented atop the VDB transport level functions. These functions are implemented by infrastructure, end node, and IA components of the VDB, which are described in later sections.

3.3.2 Provide Information Assurance

VICTORY defines component types that can be implemented in a VDB instance and configured to provide information assurance controls to protect the resources in accordance with, and at a level appropriate for, the specific system and mission requirements.

The Information Assurance capability group includes three sub-groups, including access control, data integrity and confidentiality, and host and network protection. These groups contain component types that can be configured to meet a wide variety of information assurance requirements. Information assurance controls also imply interfaces on other groups of component types. Specific requirements affect the overall design patterns as well as the individual component types.

These interfaces are available to control access to data and resources based upon entity attributes and policy, and to cryptographically protect data both at rest and in transit while on the platform or external to it. They can also be configured for end node protection and security event logging. These tools, and the component types and interfaces that comprise them, are designed to be flexible enough to support many different IA needs.

3.3.3 Provide Shared Hardware Infrastructure

The VDB provides shared hardware infrastructure that can support the implementation and integration of C4ISR/EW and other applications, as well as WMIs on vehicular systems. Shared hardware resources include network switches, shared processing units, and network attached storage, which can support the needs of multiple software applications, services and user interfaces. In many cases, this eliminates the need for dedicated hardware devices. Utilizing these shared hardware resources has the potential to reduce the SWaP of vehicular electronic systems. In order to make this feasible, the architecture and specifications define a set of open APIs that are used by software components that access shared hardware resources.

3.3.4 Provide Shared PNT

The VDB provides a set of services that provides PNT data to all network nodes attached to a VDB. The shared PNT services can be instantiated to share PNT data streams and management interfaces with systems attached to the VDB. The integrated PNT capability shares PNT data across systems that currently include their own PNT devices, resulting in duplication of functionality. Implementing the shared PNT capability is meant to promote the sharing of data from this set of sensors, reducing the need for duplicated sensors across systems.

To implement the shared data service function, the architecture identifies a set of shared data service interfaces. Service interfaces are described instead of the services themselves in order to avoid constraining the possible implementations of the services. For instance, a designer may choose to implement all of the shared data service interfaces in one software component that runs on an SPU. Alternatively, another designer may choose to implement separate software components for each interface so that the services can be distributed across several processing units. As the needs of a particular design should drive the implementation choices, the architecture and specifications define the service interfaces, not the services. The U.S. Army, or an organization within the U.S. Army, may choose to select a “common” implementation of the shared data services that implement these interfaces. This choice is outside of the scope of the VICTORY initiative, but the VICTORY Architecture and standard specifications provide the concepts upon which a common implementation can be based.

3.3.5 Enable Integration of Modular Systems

The VDB enables integration of modular systems by implementing data and management interfaces and by providing the primary data transport between their components. These systems and their components may also utilize shared data services, and may be hosted on the shared processing resources available on the VDB. Referring to Figure 3, note that VICTORY defines component types with VDB interfaces for Audio Communications, Video & Imagery Situational Awareness, Off-Vehicle Network, and Modular Radio Frequency (RF) Resources.

3.3.6 Enable Interfaces to Non-Modular Systems

The VDB also enables interfaces to systems that are not modular and are not integrated by VICTORY, but for which there is an advantage to interface at the system-level. These systems may benefit from access to the shared data services or data streams available on the VDB, and may also publish data on the VDB that can be leveraged by other systems. Referring to Figure 3,

note that VICTORY defines system-level VDB interfaces for vehicle, lethality, protection, logistics, training, EW, and intelligence systems.

4. VICTORY COMPONENT TYPES AND INTERFACES

The VICTORY Architecture is made up of sets of modular abstract entities called CTs, which interoperate through their interfaces. Note that the VICTORY Component Type is an abstract concept that corresponds to the modular functional entity concept that was described in the introduction. Note that a component is an element of a system design, while an artifact is a tangible implementation made up of one or more components (see earlier definitions).

Some component types are intended to provide "system-level" interoperability with systems that are not integrated by VICTORY, and thus only require high-level interfaces. Others are intended to provide interoperability at the component-level, and thus have lower-level interfaces.

For brevity, the term component type is used for both situations. The component-level and system-level component types can be differentiated by the titles of their capability groups. Groups with titles containing the word "system" contain system-level component types, while groups with titles without the word "system" contain component-level CTs.

The architecture identifies a core structure called the VICTORY Data Bus, which is a conceptual entity that is instantiated to provide the capabilities necessary to support the requirements of a particular system. The set of possible capabilities that may be provided by a VDB include: network, processing, and storage infrastructure; shared PNT services; system management; recording and logging services; access control; data integrity and confidentiality; and host and network protection.

A component type is a modular entity consisting of a set of functions that form a cohesive capability, along with a set of interfaces necessary to support the functions. Component types may be realized in hardware, software, firmware, or combinations, and can be combined together in single devices or software components when implemented. Some component types are inherently service oriented, in that their function is to implement network interfaces that publish data or provide information upon request. Other component types imply underlying hardware, such as a switch and shared processing unit (computer).

There are different kinds of interfaces that a component type may have. These include, but are not limited to:

- *Data Transport Interfaces* address communication functions from the transport layer and below, including the data formats, protocols, signaling, and perhaps physical media specifications.
- *High Volume or Low Latency Data Transport Interfaces* are data transport interfaces capable of delivering high volume data steams (e.g., high definition video), delivering data with low latency (end-to-end latency on the order of 1 microsecond), and/or providing higher levels of certainty and reliability than the normal data transport interfaces.
- *Data Interfaces* address the communication functions above the transport layer, up to the application layer, including data formats and protocols. Data interfaces utilize a publish-subscribe pattern that allows one or more publishers to share data periodically with zero or more subscribers. The data streams are organized into "channels," a concept similar to a "topic" in Data Distribution Service (DDS).
- *Data Logging and Retrieval Interfaces* allow data acquired from the network or generated internally to a component to be logged to a non-volatile medium, then retrieved in a standard format, and for that logging functionality to be managed from the network.
- *Auto-Discovery Interfaces* allow for components to be automatically detected on the network to support the concept of *plug and play*.
- *Health Reporting Interfaces* provide publish-subscribe style reporting of status and fault information.
- *Management Interfaces* address configuration, control, and health (status and fault reporting) functions for systems, components, and data publishing mechanisms. Management interfaces are based mainly on a request-response model, but also include event-driven notification behaviors.
- *Security Log Interfaces* provide a standard method for logging and retrieving data related to significant security events.
- *Access Control Interfaces* are used to control access to other interfaces, particularly the request-response style management interfaces.

The following sections walk through the component and system types that make up the VICTORY Architecture. The top-level organization is as follows:

- VICTORY Data Bus: Component types that make up the VDB, and the interfaces to be standardized.
- Information Assurance: Component types that protect data and control access to it, and the interfaces to be standardized.
- C4ISR/EW Systems: Component types that support C4ISR and EW functions, and the VDB interfaces to be standardized.
- Platform Systems: Component types that connect to systems that make up the platform, and the VDB interfaces to be standardized.

Figure 4, Figure 5, Figure 6, and Figure 7 provide hierarchical breakdowns of the VDB, Information Assurance, C4ISR/EW Systems, and Platform Systems architectures. This top-level breakdown was chosen based on commonality in function.

For each of these four architectural breakdowns, the CTs are grouped by "capability groups" that they support. Thus, it is correct to say that the VICTORY Architecture is decomposed functionally. However, it is important to understand that the hierarchical breakdown is for convenience and accessibility to the reader. The capability groups of CTs were chosen based on commonality of function, with the attempt being made to put CTs together that support a common capability. That said, this does not imply that a particular CT does not support multiple capabilities. This is an organizational hierarchy, not a "class hierarchy," so there are no functions, attributes, or interfaces that can be assumed as being inherited from the groups. Thus, the placement of CTs into capability groups does not imply commonality in content or the interface specifications, nor do the capability groups imply additional meaning to the functionality of the component types described herein.

Figure 8 describes the sections of the VICTORY Configuration Description, which is used to describe and/or prescribe the configuration of a particular VICTORY-enabled system. The VICTORY Configuration Description is realized in an XML-based configuration file that follows the VICTORY Configuration Language (VCL) schema. A particular configuration file is called a VICTORY Configuration Language Instance Document (VCL ID). Note that VCL ID is pronounced "VCL Instance Document." Each VICTORY Component Type that has a VCL configuration interface defines how its configuration is described and/or prescribed in a section of the VCL.

VICTORY Data Bus Architecture	Infrastructure	Switch
		Low Latency Switch
		Generic End Node
		Shared Processing Unit
		Network Attached Storage
	Shared PNT	Time Synchronization Service
		Position Service
		Orientation Service
		Direction of Travel Service
		GPS Receiver Service
	System Management	VDB Management Service
		VICTORY Configuration Description
	Recording and Logging	Mission Recorder
		Data logger

Figure 4. VICTORY Data Bus Architecture Organization

Information Assurance Architecture	Access Control	Authentication Service
		Attribute Store Service
		Policy Decision Service
		Policy Enforcement Service
		Policy Store Service
	Data Integrity and Confidentiality	Data Signing Service
		Signed Data Verification Service
		Data at Rest Encryptor
		Inline Network Encryptor
	Host and Network Protection	Host-Based Antivirus
		Host-Based Firewall
		Intrusion Detection and Prevention System
		Network Firewall

Figure 5. Information Assurance Architecture Organization

C4ISR/EW Systems Architecture	Audio Communications	Legacy Intercom System
		Legacy Radio
		Digital Audio Source
		Audio Announcement Source
	Video and Imagery Situational Awareness	EO/IR Sensor
		EO/IR Encoder
		Gimbal
	Off-Vehicle Network Interfacing	Router
		Network Radio
		Off-Vehicle Network Gateway
	Modular RF Signal Resources	MORA Device
		MORA Signal Resource Manager
		MORA Signal Port Manager
	RF Processing	RF Direction Finding Engine
		RF Geolocation Engine
		RF Spectrum Sensor
		RF Signal Detector
		RF Signal Generator
		Electronic Warfare System

Figure 6. C4ISR/EW Systems Architecture Organization

Platform Systems Architecture	Vehicle Systems Interfacing	Automotive System
		Power Distribution System
		Mechanical Articulation Joint
	Lethality Systems Intterfacing	Weapon System
		Targeting System
		Remote Weapon System
	Protection Systems Interfacing	Threat Detection System
		Active Protection System
	Logistics Systems Interfacing	Vehicle Configuration Service
		Vehicle Health and Usage Monitoring Service
	Training Systems Interfacing	Live Training and Transformation Adapter
		Lasing System
		Laser Detection System

Figure 7. Platform Systems Architecture Organization

VICTORY Configuration Description	VDB Configuration Description	Infrastructure Configuration
		Shared PNT Configuration
		System Management Configuration
		Recording and Logging Configuration
	Information Assurance Configuration Description	Access Control Configuration
		Data Integrity and Confidentiality Configuration
		Host and Network Protection Configuration
	C4ISR/EW Systems Configuration Description	Audio Communications Configuration
		Video and Imagery Situational Awareness Configuration
		Off-Vehicle Network Interfacing Configuration
		Modular RF Signal Resources Configuration
		RF Processing Systems Configuration
	Platform Systems Configuration Description	Vehicle Systems Intterfacing Configuration
		Lethality Systems Interfacing Configuration
		Protection Systems Interfacing Configuration
		Logistics Systems Interfacing Configuration
		Training Systems Interfacing Configuration

Figure 8. VICTORY Configuration Description Organization

Note that as of version A4 of the VICTORY Architecture, several of the component type names have been modified. The intent of these modifications is to promote clarity and to better communicate the intended functionality of those component types. The intent is not to modify the functionality of the component type, but to have the names related more closely to the intended functionality. In the case that the CT specifications have already been developed and matured, these name modifications do not necessarily require changes to the specifications or

compliance test artifacts. The updated names can be thought of as aliases for the existing component type names. For instance, the following statement is accurate: "The legacy intercom system component type has also been known as the intercom component type."

Table 2 lists all of the component types that have updated names as of the A4 version of the VICTORY Architecture. In each case, the functionality of the component type does not change. It is up to the VICTORY Standards Body to determine whether any actions are to be taken in the specification based on the naming updates. In general, the specifications may define the component type specifications using different (but related) naming schemes. The naming conventions in the architecture and specifications are conceptually coupled, but that coupling is flexible.

Table 2. Component Types with Updated Names

Original Name	Updated Name (alias)
Intercom	Legacy Intercom System
Streaming Audio Data Source	Digital Audio Source
Single-Instance Audio Data Source	Audio Announcement Source
Video or Image Sensor	EO/IR Sensor
Video or Image Encoder	EO/IR Encoder
Camera Gimbal	Gimbal
Data Radio	Network Radio
Extra-Vehicle Gateway	Off-Vehicle Network Gateway
Electronic Warfare Device	Electronic Warfare System
Articulation/Frame of Reference Sensor	Mechanical Articulation Joint
Threat Detection and Reporting Service	Threat Detection System
Laser Detector	Laser Detection System
Live Training Transformation Bridge	Live Training Transformation Adapter

5. COMMON FUNCTIONS

This section lists functionality that may be common across many of the component types that make up the VICTORY Architecture. Not all of these functions apply to every component type. The specification identifies the mapping between the component types and common functions.

5.1 Data Transport

The VDB must facilitate communications between many different systems, and as such, must be prepared to host a variety of network protocols that are utilized by these systems. For example, the VDB is responsible for providing both unicast (point-to-point) and multicast (point-to-multipoint) delivery and both dependable (with automatic failure detection and retries) and non-dependable (without automatic failure detection and retries) transport styles. It must also be able to support various kinds and levels of quality of service. Transport interfaces provide, as the name suggests, transport layer and below services. The data interfaces of the systems and applications attached to the VDB, leverages the VDB transport services, and provides additional higher layer data formats and protocols on a system-by-system basis. VDB data transport provides the basis for the system and application interfaces, so it is important that the transport provide a rich set of services.

The VDB transport functions are realized by the transport interfaces of the network components of which it is composed (e.g., switch, router, and firewall). The specification identifies a set of technologies that are available for implementation of the transport interfaces of the individual components, and which can be utilized by the data interfaces of service and application interfaces.

5.2 Auto-Discovery

Part of the philosophy of the VICTORY Architecture is to provide the ability to automatically discover the existence and type of components on the network. This concept is sometimes referred to loosely as “plug and play,” but this term can confuse the intent. Auto-discovery interfaces are provided for many of the component and system types that make up the VDB, the C4ISR/EW systems, and the platform systems types. The intent of these interfaces is to support dynamic discovery of components in appropriate settings, such as during development, integration, and early testing phases. Having the auto-discovery feature enabled during development, integration, and earlier test phases can be very useful to a program. It could ease integration of components developed by different organizations, and may also reduce the cost and time of configuring and integrating systems in non-operational settings.

This does not imply that the interfaces must be enabled in all operational settings, as security and predictability requirements may require that the system have a “static” configuration and that auto-discovery be disabled. The configuration of a system can always be “locked down” using a configuration file, allowing auto-discovery to be disabled, thus stabilizing the system behavior and performance.

Auto-discovery interfaces are intended to be supported by components implementing VICTORY Component Type specifications, but the architecture is flexible enough to allow them to be disabled to support the requirements of specific operational environments. Thus, auto-discovery interfaces are not required for all implementations.

Auto-discovery functions include the following:

- Announcing the existence of a component type or interface on the vehicle network
- Providing information describing how to communicate with the component type
- Describing the type of the component or interface
- Enabling and disabling the auto-discovery functionality

5.3 Health Reporting

Another philosophy of the VICTORY Architecture is to provide functions for obtaining the health of the system and its components in a standard way. The health reporting functions are applicable to most of the VDB, C4ISR/EW, and platform systems components, and specify how the components are to report status and faults on the VDB. These are publish-subscribe style interfaces that allow for health monitoring applications to gather and log health information, and possibly to automatically respond to fault conditions. Note that the publish-subscribe health reporting interface is supported by request-response interfaces for managing the publication. As with the auto-discovery interfaces, health reporting interfaces are intended to be supported by components implementing VICTORY Component Type specifications, but could be disabled to support the requirements of specific operational environments.

Health reporting functions include the following:

- Reporting and querying information about the status of a component type
- Reporting and querying information about faults that have occurred on a component type
- Handling acknowledgements for certain fault conditions

5.4 Common Management Functions

Most of the VICTORY Component and System Types include management interfaces to enable configuration, control, and health management (status and faults) of the individual components, and also the VDB as a whole. These are mostly request-response style interfaces, along with periodic and event-based reporting mechanisms. Although there are many different functions that are provided by the management interfaces of the various component and system types called out in the VICTORY Architecture, there is a common set that is used for most of them.

One common management interface, Version Information management, provides a common method of obtaining the type and version for any VICTORY Component type, and querying its capabilities and resources.

Other common management functions include:

- Request and update the configuration of the system and/or CT
- Configure and control data publishing interfaces
- Configure and control health publishing interfaces
- Request current health report
- Acknowledge faults

These common management interfaces provide a method of identifying components and their capabilities on the VDB, and allow for the management of data publication settings. Management interfaces are intended to be supported by components implementing VICTORY Component Type specifications, but could be disabled to support the requirements of specific operational environments.

The management interface specifications mostly define the service side of interfaces, but also require client-side interactions with other services in some cases. For example, the common management functions require a client-side interaction between the management service and the access control services in order to restrict access to potentially vulnerable interfaces to authorized entities. The specifications require that the management services be capable of interfacing as a client to the access control mechanism to check that the requesting entity has authorization before affecting any change on the service or returning any information to the client. In this case, a service definition specifies a client-side interaction to another service.

5.5 Activity Marking

The majority of VICTORY Component and System Types include publish-subscribe style data interfaces. The content of the data that is published by the data interfaces varies depending upon the component type, but there is a common set of attributes that may apply across many data interfaces. One such common attribute is *activity marking*. By marking the activity associated with data provided on the VDB, consumers of that data can easily and readily determine the nature of the data and the activity that originated it.

The activity marking concept includes identifying the *data nature* and originating activity. Data nature indicates whether the data is real (direct or derived product of a real/physical operational/tactical system, sensor, or device operating in its normal mode) or simulated (a direct or derived product of a live, virtual, or constructive simulation). Originating activity may indicate the type of activity whence the data originated, e.g., training event or exercise; operational test, evaluation, experiment, or exercise; or “real world” operational data.

In an effort to not unnecessarily constrain the design space of any particular vehicle design, it may be advantageous to enable a design in which a single instance of a component type can concurrently process and/or produce data originating from different activities. For example, during a virtual training exercise, a single platform sensor service may be actively monitoring the real/live environment while concurrently providing simulated data through its data interface as a result of modeling behavior based on simulated events and conditions in a synthetic

environment. In cases where the nature of data is real, the originating activity type may not always be discernible.

Activity marking functions include the following:

- Configuring and retrieving the *data nature* of a particular data stream
- Configuring and retrieving the *activity marking* of a particular data stream

5.6 Security Logging

Antivirus software, firewalls, intrusion detection and preventions systems (IDPSs), operating systems, networking equipment, and C4ISR/EW applications all generate security relevant logs. In order to make the security logging functions interoperable, it is necessary to standardize the logging format and the interfaces for configuring, controlling, and retrieving the logs. IA components and certain other component types may implement standard interfaces for logging and reporting significant security events. Security logging interfaces identify categories/classes of messages, which the working group standardizes, while providing standard structure for its content. There is also a need for a common security log management interface to ensure that security records are stored in sufficient detail and for an appropriate period of time. The security log management interface standardizes methods of generating, transmitting, storing, analyzing, and disposing of security log data.

Log management interoperability is accomplished through additional interfaces added to many of the VICTORY Component Types.

Security logging functions include the following:

- Logging significant security events
- Managing security logs
    - Storing, analyzing, and deleting security log data
- Retrieving security logs
    - Providing logs in a standard structure and format

5.7 Incident Response

Incident response involves identifying affected entities and performing the appropriate containment, eradication, and recovery actions. Protection of the VICTORY network requires the ability to detect and analyze incidents rapidly to minimize the number of infected components and/or hosts and the amount of damage the VICTORY Data Bus sustains. The response mechanisms must be simple and straightforward as there are not normally personnel assigned to the vehicles with training to perform these functions.

The incident response capability may utilize log information from the security log interfaces to identify potential cyber security incidents.

Incident containment, eradication, and recovery must be done in accordance with Army policy, and generally require close coordination with the Army Computer Emergency Response Team (ACERT). Therefore, an automated response is currently not possible.

Incident response functions include the following:

- Responding to security incidents
    - Identifying affected entities
    - Initiating appropriate containment
    - Executing eradication and recovery actions
- Configuring, controlling, and managing the health of the security logging mechanisms

6. VICTORY DATA BUS ARCHITECTURE

The VDB is the core structure of the VICTORY Architecture. It consists of a set of component types, and each component type consists of a set of interfaces. The functions of the VDB include network infrastructure (data transport), shared processing, management, shared data, access control, and data protection. These functions are implemented by the interfaces of the component types that make up the VDB.

The VICTORY Data Bus represents a standard structure and set of interfaces that support the basic functions required for an in-vehicle network (IVN). An IVN that implements the VDB standard interfaces is said to be an instance of a VDB.

This section describes the capability groups and component types that make up the VDB architecture.

6.1 Infrastructure

The Infrastructure capability group includes the core functionality of the VDB, which includes the transport, routing, and delivery of data with QoS on the VDB from data sources to data sinks; shared processing and storage resources; and basic transport and management interfaces to all end nodes that allow the overall system to be managed in a holistic fashion.

Transport, routing, and delivery of data includes encapsulating data messages into the appropriate network packages, transmitting the packets, routing the network packets through the network to the appropriate network segments and nodes, and receiving and unpacking the message data at the network nodes that are sinking the data. In a typical routed switching network configuration, each end node is directly connected to the port of a switch (or VDB component that provides switching functionality), which may queue the data before transmitting the data packets to a subset of its other ports. The packages transit the network by hopping from port to port and are eventually received and unpacked by the end nodes that are to sink the data. There are many options in this process that affect the way messages are addressed, whether the connections are point-to-point or connectionless, whether the connections are reliable, and which packages are to be dropped if there is contention for network resources (otherwise known as differentiated services, or QoS).

The VDB is currently based on IP/Ethernet networking technology due to its ubiquitous nature and increasing performance. IP implements the physical, data link, and network layers of the open system interconnection (OSI) reference model. The transport and higher layer protocols can be chosen depending upon the needs of the applications and logical components (e.g., shared data services). New networking technologies are constantly being introduced, and these technologies will eventually provide better performance than IP/Ethernet. VICTORY will continue to evaluate and incorporate appropriate network technologies as they become available. However, for the purposes of this document, IP is to be considered the “convergence layer” for the VDB. It is to be understood that the physical and media-access layer interfaces could be modified in the future, but that IP will remain the core of the VDB standards.

The infrastructure component types include switch, low latency switch, generic end node, shared processing unit, and Network attached storage.

6.1.1 Switch

The function of the switch component type is to receive messages from its ports, queue the data, and transmit the messages to the appropriate subset of its ports. The various options for how, when, and which data is queued, dropped, and transmitted to which ports are controlled by the addressing, routing, and QoS protocols being implemented.

Switch Data Transport Interface

The switch must implement a wide variety of protocols to facilitate the transport of data across the VDB. This interface mostly includes the protocols from the transport layer down.

Switch Management Interface

The switch management interface provides functions for configuring, controlling, and managing the health (status and faults) of the switch.

The configuration of the switch can be queried and updated via this logical interface. Settings such as IP subnet, static routes, and multicast functionality are essential to the transport of data on the VDB. Certain functionality implemented by the switch can be altered via the interface with the VDB, including port activation and power cycling. The status of the switch is available to authorized nodes on the VICTORY Data Bus. Examples of this status include information regarding which network nodes are attached to the switch, port, queue, and central processing unit (CPU) utilization, and data traffic patterns.

6.1.2 Low Latency Switch

The low latency switch component type has the same functionality and interfaces as the switch component type, but provides higher performance and QoS.

Low Latency Switch Data Transport Interface

The low latency switch data transport interface supports all of the functions of the switch data transport interface, but supports 4 to 10 times the bandwidth, lower latency (port to port latency on the order of 100 nanoseconds), and more reliable delivery than the normal switch.

6.1.3 Generic End Node

The generic end node is a term for a device that is attached to the VDB and does not fit into the other end-node categories.

The generic end node provides the basic functions necessary for a system to interface with the VDB transport and for the system to be managed by the VDB management system. These functions include data transport (the network stack) necessary to send or receive data on the VDB and a set of basic management functions.

An example is a device that has been designed to adapt one or more legacy components or systems to the VDB. Adapters are needed to integrate devices or systems that do not have native VICTORY interfaces (current force systems). Network adapters implement the VICTORY Standard Interfaces on one side; the native device interfaces on the other; and convert the physical, transport, and application layers of the native device, transform data structures into VICTORY Standard Interfaces, and visa-versa. As future components and systems adopt the VICTORY Standard Specifications, the need for network adapters will decrease over time.

Generic End Node Data Transport Interface

A generic end node has at least one data transport interface compliant with the VICTORY Standard Specifications. Implementing this transport interface allows components or systems to leverage the VDB transport and to be managed as part of the VDB, even when there are not yet relevant VICTORY Standards.

Generic End Node Management Interface

The generic end node management interface provides functions for basic control and health management of end nodes that are not one of the other VICTORY end node types (e.g., shared processing unit).

6.1.4 Shared Processing Unit

The SPU component type represents a computing platform (physical or virtualized processing resource) that can be accessed and utilized by multiple applications. Many software components are typically hosted on an SPU, sharing the processing, memory, and storage resources. An SPU hosts at least one VDB interface.

An SPU may include a display and user interface devices, may be headless, or may be attached to a passive (dumb) display. Applications can utilize the display, user input, and other peripheral capabilities of the device based on the access that has been provided to the software components it runs.

SPUs may host software applications that implement the VDB shared data service interfaces, C4ISR/EW, or other applications. SPUs may also provide display and other devices used to implement WMIs. Designers have significant flexibility when mapping logical components to the shared processing resources for implementing applications and WMIs. However, note that it is the responsibility of the system designers to ensure that the processing resources can support the processing needs of the applications and allocate system resources in a way that the overall performance requirements can be met.

The SPU component type includes data, management, and platform API interfaces.

The SPUs may also have non-networked interfaces to internal components, such as storage devices, user interface devices, and peripherals to which they connect with non-networked interfaces. Examples of such devices are analog displays (“dumb” displays), keyboards, and

pointing devices. These have direct connections to processing units, and generally do not have processing capabilities. Other devices, such as direct attached storage media may also be attached to processing units.

While these devices do not have networked interfaces, and do not interoperate with the VDB directly, it may be advantageous to identify and specify standards for interfaces to these devices to further the cause of device interoperability and commonality.

Shared Processing Unit Data Transport Interface

VICTORY does not define application layer data messages specific to the SPU data interface. VICTORY does define the SPU data transport interface to support the transport layer and below protocols necessary to provide network services to the SPU so it can interface to the VDB. The SPU hosts software components, each with their own data and management interfaces, which depend upon the SPU data transport interface.

Shared Processing Unit Management Interface

The interface for managing the SPU component type supports functions for configuration, control, and health management (status and faults).

The configuration functions of the management interface allow items to be set from the network, such as network interface settings and software configuration. The control functions allow for the system to be controlled from the network, including restarting the node, and bringing services up and down. The health management functions allow for status and fault information to be acquired from the network, such as network physical link status, CPU and memory usage, and processor over temperature conditions.

Shared Processing Unit Platform API

In order to allow for the hosting of several different software applications on the SPU, and to ease the difficulty of developing applications for multiple targets, a platform API is necessary. This platform API specifies interfaces to the shared resources, underlying runtime system, and to commonly used libraries. It also provides an API for accessing any display or user interface resources on the unit.

The SPU platform API should be developed to be as light as possible and, if at all possible, should avoid specifying hardware or operating system (OS) specific interfaces. It should not imply the use of specific runtime software that could cause undue performance overhead to the processing unit, drive up SWaP, and reduce the ability to scale systems down for very small applications.

Direct Attached Storage Encryption Management Interface

Storage devices are integrated with shared processing units and other infrastructure nodes. It is sometimes necessary to encrypt data on the direct attached storage devices for data at rest (DAR) protection. Encryption can secure data on storage devices when the devices leave the owners' control, such as when devices are repurposed, lost, returned for repair, or for warranty replacement. It can protect data from unauthorized discovery, lowering the risk of data exposure while preserving retention requirements and assuring information security. In the event that the storage device is lost, stolen, or tampered with, its data remains encrypted so no sensitive information is compromised. The DAR component type may also provide functions for emergency data destruction without destroying equipment.

The direct attached storage encryption management interface supports configuration, control, and health management (status and faults) functions for data encryption. The configuration function includes managing the encryption of the data at rest stored on the device.

The VICTORY Standards Body has specified and standardized the connection between the direct attached storage devices and the shared processing units. This interface includes device and data management functions, potentially covering low-level electronic connections, as well as encryption interfaces. The device interface to the direct attached storage component utilizes existing standard technologies for interfacing to the host device, transferring data, and for encrypting and decrypting data.

6.1.5 Network Attached Storage

The primary function of the Network Attached Storage (NAS) component type is to provide a repository for files accessible by multiple components attached to the VDB. The NAS may also have a data at rest protection function, which protects data with encryption. The NAS network interfaces may be used for reading and writing data, obtaining a list of contents, and managing the overall functions of the NAS.

Network Attached Storage Device Interface

The NAS device interface provides application layer functions for storing files and accessing files previously stored, and may provide a file system for organizing the files into directories.

Network Attached Storage Management Interface

This interface provides functions for configuring, controlling, and managing the health (status and faults) of the network attached storage.

Network Attached Storage Encryption Management Interface

The network attached storage encryption management interface supports configuration, control, and health management (status and faults) functions for data encryption. It is similar to the SPU direct attached storage encryption management interface.

6.2 Shared PNT

The Shared PNT capability group includes services that, if built into a VDB instance, make it possible for systems that are to be integrated to not require separate PNT devices, such as GPS receivers and inertial units. This will tend to reduce duplicated functionality and lower SWaP.

6.2.1 Time Synchronization Service

The VDB time synchronization service component type provides time synchronization to devices connected to the VDB network. The service sends messages over the VDB with which the various systems synchronize their clocks. This is a critical capability of the network, and the C4ISR/EW systems may need to utilize this common time for interacting with other systems in a coordinated manner. The service relies on a network time reference that, in most cases, is synchronized with "global" time by a GPS receiver with 1 pulse per second (1PPS) signaling. A standard network-based interface is used to synchronize the time of network nodes with the network time reference.

Time Synchronization Service Data Interface

The time synchronization service data interface provides network time to all VDB network nodes, utilizing the VDB data transport interfaces. A common implementation method would be for time to be derived from a GPS receiver, then provided to the network. The nodes are synchronized by interoperating with the time synchronization service via data interfaces.

Time Synchronization Service Management Interface

The time synchronization service management interface may provide configuration functions to allow certain settings to be managed from the VDB network, such as the network time update frequency. The control functions may allow the mode of the time synchronization system to be changed via the VDB interface; for example, to activate and deactivate the service. The status functions may provide information such as whether the time source is synchronized with the external time reference.

6.2.2 Position, Orientation, and Direction of Travel Services

These shared data services are three separate component types in the architecture, but are described in this single section due to their similarity. Each of these component types includes a data interface, which publishes data to the VDB, and a management interface, which can be used to configure, control, and manage the health of the service.

The position service component type publishes data describing the physical location of the vehicle to the VDB. This information typically includes latitude, longitude, and altitude, and is usually obtained from a GPS receiver or an inertial navigation system (INS) unit.

The orientation service component type publishes data describing the orientation of the vehicle to the VDB. The orientation data is in terms of roll, pitch, and yaw relative to a reference coordinate system, and is typically obtained from an inertial measurement unit (IMU) or an INS.

The direction of travel service component type publishes data describing the direction the vehicle is currently moving (which may be different than the direction in which it is pointing) to the VDB. The direction of travel may be derived from a compass or from GPS data, although other methods are possible.

Position, Orientation, and Direction of Travel Service Data Interfaces

These service data interfaces publish the position, orientation, and direction of travel information to the VDB network; and nodes attached to the VDB can subscribe to and receive updates to the data.

Position, Orientation, and Direction of Travel Service Management Interfaces

Each of these shared service component types includes a management interface. Configuration and control functions allow authorized entities to configure the transport settings for publication, set the publication frequency, view the uncertainty of the position data, and enable and disable data publication. Health management functions allow nodes on the VDB to view the status of any faults generated by the service.

6.2.3 GPS Receiver Service

The core functionality of the GPS receiver service CT is to provide an open network interface for obtaining the status of the GPS device and its satellite connections, which can be useful in understanding the quality of the current values of the parameters derived from its signal (e.g., Position, Direction of Travel, Time). This service does not publish the PNT data. The other Shared PNT CTs are responsible for distributing PNT information on the VDB, not the GPS receiver service.

GPS Receiver Service Device Interface

The GPS receiver service device interface is normally a serial connection with a two-way messaging protocol. Multiple standards exist for the messaging language and for the connection. VICTORY does not standardize this interface. It is up to the GPS receiver service implementation to support the native device interface, as well as the VICTORY standard GPS receiver service interface.

GPS Receiver Service Management Interface

The GPS receiver service management interface provides functions for configuring, controlling, and managing the health (status and faults) of the GPS. Parameters include the number of satellites currently seen and whether the device is currently time synchronized with the satellites.

6.3 System Management

The VICTORY System Management capability group supports functions for configuring, controlling, and managing the health of the VDB infrastructure, all CT implementations, and the systems attached to the VDB.

6.3.1 VDB Management Service

The VDB management service component type provides the ability to configure, control, and manage the health (status and faults) of the VDB as a whole, its components, and the systems that it interfaces or integrates. The ability to configure and control a network (and components connected to the network) is extremely valuable from a systems engineering perspective. VICTORY provides the ability to not only control a VDB, but also to configure the VDB as a whole. The VDB Management Service supports monitoring and resetting a system during integration. It is understandable that system configuration would be disabled in an operational context, and VICTORY specifies the security constructs that would make this possible.

VDB System Management Interface

The VDB system management interface includes functions for configuring, controlling, and managing the health (status and faults) of the integrated electronics system.

The configuration function may allow querying or setting of configuration parameters of the network, its components, and systems interfacing the VDB. At the VDB level, the configuration function may provide the ability to adjust parameters for traffic shaping, QoS policies, allocation of services and applications to processing units, and transport settings for data streams.

The component types that make up a VDB provide configuration interfaces that can be used to update or request the current values of configuration parameters. The VDB system management interface provides configuration functions for updating or requesting the current configuration of the overall VDB, as well as the configuration of the individual components. The VDB system management interface utilizes a configuration file for describing the configuration of the VDB and the systems it integrates or interfaces. The configuration file follows the VICTORY configuration description component type specifications and is described in a later section.

The control function includes the ability to monitor and change the operating state and/or mode of the VDB and systems it integrates. For example, it may allow for the components of the VDB to be reset, or all applications running on the system to be shut down and restarted.

Note that it is clear that these kinds of control functions could be potentially misused, either unintentionally or otherwise. VICTORY addresses this by providing mechanisms for setting up and enforcing policies that restrict access to critical resources (such as control functions) to entities with the proper authorization. That mechanism is provided by the component types in the access control capability group of the information assurance architecture. These are defined in a later section.

The status reporting function may allow the monitoring and reporting of the status of the VDB and its components. Status information could include how many nodes are connected to the VDB at any given time, status of the overall system, and network bandwidth utilization.

The fault management function allows for the monitoring and reporting of faults on the VDB and its components. At the VDB level, it may be used to provide a top-level VDB system fault status with a green-, yellow-, and red-light condition indication. It may also indicate faults of individual components in the system that can then be polled for detailed fault information.

6.3.2 VICTORY System Configuration Description

To support management of the electronic systems that are integrated with a VDB on a vehicle, the architecture defines a set of components that are used to describe the system configuration. The function of the VICTORY configuration description is to document the design and configuration of a system.

The design description could include the breakdown of the system into software and hardware components, the connections between the VDB and sensors and platform systems, the allocation of software to hardware, and the mapping of component types and their interfaces to these hardware and software components.

The configuration description could include the system mode and operating state, as well as identifying whether inputs are simulated or real.

The configuration description could include the logical network structure, node and data stream addressing, mapping of software components to hardware components, routing and quality of service parameters, and information assurance configuration.

The VICTORY configuration description is realized by a language that includes syntactic and semantic definitions. Instances of the language, or specific system configuration descriptions, may be stored in files. The language may be useful in both descriptive (used to document the configuration), and prescriptive (used to affect changes to the configuration) capacities.

As is shown in Figure 8, the VICTORY configuration description has four major parts, which are described in the following sections.

The VICTORY configuration description includes additional information not discussed here.

<u>VDB Configuration Description</u>

The VDB configuration description includes concepts for identifying infrastructure components, their interfaces, and their configuration settings. The types of components covered include:

- Infrastructure Configuration
- Shared PNT Configuration
- System Management Configuration
- Recording and Logging Configuration

Information Assurance Configuration Description

The information assurance configuration description includes concepts for documenting the design and configuration of the security system. The types of components covered include:

- Access Control Configuration
- Data Integrity and Confidentiality Configuration
- Host and Network Protection Configuration

C4ISR/EW Systems Configuration Description

The C4ISR/EW systems configuration description includes concepts for documenting the design and configuration of the C4ISR and EW systems integrated with the VDB. The types of components covered include:

- Audio Communications Configuration
- Video and Imagery Situational Awareness Configuration
- Off-Vehicle Network Interfacing Configuration
- Modular RF Signal Resources Configuration
- RF Processing Configuration

Platform Systems Configuration Description

The platform systems configuration description includes concepts for documenting the design and configuration of the platform systems interfaced to the VDB. The types of components covered include:

- Vehicle Systems Interfacing Configuration
- Lethality Systems Interfacing Configuration
- Protection Systems Interfacing Configuration
- Logistics Systems Interfacing Configuration
- Training Systems Interfacing Configuration

6.4 Recording and Logging

The Recording and Logging capability group is self-descriptive. These components are responsible for the storage of particular types of network data for playback or for bulk retrieval at a later time. The hardware components that are used to support this capability can range widely, but typically involve a network-ready processing device, as well as a mass (non-volatile) storage device to retain the network data.

6.4.1 Mission Recorder

The functions of the mission recorder component type are to sink network messages, store the messages on a non-volatile medium, and to play back the messages onto the network, approximating the timing with which the messages were received. This recording and playback

functionality could be used to document and recreate sensor and other data streams generated in test, training, or operational scenarios.

Because the mission recording hardware varies depending upon mission objectives and platform limitations, the VICTORY Architecture aims to standardize only the network-facing interfaces included in the CT.

Mission Recorder Data Interface

The mission recorder data interface is used for playback of stored data packets on the VDB. The mission recorder must also have data interface clients to sink the data to be recorded from the VDB. Information to be retrieved and played back could include video, digital audio, and situational awareness (SA) data.

Mission Recorder Management Interface

The mission recorder management interface enables the recorder to be configured and controlled, and its health (status and faults) managed.

This includes setting up configuration items, such as which data is to be recorded on the network, and for how long this data should be stored in the component's storage device. Some functionalities implemented by the mission recorder component type may be altered via the control interface, such as stopping and restarting the mission recording services, as well as selecting which VICTORY nodes can request to have the stored data played back over the network. The status of the mission recording functions is available to authorized nodes on the VDB. Examples of this status could include the amount of storage capacity that remains in the mass storage device and the rate at which the device is recording network packets.

6.4.2 Data Logger

The Data Logger component type is similar to the mission recorder in that it receives and stores network traffic on a persistent medium. The difference is that instead of providing a "playback" function for recreating the message traffic as received, it stores the data in a "log," which can be retrieved in bulk.

The functions supported by the data logger include:

- Logging data
    - Receiving network messages, which are assumed to be multicast on the vehicle network
    - Storing messages on a persistent medium
- Retrieving data logs
    - Retrieving data (bulk data) upon request
        - Data may be provided as a string or a file

The data logger component type implements the pattern of interfaces common to many VICTORY CTs, including discovery and management (configuration, control, and health) interfaces. It does not implement a data interface, although it is a client to many of the VICTORY CT data interfaces. In addition, the data logger implements a *data log retrieval interface*, which is described in the following section.

Data Log Retrieval Interface

The data log retrieval interface serves requests to retrieve the log data in bulk. Client entities can request data from the logs in bulk format based on time periods. The interface provides the requested data as a network transfer, the duration of which will depend upon the performance of the network and the current resource utilization.

Data Logger Management Interface

The data logger management interface supports configuration, control, and health management of the data logger.

- Managing the configuration of the logging, including:
    - Storing, viewing, and updating the logging configuration
    - Configuration parameters include the following:
        - Data that is to be received and logged
        - Maximum log space
        - Policy that describes action(s) to be performed when stored log data exceeds a defined maximum space or storage space runs out
            - E.g., overwrite oldest, raise fault, stop logging but continue operation
    - Controlling the logging function
- Managing the health (status and faults) of the logging
    - Reporting or retrieving logging status. Status parameters include the following:
        - Current log space, space available, etc.
    - Reporting and handling logging faults. Faults include the following:
        - Log space at or near maximum, disk space at or near capacity, etc.
- Controlling the logging execution. Controls include the following:
    - Start/stop (immediate, and at specified times or duration)
    - Delete logs (with a time period provided)

7. INFORMATION ASSURANCE ARCHITECTURE

7.1 Access Control

VICTORY is an open systems architecture (OSA), which implies that the interface protocols may be available to a wide community. The open network interfaces provide power and flexibility, allowing functions such as configuration and control to be performed via common, standard protocols. However, adding this power would result in a potential vulnerability if they were not protected from unauthorized access. The VICTORY Architecture access control capability group can be used to protect potentially vulnerable interfaces from unauthorized access.

The access control architecture employs an attribute-based access control (ABAC) model, which can be used to implement a wide array of access control styles. The core concept is that entities, once authenticated, are provided access to resources based on policies, which describe the relationship between the attributes of entities and resources. The attributes and policies that are used to make and enforce decisions for entities accessing resources are managed through network-based management interfaces. Together, the access control services can be used in an implementation to provide single sign-on login access to multiple devices and software applications.

The access control component types include the authentication, policy decision, policy enforcement, policy store, and attribute store services.

7.1.1 Authentication Service

The function of the authentication service component type is to provide interfaces that can be used to authenticate entities (determine that the entity is actually the one that it is claiming to be). The authentication service component type manages entity credentials, which are used to verify a requestor's identity and establish a basic trust relationship. The authentication service component type may support many kinds of authentication, such as username and password, certificate, and biometrics-based approaches.

<u>Authentication Management Interface</u>

The authentication management interface provides functions for managing the health (status and faults), configuring, and controlling the authentication service.

7.1.2 Policy Decision Service

The function of the Policy Decision Service (PDS) component type is to calculate responses to authorization queries and return authorization decision assertions. The PDS communicates only with the attribute and policy stores and the policy enforcement service. The PDS decides whether a particular entity can perform the requested operation on the desired resource, optionally bound by environmental data like time, by evaluating attributes and policies applicable to the contents of the request.

Policy Decision Management Interface

The policy decision management interface provides functions for configuring, controlling, and managing the health (status and faults) of the policy decision service.

7.1.3 Policy Enforcement Service

The function of the policy enforcement service (PES) component type is to provide an access point for services to make authorization enforcement requests before allowing access to entities requesting to perform an operation on a protected resource. The PES enforces the authorization model, which is described in the attribute and policy stores. It interfaces with the policy decision service. The PES interprets the variety of possible responses from the PDS, and provides a simplified permit/deny enforcement action back to the requesting client.

Services must interface only with the PES in order to apply system authorization policies. The service requesting the authorization decision is responsible for enforcing the action returned from the request. This organization helps to avoid performance bottlenecks that would occur with different organizations.

Policy Enforcement Management Interface

The policy enforcement management interface provides functions for configuring, controlling, and managing the health (status and faults) of the policy enforcement service.

7.1.4 Policy Store Service

The function of the policy store (PS) service component type is to provide network interfaces for storing and retrieving security policies. Policies may be retrieved by ID, or a request may be provided to it where the store determines relevant policies to return.

Policy Store Management Interface

The policy store management interface provides functions for configuring, controlling, and managing the health (status and faults) of the policy store service component type and any underlying data store. The functions for managing the policy store include get, add, and remove operations on security policies.

7.1.5 Attribute Store Service

The functions of the attribute store (AS) service component type are to provide network interfaces for querying attributes for entities, to store those attributes, and to allow the management of the attribute store by attribute authorities.

Attribute Store Management Interface

The attribute store management interface provides functions for configuring, controlling, and managing the health (status and faults) of the attribute store service and any underlying data store. The interface functions include get, add, and remove attributes within the attribute store.

7.2 Data Integrity and Confidentiality

The Data Integrity and Confidentiality capability group is responsible for functions that ensure that data is received from trusted sources and protects data in transit and at rest from unauthorized access on the VDB. Component types that fall into the Data Integrity and Confidentiality capability group include data signing service, signed data verification service, DAR encryptor, and inline network encryptor (INE).

7.2.1 Data Signing Service

The core function of the data signing service component type is to sign data of any type (software binaries, configuration files, databases, etc.). One use case for this service is to provide an authenticator in the form of the digital signature for executable code, which in turn can be authenticated before executing the code on a platform.

7.2.2 Signed Data Verification Service

The function of the signed data verification service component type is to verify the integrity and authenticity of previously signed data.

7.2.3 Data at Rest Encryptor

The function of the data at rest encryptor component type is to secure data on storage devices. It is especially useful when the devices leave the owner's control, such as when devices are repurposed, lost, or returned for repair. It can protect data from unauthorized discovery, lowering the risk of data exposure while preserving retention requirements and assuring information security. The DAR encryptor also provides emergency zeroize functionality.

Data at Rest Encryptor Management Interface

The interface for managing the DAR encryptor component type supports functions for configuration, control, and health management (status and faults) of the encryptor. The DAR encryptor management interface configuration function may allow for the retrieval and setting of authentication type, security policy, and the recovery agent. The control function includes the capability to acquire the current DAR encryptor status and zeroize it.

7.2.4 Inline Network Encryptor

The function of the INE component type is to provide secure communications in IP networks. A given INE encrypts plaintext packets and transports them to other INEs as ciphertext, which are indiscernible to parties without the proper encryption hardware and keys. The receiving INE alone is able to decrypt the ciphertext packets, and sends the unencrypted data as plaintext to the proper network segments. This ensures that no unauthorized parties can make use of the data that is passed from the sender to the receiver. INEs are also often used to place data originating from different classification levels on the same physical medium, which would otherwise be prohibited.

Inline Network Encryptor Transport Interface

There are two main transport interfaces for the INE component: the plaintext and the ciphertext interfaces. The plaintext interface communicates with the vehicular network (in this case, the

VDB). Because this interface is not encrypted, any network node on the VDB can send and receive information via this interface. The data on the ciphertext interface, however, is not discernible to normal network nodes. The payload of each ciphertext packet is encrypted by the INE, and only the headers necessary for proper routing are left unencrypted.

Inline Network Encryptor Management Interface

The interface for managing the INE component type supports functions for configuration, control, and health management (status and faults). The management interface of the INE is, in some cases, not accessible during deployment.

Certain INE status elements may be able to be retrieved via the INE's VDB interface. Examples of these elements include average data throughput and link status. The INE status interface can only be implemented on the plaintext interface (usually the VDB side) since the ciphertext interface is encrypted.

7.3 Host and Network Protection

The Host and Network Protection capability group is responsible for functions that protect the host and VDB network from attacks.

Component types that fall into the Host and Network Protection capability group include Host-Based Antivirus, Host-Based Firewall, IDPS, and Network Firewall.

7.3.1 Host-Based Antivirus

The core functionality of the host-based antivirus component type is to protect a host system (a computer) from malicious software. A host-based antivirus is typically a software package that runs on a single host or end node. It can stop and remove malicious software that it detects on that host only. It is designed to protect the host, as well as the applications and data on it.

Host-Based Antivirus Management Interface

A host-based antivirus has the ability to implement many application-specific controls such as virus signature and program updates, and virus scanning. The interface for managing the host-based antivirus component type supports functions for configuration, control, and health management (status and faults).

7.3.2 Host-Based Firewall

The core functionality of the host-based firewall component type is to protect the host system (a computer) from unauthorized access from the network to which it is attached, the local area network (LAN), or the wide-area network (WAN).

A host-based firewall is most often a software package that runs on a single host or end node. It can restrict incoming and outgoing network activity for that host only. Host-based firewalls can prevent a host from becoming infected and stop an infected host from spreading malware to other hosts. Systems that are connected to an In-Vehicle Network, but are not protected by

network firewalls, should use host-based firewalls. Host-based firewalls are particularly effective against host-specific application layer attacks. Many host-based firewalls can also act as an IDPS that, after detecting an attack in progress, takes actions to thwart the attacker and prevent damage to the targeted host.

Host-Based Firewall Management Interface

Unlike network firewalls, host-based firewalls have the ability to implement many application-specific controls such as port and packet blocking, filtering, and logging. The interface for managing the host-based firewall component type supports functions for configuration, control, and health management (status and faults).

7.3.3 Intrusion Detection and Prevention System

The functions of the IDPS component type is to identify possible incidents, log information about them, attempt to stop them, and report them to security administrators. IDPS functions also include identifying problems with security policies, documenting existing threats, and deterring individuals from violating security policies.

Intrusion Detection and Prevention System Management Interface

The interface for managing the IDPS component type may support functions for configuration, control, and health management (status and faults). This interface may be used to configure and manage security policies and log records. The IDPS configuration is described in a binary file that can be queried or set from the network.

7.3.4 Network Firewall

The core functionality of the network firewall component type is the control of IP traffic between different sub-networks within the larger IP-based network. Firewalls allow access across boundaries to authorized traffic and block unauthorized traffic from entering or leaving certain areas of the network. Following the general rule in U.S. Army networks, firewalls are required to be present at the VDB boundaries with the larger extra-vehicular network. Network firewalls protect the local area network from unauthorized access from the wide-area network.

Network Firewall Management Interface

The interface for managing the network firewall component type supports functions for configuration, control, and health management (status and faults) of the network firewall. The network firewall management interface could be accessible during deployment, or could be made accessible in controlled situations, such as during maintenance or upgrade activities.

The configuration function allows the firewall to be set up and changed via the configuration interface. The network firewall configuration is described in a binary file that can be queried or set from the network. Typically, the firewall management interface is only available from the VDB side (LAN), rather than from the larger network (WAN). The status of the firewall may be

available to authorized nodes on the VDB. Examples of this status include firewall port status, CPU utilization, and data traffic patterns.

8. C4ISR/EW SYSTEMS ARCHITECTURE

The following sections describe the C4ISR/EW related capability groups. The VDB serves as the primary transport mechanism for data that is shared between components of most of these systems. The C4ISR/EW systems will be able to take advantage of the Infrastructure and Information Assurance component types defined in the previous sections. This will enable vehicle and product programs to transition from developing individual stovepipe solutions to an open system architecture approach, which will reduce duplicated functionality and improve SWaP of the overall integrated systems.

The component types are grouped into capability groups, which are described in the following sections.

8.1 Audio Communications

Audio communications are supported on vehicle platforms by several components and interfaces; including legacy voice radios, legacy intercom systems, digital audio network data streams, and audio alerts. These may be implemented in VICTORY-compliant platforms in a number of different ways. Some deployments feature digitized audio, transporting audio packets across the VDB to other digital audio-enabled nodes. In other cases, however, the audio is transported between the components on separate analog channels, and the VDB either does not carry digital audio, or an adapter digitizes the audio to be published on the VDB as a secondary transport. Even when the VDB does not transport digital audio, it may be advantageous for the VDB to be interfaced to the devices so that applications on the shared processors can be used to manage the system or devices.

This section calls out the components and interfaces that the VICTORY Achitecture provides to support audio communications system integration.

8.1.1 Legacy Intercom System

Modern and future intercom systems will be modular, will leverage digital technologies such as Voice over Internet Protocol (VoIP), and will leverage networks for their core transport. But until this transformation occurs, VICTORY must define interfaces to the existing, closed intercom systems.

Legacy intercom systems do not utilize the VDB as the primary transport mechanism for audio data. When an intercom does provide a digital networked interface, or a network adapter is used to create an interface, the data interface will follow the Digital Audio Source component type.

Legacy Intercom System Data Interface

The intercom component type data interface includes the Digital Audio Source component data interface. It may also extend that interface with intercom-specific functions.

Legacy Intercom System Management Interface

The intercom component type management interface provides functions for configuring, controlling, and managing the health (status and faults) of the intercom interface. It may also extend that interface with intercom-specific functions.

8.1.2 Legacy Radio

As mentioned above, some audio communications system deployments do not utilize the VDB as the primary transport mechanism for audio data. When a legacy radio provides a network interface, either as its primary method of transport, as a secondary pathway, or a network adapter is used to create an interface, the data interface follows the digital audio source component type.

Legacy Radio Data Interface

The legacy radio component type data interface includes the digital audio source data interface. It may also extend that interface with legacy radio-specific functions.

Legacy Radio Management Interface

The legacy radio management interface provides functions for configuring, controlling, and managing the health (status and faults) of the legacy radio.

The interface may allow for the configuration of certain parameters, such as the selected channel and audio level. It may also provide status information about the radio to applications on the VDB, such as received power levels.

8.1.3 Digital Audio Source

Most systems currently implement audio communications within the vehicle with analog intercom systems that do not have network connections. Future systems may utilize IP-based audio systems, or systems that have analog connections for primary voice communication, but also provide network interfaces that source or sink digital audio data. A network interface can also be created for analog voice systems by implementing a network adapter that performs analog-to-digital and digital-to-analog conversion to extend the audio to the network. In any of these cases, the network side that interfaces to the audio should be specified and standardized to achieve interoperability between applications and devices that source or sink audio data streams or manage audio systems on the network.

The Digital Audio Source component type provides both data and management interfaces accessible from the VDB. The Digital Audio Source sends digital audio to the network.

Digital Audio Source Data Interface

The digital audio source data interface represents how audio data is formatted and transported on the VDB when in a digital form. This interface is built on top of the VDB data transport interface and specifies the encoding and data formats, network transport setting, streaming protocols, and

application layer protocols that enable interoperability between applications that source and sink digital audio data. Components that publish digital audio on the VDB implement this interface.

Digital Audio Source Management Interface

The digital audio source management interface provides functions for configuring, controlling, and managing the health (status and faults) of the digital audio source. The types of functions that may be provided are related to managing the interface that publishes the data on the VDB, including parameters such as transport settings and how the audio data is encoded and encapsulated.

8.1.4 Audio Announcement Source

Like the Digital Audio Source, the Audio Announcement Source component type provides both data and management interfaces accessible from the VDB. The difference in the two is that the Audio Announcement Source is meant for sending and playing sound files when an event occurs, as opposed to continuously streaming audio.

Audio Announcement Source Data Interface

This interface represents how single audio announcements (typically stored in data files) are formatted and transported on the VDB. This interface specifies the encoding and file formats for interoperability between applications that source and sink digital audio data files. Components that provide audio cues or other audio announcement messages on the VDB must follow this interface.

Audio Announcement Source Management Interface

The audio announcement source management interface provides functions for configuring, controlling, and managing the health (status and faults) of the audio announcement source. The types of functions that may be provided are related to managing the interface that publishes the data on the VDB, including parameters such as how to access the sound data and how the sound data is encoded and encapsulated.

8.2 Video and Imagery Situational Awareness

Video and imagery situational awareness (VISA) systems are implemented in vehicle platforms in order to provide the Warfighter with views of the area surrounding the vehicle or to view still images or video streams from on-board and off-board sources.

VICTORY advocates that these video streams be transported on the VDB so they can be made available to other applications on the VDB. It is anticipated that many implementations utilize the VDB as the main transport mechanism for video data from the source to the end application sinking the video streams, but it is understood that some video applications, such as drive-by-video, have very low latency requirements (e.g., a single frame of video). Other applications require large volumes of high-definition video. In these cases, it may be necessary to transport the video data directly from source to sink (for example, from the sensor directly to a display, or

on a dedicated video bus). In these cases, VICTORY advocates that the primary video stream not be transported on the VDB, but that the video streams also be made available on the VDB with higher latency or lower resolution. This makes the data available to other VDB nodes and for applications that do not have low latency or high definition requirements.

VISA system designs include component types such as video and image sensors (cameras), which are not normally networked; video encoders, which digitize and publish video data on the network; and pan and tilt modules, which control the pointing direction of sensors. The video and imagery component types are useful for the integration of equipment for operational as well as test and training use cases.

User interfaces for visualization of video, and for configuration and control of the systems, are implemented in software. The software component utilizes the VISA component type interfaces, but are not called out as architectural components because software applications are outside of the scope of the architecture.

This section calls out the component types and their interfaces that the VICTORY Architecture provides to support the VISA capability group.

8.2.1 EO/IR Sensor

The electro-optical and infrared (EO/IR) sensor component type provides a networked interface for managing the parameters of an EO/IR Sensor (in most cases a visible light or IR video camera). This component type does not describe the networked data interface (see the EO/IR Encoder), but instead defines the interfaces for managing the sensor.

EO/IR Sensor Management Interface

The EO/IR sensor management interface provides functions for configuring, controlling, and managing the health (status and faults) of the EO/IR Sensor (camera). Note that these devices are not assumed to have native network interfaces. This networked management interface is implemented by a software component running on a shared processing unit, which manages the camera sensor using its native interfaces.

8.2.2 EO/IR Encoder

The function of the EO/IR encoder component type is to digitize data from analog image or video sensors (cameras), encode the data, and publish it on the VDB in a streaming or file-based interface.

EO/IR Encoder Data Interface

The EO/IR encoder data interface defines how digital video data will be streamed on the VDB, specifying the encoding and data format, compression schemes, transport and application layer protocols, file formats, and other aspects that provide interoperability between sources and sinks of video data.

EO/IR Encoder Management Interface

The EO/IR encoder management interface provides functions for configuring, controlling, and managing the health (status and faults) of the EO/IR encoders. This includes querying or updating configuration parameters, such as the resolution and frame rate, encoding, and compression of video streams, capture timing (e.g., number of frames to buffer), and network transport settings. It also includes enabling or disabling the encoder and data publishing. It may include setting up storage parameters for an interface to store and retrieve bulk digital imagery.

8.2.3 Gimbal

The function of the Gimbal component type is for sensing the offset and pointing direction of a movable gimbal used to mount a camera, providing this data to the VDB, and managing the control and configuration parameters of the gimbal.

Note that the functions of the Gimbal component type are similar to that of the Mechanical Articulation Joint component type. The former is specific to VISA systems and cameras, while the latter is a generalization that applies to any movable action point on a vehicle platform, such as a turret.

Gimbal Data Interface

The function of the Gimbal data interface is to allow for the offset and pointing direction of the gimbal to be published on the VDB. The network interface is not always integrated with the movable system itself, but may be accomplished via a network adapter that interfaces to the Gimbal in its native format (e.g., serial, analog signals) and publishes that on the VDB. Note that the network adapter may be a software component running on an SPU, with the SPU physically connected to the system through a serial port or other connection.

Gimbal Management Interface

The gimbal management interface provides functions for configuring, controlling, and managing the health (status and faults) of the Gimbal. The configuration and control functions include setting up the data reporting parameters, querying the current pointing direction of the gimbal, and controlling the pointing direction of the gimbal via the VDB. The health functions include reporting faults in the Gimbal system or its data interface.

8.3 Off-Vehicle Network Interfacing

The Off-Vehicle Network Interfacing capability group provides the functionality necessary to build an interface between an in-vehicle network (a VDB) and an off-vehicle network. This group includes routing, wireless transport, and gateway functions.

8.3.1 Router

The core functionality of the router component type is the direction of IP data packets, typically at a higher level than that of the switch. In typical network configurations, routers are present at the interface between two networks that are located on different logical subnets. In a vehicular

platform, this typically means they are found at the border that the VDB shares with the extra-vehicular network.

Router Data Interface

The router data interface implements a wide variety of protocols to facilitate the transport of data across the various sub-networks that it joins.

Router Management Interface

The router management interface provides functions for configuration, control, status reporting, and fault management of the router.

The configuration of the router can be set up and changed via the management interface. Settings such as IP subnet parameters, static routes, and multicast functionality are essential in the transport of data on the VDB. Certain functionality implemented by the router can be altered via the interface with the VDB, including port activation and power cycling. The status of the router is available to authorized nodes on the VDB. Examples of this status include the link status of ports, CPU utilization, and data traffic patterns of the router.

8.3.2 Network Radio

Most vehicle platforms host Network Radios that provide data communications with entities outside of the platform. The radios that are used to accomplish this are outside the scope of the VICTORY Architecture, and VICTORY makes no attempt to standardize or otherwise influence items such as radio waveforms and tactical Internet architecture. However, these wireless devices often communicate with C4ISR/EW components on the VDB. It may be useful in some cases for the VDB to be able to receive data from or send data to these networks. It may also be desirable for applications that are used to set up and manage the Network Radios to be able to be hosted on the VDB shared processing resources.

Network Radio Data Transport Interface

Communicating with the Network Radios may require a device to adapt the radio's native transport, network, or physical/data link interfaces. It is also desirable for the data interface to include some routing or filtering functionality to control which data is selected and routed between the wireless and VDB networks. Examples of data that may be selected and routed between the networks could include SA data and Threat Detection System information.

Network Radio Management Interface

The network radio management interface provides functions for configuring, controlling, and managing the health (status and faults) of the Network Radio.

This includes setting configuration items, such as receiver sensitivity, desired throughput rates, packet format, operational bandwidth, modulation scheme utilized, allowable and desired bit-

error rates, and other configuration settings related to the health and performance of the wireless connection. Some network radio component type functionality can be altered via the management interface, such as connecting and disconnecting to the wireless network and establishing links with other radios on the network. The status of the network radio component is available to authorized nodes on the VDB. Examples of this status could include the status of the connectivity to the Network Radio, as well as radio bandwidth availability and current network usage statistics.

8.3.3 Off-Vehicle Network Gateway

The Off-Vehicle Network Gateway allows data to flow between a VDB and a non-VICTORY network, usually a wireless network communicating off of the vehicle. This CT is called a gateway instead of a bridge intentionally. This section first explains the differences between a gateway, a bridge, and an adapter in general, and explains why this CT is a gateway.

A *network bridge* joins two similar types of networks so that they look like one network. The word transparent is often used with bridges because network clients on either side do not know the bridge is even in place.

A *network gateway* joins two dissimilar networks, and performs the functions necessary to translate, marshal, and filter the messages that cross the boundary, and to manage the difference in timing and QoS properties of the transport on either side. A network gateway does not provide transparent integration of the networks, but is configured to allow specific data to move between the networks.

A *network adapter* converts the native interface of a device to a required network interface, allowing the device to "appear as conforming" to the network interface. An adapter is a one-sided concept because the entity being adapted is usually not modified to be aware of the entities on the other side of the adapter. If the entity being adapted has awareness or transparent access to the entities on the other side of the adapter, then it is more of a bridge than an adapter.

The Off-Vehicle Network Gateway component type provides the network gateway functionality described above. This includes managing the links to off-vehicle networks, and providing a gateway between these networks and the VDB. It does not attempt to transparently bridge the vehicle network with the off-vehicle network.

The off-vehicle networks are supported by one or more Network Radios (see the Network Radio component type description). The wireless networks support far lower data rates and are less dependable than the wired VDB networks. The Off-Vehicle Network Gateway component type ensures that egress and ingress of data from the vehicle is done in a way that matches the policies and capabilities of the off-vehicle network. The functions include identifying and selecting the subset of data that flows between the vehicle networks (VDBs) and the off-vehicle network (filtering), converting between on- and off-vehicle formats (translating), managing data queues going into and out of the vehicle, following the QoS and routing policies of the networks (marshaling), and managing the interfaces to the wireless network links.

Off-Vehicle Network Gateway Data Interface

The off-vehicle network gateway data interface may support the filtering, translation, marshaling, and QoS adaptation functions described above.

Off-Vehicle Network Gateway Management Interface

The off-vehicle network gateway management interface may provide functions for configuring, controlling, and managing the health (status and faults) of the gateway.

8.4 Modular RF Signal Resources

The component types in the modular RF signal resources capability group extend the value of VICTORY to enable modularity within systems that support complex RF missions. These component types have resulted from the Modular Open RF Architecture (MORA) effort. MORA decomposes RF systems by layers and functions to promote efficiency, enable resource sharing, and support low-risk reuse in future capability insertions while enabling multi-mission flexibility in a single RF node.

8.4.1 MORA Overview

MORA extends the scope of VICTORY by adding a low latency transport mechanism, data streaming interfaces, new message types, management operations, and functional concepts that are specific to RF applications. Examples of RF applications include voice communications, data communications, and EW.

MORA enables the implementation of modular, flexible RF systems capable of performing multiple missions with the same physical resources. In basic terms, a MORA system consists of an integration of multiple MORA devices utilizing a dual network topology. This approach allows MORA systems to leverage the VDB for non-time-critical functionality, and extend it with a separate network partition that can support time-critical functionality, such as low latency message control signaling to replace discrete signaling.

The MORA extension of VICTORY modularizes RF missions, which require additional capabilities including:

- Resource management and monitoring of the operation of the RF devices and applications.
- Dependable, low latency, and high-speed access and transport of high-volume digital RF signal data sample streams.
- Message-based, real-time (low latency) control of the RF chain including antenna arrays, RF switches, RF filters and gain stages, frequency converters, analog-to-digital converters (ADCs), digital-to-analog converters (DACs), field-programmable gate arrays (FPGAs), general purpose processors (GPPs), and digital signal processors (DSPs).
- Formatting, encoding, encapsulation, transport, and routing of messages between the MORA and VICTORY components.

MORA Low Latency Bus

The MORA Low Latency Bus (ML2B) extends VICTORY by introducing an addressable bus to provide higher bandwidth, lower latency, and more deterministic transport. The ML2B provides more deterministic and reliable message delivery with very low latency (end-to-end latency on the order of 1 microsecond), where latency is defined as the overall time to send/receive a message, including transmission, propagation, and switch delays. The ML2B replaces current point-to-point discrete signals (e.g., Transistor-to-Transistor Logic [TTL]) between RF system components. Specific uses of the ML2B include transmit/receive switching, blanking, and tuning messages. Use of an addressable bus instead of point-to-point signals improves scalability of the MORA architecture and enables component distribution throughout the platform. The ML2B can also be used for digital RF, including real-time control and delivery of high volume, high rate signal data.

The ML2B uses 10 Gigabit Ethernet (10GbE) or greater due to its ubiquity in commercial products. Since "when" a device can send a message is as important as "how long" it takes for the message to be delivered, deterministic protocols that use a scheduling approach are not appropriate as they limit the maximum message rate. MORA achieves near-deterministic message delivery by using a "fat pipe" to send very small control messages that receive priority in the network based on QoS policies. Discrete signals that are currently hard-wired to share small binary fields can also be conveyed through the use of binary control messages over the ML2B that fit within a minimum size Ethernet frame (i.e., 64 bytes). Signal data messages over the ML2B may use larger packets, such as jumbo frames (i.e., 9000 bytes), to improve protocol efficiency. The combination of high throughput and QoS mechanisms reduces the probability that buffering delays occur. This is further reduced by the fact that most commercial switches are able to switch at line rate, which limits the potential for network contention to traffic to/from the same device.

Note that the MORA low latency bus (ML2B) provides the transport for the modular RF signal resources capability group shown in Figure 6. An ML2B is composed of a set of network switches that are compliant with the low latency switch component type, which is shown in the Infrastructure capability group in Figure 4.

A combination of VICTORY- and MORA-specific component types defines the interface(s) and transport methods with which the devices in a system interact with one another. Figure 6 lists the MORA component types that can be used to create modular RF applications. These include the MORA device, MORA signal resource manager, and the MORA signal port manager. These can be combined with the RF direction finding engine, RF geolocation engine, RF spectrum sensor, RF signal detector, and the RF signal generator component types from the RF processing category to create a large breadth of integrated capabilities.

MORA decomposes a monolithic RF system into three different device types including software-defined radio (SDR), radiohead (RHD), and RF conditioning and distribution (RCD) device types. Other VICTORY Component Types are recommended to complete the system

functionality required by RF missions. The ML2B supports low latency, highly deterministic messaging for both resource control (as a means to replace discrete signaling) and digital RF signal data sample streams, which may eliminate the need for analog RF cables. MORA also supports the transport of analog RF signals on separate fiber optic or coaxial cables.

MORA Top-Down Perspective

MORA is logically designed to create efficiency, flexibility, reusability, and scalability in future RF systems. Current RF system implementations are typically single-mission focused with closed or proprietary interfaces between stove-piped resources. By defining open interfaces for the processing and signal resources, MORA enables multi-mission capabilities that utilize and share resources within a single RF node in the tactical environment. Figure 9 below depicts the top-down perspective of a MORA system and the boundary of layers that the MORA specification encompasses.

The operation of RF missions typically consists of a combination of complex hardware, firmware, and software resources. These resources can require significant space and power and therefore can often be constrained in smaller platforms. It is essential that future systems be based on an architecture that fully promotes maximum efficiency and flexibility of resources. At the foundation of all RF missions, signal resources are required to capture, radiate, condition, distribute, translate, and/or convert RF signals. For RF chain receive operations, signal resources provide digital signal data to processing resources in order to develop information. Processing resources provide information to platform application resources in order to develop comprehensive knowledge of the electromagnetic environment within the RF range of the platform. In some RF missions, this platform knowledge can be subsequently provided to theater/mission command resources to further develop a broader comprehensive knowledge of the regional or global electromagnetic environment. For RF chain transmit operations, the interaction is the same, but in the reverse order and flow.

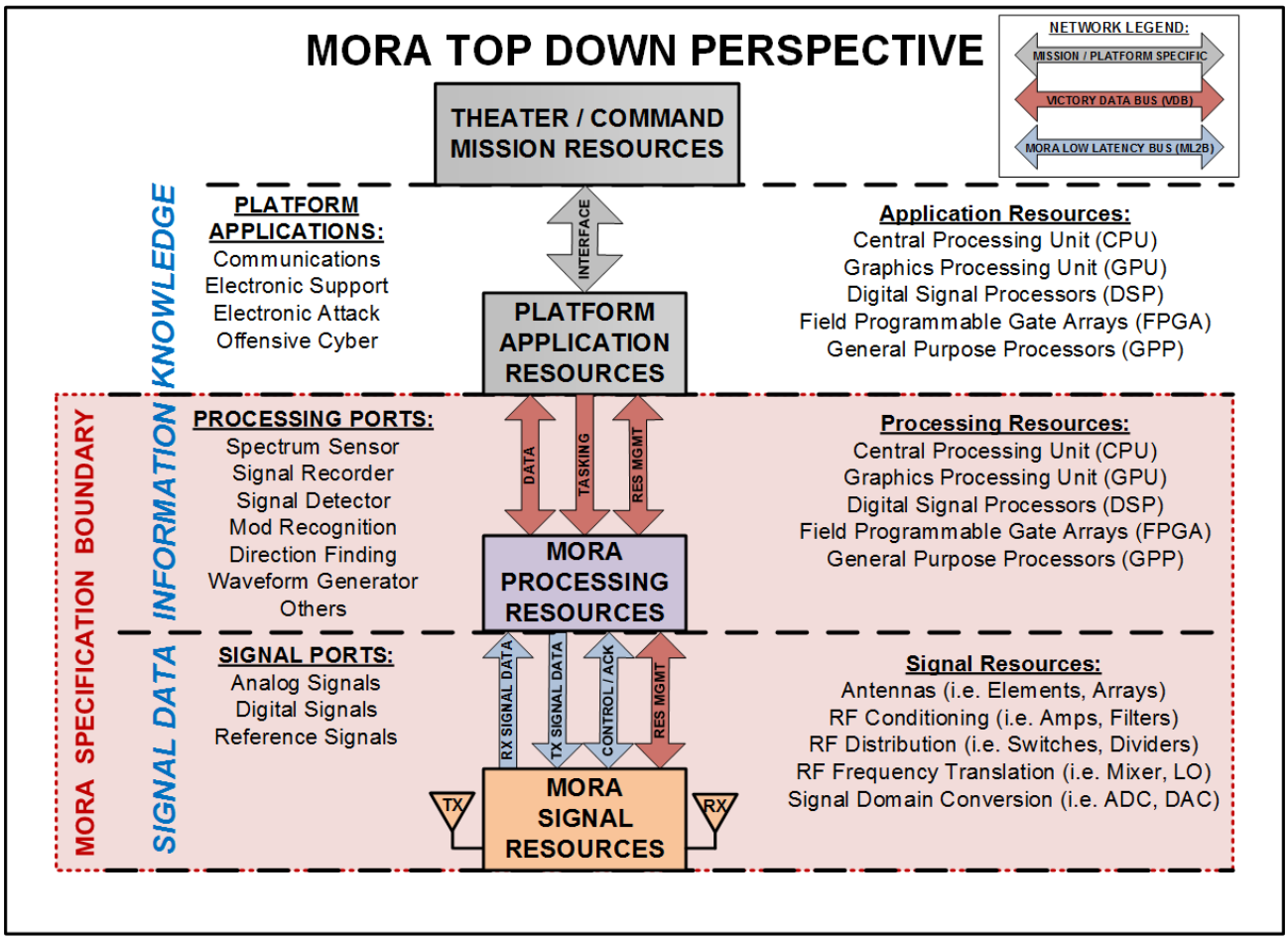

Figure 9. MORA Top-Down Perspective

MORA seeks to standardize the interfaces for these signal resources and processing resources within the RF mission. In particular, the messaging interfaces required between the signal resources and processing resources must be defined so that signal resource capabilities can be understood, allocated, and controlled by processing resources such that signal chains can be constructed, ultimately transferring digital signal data between the signal and processing resource layers. In addition, the messaging interfaces required between processing resources and application resources must be defined so that processing resource capabilities can be understood, allocated, and controlled by application resources. This enables the construction of information chains that can transfer information between the processing and application resource layers.

MORA Signal Resource

A MORA Signal Resource is a resource within a MORA Device that captures, radiates, conditions, distributes, translates, and/or converts RF signals in support of receive and/or transmit operations in a MORA system. Examples of signal resources include antennas, amplifiers, filters, switches, combiners, dividers, mixers, oscillators, ADCs and DACs. The

Signal Streams related to a Signal Resource are transported between devices in analog or digital formats, depending upon whether it is an analog or digital Signal Resource. Analog signals are routinely routed between devices via coaxial cables or could be transformed for transport on fiber cables. Digital signals are routinely transported between devices via network copper cables (e.g., Ethernet) or could be transported on network fiber cables in high volume/high data rate systems.

Standardized access to and control of MORA signal resources are accomplished over the VDB and the ML2B. MORA Device and signal resource management interfaces provided over the VDB are used for general discovery, announcement of capabilities, resource reservation, and resource release. Once a resource is reserved, the real-time operation of signal resources takes place over the ML2B utilizing MORA data messages (MDMs). MDMs follow a specific implementation of the VITA-49.2 Standard, which support the transport of digital packets for control, context, and signal data (time or frequency domain) over the ML2B.

MORA Processing Resource

A MORA processing resource is a resource within a MORA Device that processes signal data into information, processes information into signal data, or further processes information in support of receive and/or transmit operations in a MORA system. Examples of processing resources include central processing units (CPUs), DSPs, field programmable gate arrays (FPGAs), GPPs, and graphics processing units (GPUs). Processing resource signal or information streams are transported between devices in digital formats. Digital signals are routinely transported between devices via network copper cables (e.g., Ethernet) or could be transported on network fiber cables in high volume/high data rate systems.

Standardized access and control of MORA Processing Resources is accomplished over the VDB. MORA Processing Resource management interfaces provided over the VDB are used for general discovery, announcement of capabilities, resource reservation, resource release, and tasking from the processing or application layers. MORA Processing Resource data interfaces provided over the VDB are used for exchanging information with other resources within the processing or application layers. MORA Signal Resource interfaces on the ML2B are used to establish an interface with the signal resource layer for processing resources that require access to signal resources.

MORA Transitional Implementation

Past investments in open architecture standards commonality and reuse within the Department of Defense (DoD) need to be supported by MORA. It is unrealistic to expect a new or emerging architecture to be immediately and/or fully implemented across existing systems, platforms, and projects. Instead, a new architecture should be implementable in varying degrees over time by providing a means to leverage past investments. MORA supports partial implementations for capability and technology insertions into existing systems by offering standardized access to new MORA signal and processing resources from other specific resources already in place. Figure 10

below shows the concept of a MORA implementation with open access to existing specific resources at both the processing and signal layers.

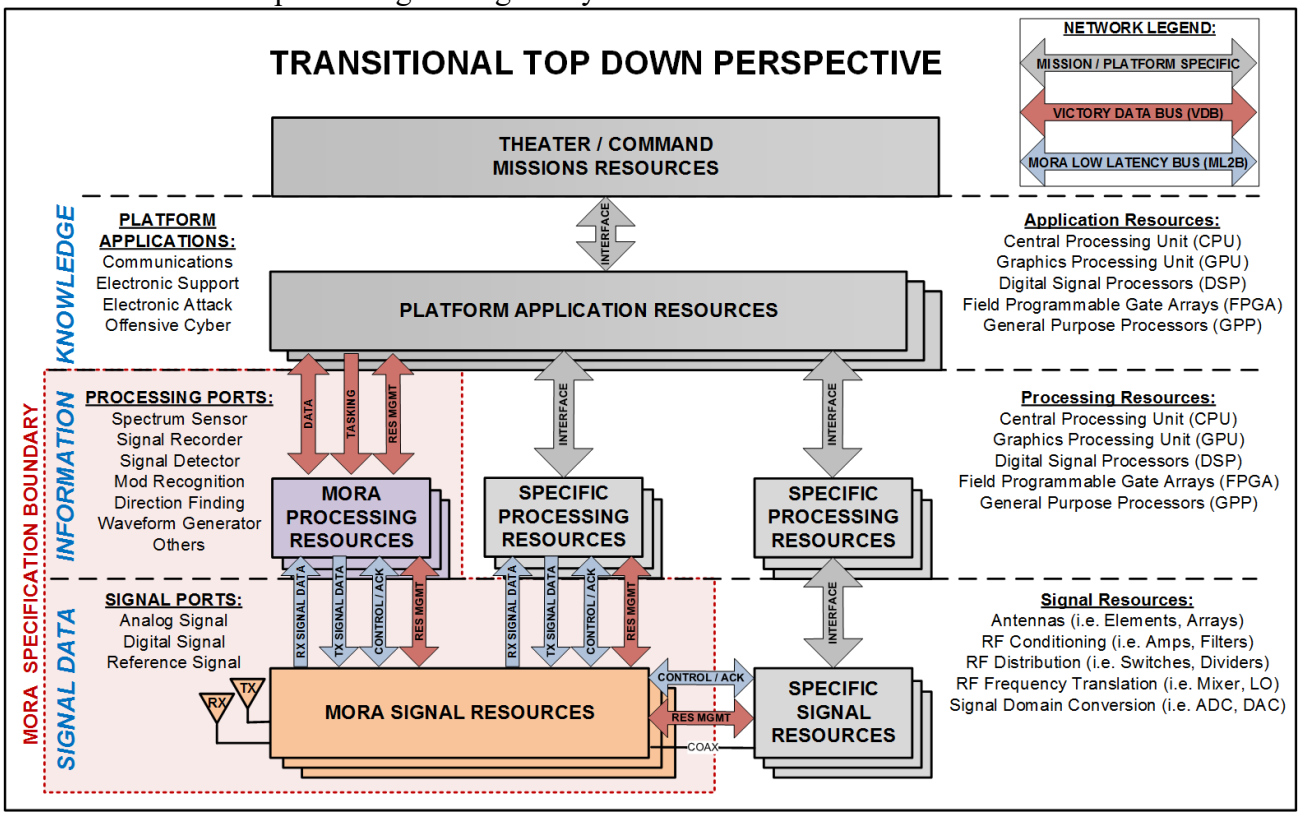

Figure 10. MORA Transitional RF System Top-Down Perspective

RF missions constantly face new challenges which require rapid insertion of new capabilities with the goal of minimal risk, schedule, and cost. MORA is a key enabler for achieving this goal. When new capability insertions require that new resources be integrated into an existing system, the system designer should strive to add MORA resources to expand the system's access to, and control of, these new resources while also promoting reuse into other MORA-based systems.

As depicted in the figure above, MORA Signal and Processing Resource insertions promote flexibility and efficiency to the signal, processing, and application layers. When MORA Signal Resources are added, they can be accessed by other pre-existing, specific signal resources and pre-existing, specific processing resources. These new MORA Signal Resources can also feed into new MORA Processing Resources which, in turn, provide new capabilities to pre-existing, specific application resources.

8.4.2 Modular RF Signal Resources Component Types

As has been described, MORA defines a set of VICTORY Component Types that combine the functionality and interfaces described in the MORA overview. MORA defines the MORA low latency bus (ML2B) to low latency transport. The following CTs leverage the ML2B for low latency signaling, and make up the core MORA capability:

- MORA Device
- MORA Signal Resource Manager
- MORA Signal Port Manager

These component types can be combined together with other VICTORY component types to form sophisticated, modular RF processing capabilities.

MORA has defined a set of atomic blocks called MORA processing ports, which are partitions of processing resources. A MORA processing port is analogous to the component types defined in the RF processing section. Processing ports include management and data interfaces on the VDB and may also interface with signal resources through signal ports.

Processing ports request use of and manage Signal Ports and Signal Resources they require, and provide a straightforward network-based interface for applications.

The MORA processing ports that have been defined by MORA include the following:

- MORA Direction Finding Engine
- MORA RF Spectrum Sensor

The component types that are analogous to MORA processing ports are described in the RF processing section. This section defines component types that support the core modular RF capability of MORA, and which are combined with those in the RF processing capability group to form modular RF systems.

8.4.3 MORA Device

MORA device component type provides signal resources and/or processing resources to support RF receive and/or transmission applications. This component type supports flexibility by allowing the system designer to construct a system of devices that best accomplishes the RF mission needs within the platform constraints while maintaining standardized access and control of resources.

As described in Section 8.4.1, there are three MORA device types, including software defined radio, radiohead, and RF conditioning and distribution devices. The differences are merely the combinations of signal and processing resources that they include. From an architectural perspective, they are equivalent.

An instance of the MORA device component type has one or more VDB interface, and zero or more ML2B interfaces. Each MORA device instance must implement one MORA signal resource manager, zero or one MORA signal port managers, and zero or more processing resources.

MORA Device Management Interface

The MORA device management interface supports basic management of the device as an End Node on the VDB. The RF domain-specific interfaces are provided by the other component types that a MORA device hosts.

8.4.4 MORA Signal Resource Manager

The MORA Signal Resource Manager provides VDB interfaces to describe capabilities of signal resources. Capabilities of a signal resource may include signal domain conversion, RF conditioning, and RF distribution capabilities. It also provides an interface to reserve signal ports and discover the endpoints for real-time control and data interfacing with the VDB.

MORA Signal Resource Management Interface

The MORA signal resource management interface is a VDB command-response style interface that provides the means of determining the RF processing capabilities of a MORA Device, requesting access to resources (reservation), and releasing resources when they are no longer required. The unit that can be reserved is a signal port, which has an associated set of RF processing capabilities that may be set up dynamically at runtime.

The MORA Device capability description includes the signal ports and RF processing chain(s) that the device can support. Note that signal resources on a MORA device may be shared among its signal ports, so the set of RF processing capabilities of a signal port may depend upon the current reservations and configuration of other signal ports. The reservation requestor includes the capabilities it needs, and the response may be a denial of the reservation based on those requested capabilities due either to capability limitations of the MORA Device, or to capability limitations based on the current reservations.

8.4.5 MORA Signal Port Manager

The MORA signal port manager component type interfaces support RF signaling, data, and flow control interactions, which are (usually) transported on the ML2B, not on the VDB. Its data interfaces exchange RF data encoded into MDMs instead of VICTORY Data Messages. Its management interactions are also encoded into MDMs and interleaved on the same transport as the data messages, as opposed to the command-response network interfaces used by most VICTORY CTs. The MDMs follow a binary encoded format designed for optimized transport of RF sample streams and small data structures.

The Signal Port Manager does not follow the common pattern of transport, data, and management interfaces present on other VICTORY CTs, and is not accessible directly on the VDB via the usual kinds of data and management interfaces.

8.5 Radio Frequency Processing

Electronic warfare, intelligence, and communications systems operate in the electromagnetic spectrum to support a wide range of missions. These systems collect data, process signals to

classify or locate entities, and/or generate signals. RF processing capabilities utilize a variety of technologies and components, and may operate across vehicle boundaries.

VICTORY aims to define component types that provide modular capabilities that can be composed in many ways to support large set of RF functions. The future of RF systems is that they will become more modular over time, and VICTORY is laying the pathway for supporting that modularity. The RF processing component types are organized to support integration of future modular RF systems, but to also support system-level interfaces to existing, current-force products that are not modular.

This approach requires defining system-level interfaces that align with current-force RF processing systems, while also defining modular-level interfaces to component types that separate common, shared functionality.

By supporting system-level interfaces between RF processing capabilities and the VDB, VICTORY makes it possible for other entities with appropriate authorization can access system-level information from current-force systems that is relevant to the situational awareness or battle command, increasing the value of the individual systems to the mission.

By defining a set of component types that represent common RF processing functions, VICTORY looks to the future where RF processing capabilities will be implemented by composing modular RF components in multiple ways.

Note that the MORA effort described earlier is defining extensions to VICTORY that will promote modularity internally to RF systems. The concepts in the modular RF processing component types described in the following originated in MORA. However, they are defined in a way that allows them to define interface with fixed or modular implementations.

To support the case that the implementation of an RF processing capability is itself to be modular, VICTORY encapsulates the modular RF signal resources component type category. These component types were conceptualized and defined by MORA.

The approach of defining top-level RF processing capabilities to support both modular and fixed internal structure has been taken for each of the modular RF processing component types.

This section first defines the modular RF processing component types, and then the system-level EW system component type at the end.

8.5.1 RF Direction Finding Engine

The RF direction finding engine (DFE) component type encapsulates the functionality to listen for and detect RF signals of interest, and to determine the incident angle of arrival of the signal. The DFE determines the line of bearing (LOB) of incoming RF signals of interest by analyzing the phase shifts in multiple geometrically-diverse antennas. The DFE receives multi-channel

data, commonly referred to as IQ data, from a digital receive signal port, and calculates the LOB with knowledge of the relative spacing of the antennas.

The DFE component type has management interface through which it can be tasked from the VDB. This management interface includes task-specific settings such as the desired mode (set-on DF or scan DF), LOB type (2D or 3D), timing, duration, number of repetitions, repetition interval, frequency, bandwidth, and threshold. LOB information is sent to the client through its data interface on the VDB. This data interface includes task-specific responses that include azimuth bearing and confidence, elevation bearing and confidence (for 3D types), received signal strength, and position (starting point of the LOB vector).

In the case that the DFE is internally modular, and leverages the modular RF components defined by MORA, the signal data inputs to the DFE will be received from a MORA signal resources through MDM interfaces on the ML2B. Those concepts are defined in the modular RF signal resources section.

RF Direction Finding Engine Data Interface

The RF direction finding engine data interface publishes signal detection line of bearing reports.

RF Direction Finding Engine Management Interface

In addition to the common service management functionality, the DFE management interface enables the system to be configured for specific signals or areas of interest. It also enables current and recent detection history to be requested.

8.5.2 RF Geolocation Engine

The RF geolocation engine (GLE) component type is an instance of an RF processing resource that encapsulates the functionality to listen for and detect RF signals of interest, and to determine the geodetic coordinates of the signal emitter, in a geodetic coordinate system.

Similar to the DFE, the GLE determines the line of bearing (LOB) to the incoming RF signals of interest. It also has the capability of determining the distance to the emitter. Using this information and the knowledge of the location and an attitude of the sensor in a geodetic coordinate system, the GLE calculates an estimate of the geodetic location of the emitter.

RF Geolocation Engine Data Interface

The RF geolocation engine data interface publishes signal detection geolocation reports on the VDB. The geolocation reports include an estimate of the global (geodetic) location of the emitter, including a region within which the emitter is likely to exist, considering the uncertainties in the sensor data and calculation methodology.

In the case that the RF geolocation engine is internally modular, and leverages the modular RF components defined by MORA, it will receive signal data from RF Signal Resources through MDM interfaces on the ML2B.

RF Geolocation Engine Management Interface

In addition to the common service management functionality, the GLE management interface enables the system to be configured for signals and areas of interest, and enables the current and recent detection history to be requested.

The GLE processing resource is tasked by clients on the VDB through the management interface. This management interface includes task-specific settings such as the desired mode, timing, duration, number of repetitions, repetition interval, frequency, bandwidth, and threshold.

8.5.3 RF Spectrum Sensor

The RF spectrum sensor component type is an instance of an RF processing resource that derives RF spectral information, such as received power versus frequency bin (frequency interval), typically by applying a Fast Fourier Transform (FFT) to quadrature signals (IQ data) from a digital receive signal port. The RF spectrum sensor processing port uses context information about the digital IQ stream to determine antenna reference, frequency, bandwidth, format, and sampling rate details of the input raw signal data. The RF spectrum sensor messages may also include the number of samples used to compute average power, peak power versus frequency bin, and minimum power versus frequency bin.

RF Spectrum Sensor Data Interface

The RF spectrum sensor data interface publishes information about the observed RF spectrum, such as power spectral density (PSD) on the VDB.

In the case that the RF spectrum sensor is internally modular, and leverages the modular RF components defined by MORA, it will receive signal data from RF Signal Resources through MDM interfaces on the ML2B.

RF Spectrum Sensor Management Interface

In addition to the common service management functionality, the RF spectrum sensor management interface allows the spectrum sensing parameters to be viewed and configured, and for the current and recent detection history to be requested.

8.5.4 RF Signal Detector

The RF signal detector component type is an instance of an RF processing resource with the functionality to observe signals in a selected band and report detections of particular types of signals (signals of interest). It does not include the capability to determine the direction of arrival and does not emit energy.

RF Signal Detector Data Interface

The RF signal detector data interface publishes signal detection reports on the VDB.

In the case that the RF signal detector is internally modular, and leverages the modular RF components defined by MORA, it will receive signal data from RF Signal Resources through MDM interfaces on the ML2B.

RF Signal Detector Management Interface

In addition to the common service management functionality, the RF signal detector management interface allows the detection parameters to be viewed and configured and for the current and recent detection history to be requested.

8.5.5 RF Signal Generator

The RF signal generator component type is an instance of an RF processing resource that emits RF energy following a specified waveform (wireless signal format and protocol).

RF Signal Generator Data Interface

The RF signal generator data interface publishes signal generation action reports on the VDB.

In the case that the RF signal generator is internally modular, and leverages the modular RF components defined by MORA, it will receive signal data from RF Signal Resources through MDM interfaces on the ML2B.

RF Signal Generator Management Interface

In addition to the common service management functionality, the RF signal generator management interface allows waveform and transmission parameters to be configured and transmissions to be controlled.

8.5.6 Electronic Warfare System

As discussed previously, the Electronic Warfare System CT was defined to encapsulate the functionality provided by some of the currently deployed counter radio-controlled improvised explosive device (RCIED) electronic warfare (CREW) systems. For that reason, the electronic warfare system component type provides a wide range of EW and intelligence capabilities, which are normally thought of separately in the RF community. The EW system functionality encompasses spectrum sensing, signal detection, signal generation, direction finding, and geolocation capabilities.

The EW system component type defines system-level interfaces, and does not assume any modularity with the integrated EW system.

Earlier in this section, a set of modular RF processing component types were defined which separate common functions in a way that will lay the groundwork for deploying emerging modular components to provide flexible RF processing capabilities. This will provide more flexibility, and to align better with existing and emerging systems. These component types may be specified by the standards body in a way that allows them to be combined to support the same functionality and interfaces as the EW System component type. The standards body may decide to reorganize the functions across this set of component types.

That said, VICTORY Standards Specifications are defined by the stakeholders, including implementation programs. This will ensure that the RF processing component type specifications will support existing implementations of the EW System component type functionality, while also providing a smooth transition path for leveraging the modular RF component types.

In the case that an EW system is internally modular (made up of modular components) it should be defined using the component-level modular RF processing component types defined earlier in this section.

Electronic Warfare System Data Interface

The EW system data interface may be used for publishing data relevant to its objective and operation on the VDB, making it available to other systems and components attached to the VDB. EW systems may also sink data from the other systems and components on the VDB.

Electronic Warfare System Management Interface

The EW system management interface provides functions for configuring, controlling, and managing the health (status and faults) of the EW system.

The configuration function allows viewing and setting configuration parameters such as the load set. The control function allows the state of the EW system component type to be viewed or altered by authorized entities on the VDB. This includes functions to power the system on or off, control the system state, or zeroize the system. The health management functions of the EW system management interface allow status and fault information to be provided to entities on the VDB, such as the amount of data that the system is currently storing, the current draw, or the temperature of the system.

For now, this combined EW System CT will continue to be supported by VICTORY, but over time its capabilities may be replaced with the modular RF processing component types listed above. Once the capabilities are completely supported, the EW system component type may be phased out.

9. PLATFORM SYSTEMS ARCHITECTURE

Most vehicle platforms host a variety of electronic systems that monitor, control, and/or support diagnostics of the vehicle and its subsystems. While the primary functions of these systems are outside of the scope of the VICTORY Architecture, VICTORY does identify system-level interfaces for interoperating with the VDB and applications running on it. The following sections identify some of these platform system capability groups.

Note that the platform systems themselves are outside of the VICTORY scope, so these systems are interfaced to, but not integrated by the VDB. The component types that make up the platform system architecture define only system-level VDB interfaces (as opposed to interfaces to the system components).

9.1 Vehicle Systems Interfacing

A Vehicle platform is a complex system of systems, and include many sensors, actuators, buses, and other devices that may be tightly integrated with the platform. The component types in the vehicle systems interfacing capability group cover interoperating with electronic automotive and power distribution systems, as well as with the systems that articulate (move parts with respect to each other) mechanical elements of the vehicle. This third type of system is called a mechanical articulation joint.

These capabilities allow data from systems that make up the vehicle to be passively retrieved, and reported on the VDB, and in some cases for applications on the VDB to send data configuration and control requests to the system.

These systems are employed at varying levels, depending upon the vehicle complexity. The VICTORY Data Bus does not integrate these systems; they operate on domain-specific buses that have matured over decades, and their designs and behavior are based on deep domain-specific knowledge. Typically, these systems implement a CAN or a 1553B bus to connect the electronics associated with the various vehicular mechanical systems. Although the automotive systems themselves are outside the scope of the VICTORY Architecture, VICTORY identifies interfaces for receiving data from the vehicle automotive systems and for managing the interface between the VDB and these systems. The intent is not for the VDB to replace the buses that integrate the automotive systems, nor is it for the VDB to be involved in the control or instrumentation of the vehicle drivetrains. Instead, the intent of the automotive system CT is to provide a system-level network adapter for the automotive systems and the VDB. See the definition of network adapter in an earlier section. This CT allows information to flow from the vehicle drivetrain and other embedded control systems onto the VDB, in an open and standard format. This is a one-way flow (no data from the VDB is published on the automotive bus). The data is published on the VDB at a lower rate than it is produced, and the publication does not interfere with the power train control loops. This information is valuable for applications such as digital gauges, vehicle status logging, and logistics applications such as condition based maintenance (CBM). The automotive system data and management interfaces provide that conduit for bridging the VDB and the automotive systems.

9.1.1 Automotive System

Automotive buses such as CAN are used to integrate electronic sensors and actuators with systems that control the drivetrain and supporting systems. These buses are relatively low-performance, but provide a higher level of determinism than commonly used networks. This allows the systems to be controlled predictably, dependably, and safely.

The Automotive System CT provides the capability for the VDB to interface to the automotive bus, passively receiving the bus data, and publishing a subset of the data on the VDB to be used by other applications. The components type is independent of the inner workings of the automotive bus and systems. It provides the capability of publishing a standardized set of automotive messages on the VDB that have a fixed semantic independent of the type of bus or the vehicle implementation. In this way, the Automotive System CT is an adapter for the automotive bus to the VDB (a one-way gateway).

Automotive System Data Interface

The automotive system data interface publishes relevant automotive data on the VDB. Examples of information that this interface may provide include engine temperature, engine speed, and fuel level.

Automotive System Management Interface

The automotive system management interface provides functions for configuring, controlling, and managing the health (status and faults) of the adapter between the VDB and the automotive system.

Note that this management interface is not meant to manage the automotive system or the powertrain control components attached to it. It is for managing the adapter that acquires data from the automotive system and publishes it on the VDB.

The automotive system management interface does allow some configuration items to be set, such as which parameters are to be acquired and published on the VDB and the minimum update rate for publishing the data. The status of the automotive system link is available to authorized nodes on the VDB. Examples of this status include information regarding which data is being collected and the last update time for the parameters.

Note that this interface is not meant to provide an interface for configuration, control, or health management of the vehicle platform itself. The configuration and control functions are not meant to effect changes to the vehicle configuration or state, and the health-related functions relate to the health of the bus link itself, not the health of the underlying vehicle. The health of the vehicle may be obtained by interpreting the parameter values that are captured and published as messages on the VDB. The VICTORY Architecture and specifications do not have scope over the vehicle systems themselves, so do not attempt to affect them or to provide standard interpretations of vehicle-specific parameter values.

9.1.2 Power Distribution System

On some current force platforms, electronically controlled power distribution systems are used to distribute power to the various systems on the platform, and to monitor and control the power usage by the C4ISR systems, crew protection systems, EW, and other systems. These systems are often made up of components called power distribution units (PDUs) which are connected by CAN or other buses. The VICTORY Architecture defines the power distribution system component type to enable the VDB nodes to monitor and perhaps control the power distribution systems. For example, a WMI running on a VDB shared processing unit with a display may be used to control and obtain status from the power distribution system and to announce warnings of fault conditions.

The power distribution system component type represents a gateway between the power distribution system and the VDB, which gathers data from the power system and publishes it on the VDB, but allows entities on the VDB to perform management operations on the power system.

Power Distribution System Data Interface

The power distribution system data interface reports data relevant to the operation of the power system. This information could include the amount of power that is currently being dissipated by a given vehicle electronic system or the total amount of energy that has been consumed by a particular system over the course of a mission.

Power Distribution System Management Interface

The management interface to the power distribution system component type provides functions for configuring, controlling, and managing the health (status and faults) of the power distribution system and its network interface.

The configuration function allows for the setting of certain configuration items, such as which types of alert messages should be published on the VDB, and under which situations. The control function allows for certain functionality to be alterable via the management interface, such as enabling or disabling power to specific vehicle electronic systems. This function could be used by power management applications running on the VDB to optimize power use or perform “load shedding.” The health (status and faults) of the power distribution system is available to authorized nodes on the VDB. Examples of relevant health information include which electronic systems are currently being powered, their current draws, and fault events that have occurred.

9.1.3 Mechanical Articulation Joint

Vehicle platforms commonly have hulls with mechanically movable sections, such as turrets and booms. The devices and sensors that control these mechanical movements are necessarily tightly integrated with the platform. For instance, a turret may have a control system for its movement, which must be integrated with the targeting, fire control, and crew safety systems. Although the VDB scope does not include integration of those systems, information from those systems can be useful in interpreting C4ISR data. For example, knowledge of the relative azimuth and elevation

angles of a turret is necessary in order to know the pointing direction of a sensor mounted to it, and thus necessary to interpret data from that sensor.

Most of the sensors related to these systems are not network-enabled. The architecture defines the mechanical articulation joint to support the case in which data from the sensors can be captured and made available on the VDB, usually through a network adapter. The functions of the mechanical articulation joint component type include publishing data describing the joint state on the VDB, and possibly sending commands to indirectly control the joint. An example of such a sensor is a gimbal or turret that moves over time, on which a sensor may be mounted. It is possible for a sensor's frame of reference to be dependent upon multiple such articulation points, as when a camera with a gimbal is mounted on a turret. This component type is a generalization for sensing the offset and pointing direction of a movable joint.

Mechanical Articulation Joint Data Interface

The function of the mechanical articulation joint data interface is to allow for the offset and pointing direction of the movable joint to be published on the VDB. The network interface is not always integrated with the movable system itself, but may be accomplished via a network adapter that interfaces to the frame of reference sensor in its native format (e.g., serial, electrical signals), and publishes that on the VDB. Note that the network adapter may be a software component running on an SPU, with the SPU physically connected to the system through a serial port or other connection.

Mechanical Articulation Joint Management Interface

The mechanical articulation joint management interface provides functions for configuring, controlling, and managing the health (status and faults) of the sensor. The configuration and control functions allow for data reporting parameters to be set up from the VDB. This interface is not used to control the joint from the vehicle network, due to safety constraints. The health functions include querying status such as the current pointing direction, or reporting faults in the articulation system.

9.2 Lethality Systems Interfacing

Weapon and targeting systems are implemented on vehicle platforms for the purpose of identifying, tracking, and engaging and neutralizing the enemy. While the operation and control of these systems are clearly outside the scope of the VICTORY Architecture, it could be beneficial to define VDB interfaces for communicating with lethality systems in ways that do not affect their performance or attempt to control their operation. These interfaces are outside of the system fire control loop. Interfacing the VDB with weapon systems in this way can provide benefits, such as sharing data regarding the location and nature of threats, or providing the status of the weapon systems to applications on the VDB.

9.2.1 Weapon System

The VICTORY weapon system component type is a network adapter that allows a weapon system to be interfaced from the VDB network. See the definition of a network adapter in an

earlier section. The weapon system CT requires a physical VDB interface, as well as an interface to the weapon system itself. The inner workings of the weapon system are outside the scope of the VICTORY Architecture, and standards are therefore not identified for any messaging between weapon system components.

The primary functions of the weapon system component type include reporting the firing events, health, and status of the weapon system on the VDB, and to provide an interface for requesting the current configuration and state of the weapon, and the type and level of ammunition.

Note that the weapon system may access interfaces of other VDB, C4ISR/EW system, or platform system component types through its connection on the VDB. For example, the weapon system may receive recently detected shots from a threat detection system (see the section describing the threat detection system component type) to support advanced functions such as automatic slewing of the weapon in the direction of potential threats.

The weapon system component type is a generalization of part of the functionality of current weapon systems, such as the remote weapon station products currently fielded. The intent is to provide a component type that can be used for a wide variety of weapon types, including the RWS, a main gun, or a mortar system. Some of these systems may not have existing electronic interfaces for control or status reporting, but these can be adapted in the future. These weapons are often integrated with electronic instrumentation for test and training-related activities, which sets a precedent for adaptation.

Note that this component type is not meant to implement functions that are used directly in firing execution. These interfaces are intended to provide information to and accept information from other systems in a way that does not affect the weapon system's main functionality.

The weapon system component type has data, management, auto-discovery, and health reporting interfaces that follow the pattern of the other VICTORY CTs.

Weapon System Data Interface

The weapon system data interface is used to report the current weapon system state and firing events. The state and events include:

- State include the following:
  - Current weapon pointing direction
  - Change in the state of weapon and ammunition selector switches
- Firing events, including the following:
  - Trigger pulls and releases

Note that the function to report current pointing direction is similar to a similar function of the gimbal component type. These messages will be very similar, and possibly identical. Also, note that the function of reporting weapon and ammunition selector switches and firing events arguably fall into the configuration, control, and status categories. These messages may be

specified to be reported as part of the health management functionality of the management interface.

Weapon System Management Interface

The weapon system management interface follows the same pattern of other VICTORY CTs, providing functions for configuring, controlling, managing the health (status and faults) and data reporting mechanisms of the weapon system component type. The functions supported by the management interface include the following:

- Managing the configuration and health (status and faults) of the weapon system and the network adapter
    - Retrieving or updating weapon system configuration. Parameters include the following:
        - Weapon type (e.g., mortar, main gun, machine-gun, RWS, etc.)
        - Location and orientation of the weapon on the vehicle
        - Crew position from which the weapon can be used (e.g., commander, gunner)
        - Magazine info (e.g., capacity, low ammo indicator, level, type)
        - Ammunition info (e.g., types available, type selected)
        - Fire mission (the location of the current target being engaged)
    - Reporting or retrieving weapon status and faults. Status parameters include the following:
        - Current pointing direction
        - Settings or changes to safety, ready to fire, or other switches
        - Firing events (see comments in the data interface section
- Receiving inputs for actions (not direct control)
    - Accepting suggested slew directions
- Simulating weapon system operation
    - Controlling the simulation mode of the weapon.
        - For example, control the ability to “fire blanks.” If the vehicle simulated kill status is “dead,” this ability is disabled.
- Controlling weapon system test
    - Requesting, executing, and returning results of built-in test (BIT).
    - Requesting, executing, and returning results of calibration (special case of BIT).

9.2.2 Targeting System

The VICTORY targeting system component type allows a targeting system to be interfaced from the VDB network. The targeting system CT requires a physical VDB interface, as well as an interface to the targeting system itself. The inner workings of the targeting system are outside the scope of the VICTORY Architecture, and standards are therefore not identified for any messaging between targeting system components.

The primary functions of the targeting system component type include reporting the list of current targets and their tracks, allowing the list of targets to be managed (remove targets or add

locations of new targets), reporting the health and status of the targeting system, and providing an interface for requesting the current configuration, mode, state, and health of the targeting system.

Note that the targeting system may access interfaces of other VDB, C4ISR/EW system, or platform system component types through its connection on the VDB. For example, the targeting system may receive PNT information from the platform to augment its ability to interpret the geodetic coordinates of objects.

The targeting system component type provides network interfaces to a general targeting system. The intent is to provide a component type that can be used for a wide variety of targeting system types, such as the Long-Range Advanced Scout Surveillance System (LRAS3), the Tube-launched Optically-tracked Wire-guided (TOW) missile, the Improved Target Acquisition System (ITAS), and similar targeting systems.

Note that the network interfaces of this component type are not meant to be used in the safety critical fire control functions, which direct or control the firing of a weapon system. These interfaces are intended to provide information to and accept information from other systems in a way that does directly control a weapon.

The targeting system component type follows the same pattern as with other VICTORY CTs, including discovery, data and management interfaces.

Targeting System Data Interface

The targeting system data interface is used to report the information on objects of interest and targets. The functions include:

- Reporting current targeting information
    - Current objects of interest and targets
    - Tracks
    - Relative directions and lased distances
    - In some cases, geodetic locations
    - Object/target status
- Reporting or retrieving targeting information. Parameters included support:
    - Current pointing direction
    - List of targets
    - Currently engaged target information (e.g., absolute and/or relative location, type)

Targeting System Management Interface

The targeting system management interface follows the same pattern of other VICTORY CTs, providing functions for configuring, controlling, and managing the health (status and faults) and data reporting mechanisms of the targeting system component type. The functions supported by the management interface include the following:

- Managing the configuration and health (status and faults) of the targeting system and the network interface
    - Retrieving or updating targeting system configuration. Parameters include the following:
        - System type
        - Location and orientation of the targeting sensors on the vehicle
        - Crew position from which the targeting system can be used (e.g., commander, gunner)
    - Reporting or retrieving targeting system status and faults. Status parameters include the following:
        - Current sensor pointing directions
        - Object of interest and target events
- Receiving target management inputs
    - Accepting requests to observe an area of interest and identify and track objects of interest
    - Accepting requests to add targets (target "nominations")
    - Accepting requests to change the status of objects or interest or targets
    - Accepting requests to retrieve the current targets
- Simulating targeting system operation
    - Controlling the simulation mode of the targeting system
- Controlling targeting system test
    - Requesting, executing, and returning results of BIT
    - Requesting, executing, and returning results of calibration (special case of BIT)

9.2.3 Remote Weapon Station

The remote weapon station (RWS) component type was defined in the early versions of the VICTORY Architecture based on the capabilities of specific products that were evaluated. These RWS products include a weapon that can be controlled remotely by the warfighter, and also allow the warfighter to identify and track targets, and both report and accept targeting information on its external interfaces. These products are integrated systems that include special purpose computing and communication components, displays, weapons, articulation mechanisms, and targeting sensors. The RWS component type was developed as an adapter to provide a system-level interface to the VDB.

Over time it was observed that other products implement subsets of the RWS functionality in separate systems, namely remotely controlled weapon systems without targeting functionality, and targeting systems without integrated weapons. The weapon system and targeting system component types have been created to provide designers flexibility in leveraging these functions separately when needed, and combine them when appropriate.

It is notable that many of the weapon and targeting system products evaluated include the functionality of other VICTORY Component Types, such as the EO/IR sensor, EO/IR encoder,

and gimbal. These were already defined separately, which provides flexibility in implementation. Creating the weapon system and targeting system component types follows this more flexible pattern.

The RWS component type functionality is the combination of the functions of the weapon system and targeting system component types. Although instantiating those two component types separately is the preferred usage moving forward, the RWS component type is being supported in order to provide continuity for the community. It may be phased out in the future.

Remote Weapon Station Data Interface

The RWS data interface functions are the combination of those of the weapon system and targeting system data interfaces.

Remote Weapon Station Management Interface

The RWS management interface functions are the combination of those of the weapon system and targeting system data interfaces.

9.3 Protection Systems Interfacing

Many vehicle platforms host a variety of systems for detecting threats and automatically acting to protect the vehicle and crew. These can either avoid or reduce the damage incurred from a direct hostile threat, fire, or other conditions. This class of systems is currently expanding, particularly in the area of active protection systems (APSs).

The protection systems interfacing capability group encapsulates the component types with functions necessary to interface with these systems, but not to integrate of them. Note that in this statement, the differentiation between “interface with these systems” and “integrate them” must be interpreted in the context of the statements in section 3.2, which defines the different between *enabling integration* and *enabling system-level interfaces*. Here interface and integrate are used for brevity.

9.3.1 Threat Detection System

Threat detection systems are responsible for detecting and reporting active threats to the vehicle platform and to the Warfighter. Examples of threat detection systems that are currently being deployed on vehicles include acoustic or radar-based sensing devices that detect and report on various types of threats. While many of these systems are being fielded on current force platforms, they have either very limited or no integration with other C4ISR/EW systems. The Warfighter could benefit greatly from integration of these systems, and the VICTORY interfaces that follow provide a framework for this integration.

A number of different components may be used to implement a platform’s Threat Detection System capabilities. Because the technologies that are utilized by these components are constantly changing, VICTORY does not attempt to standardize the components themselves.

However, VICTORY specifies that these components must have a VDB interface, at some point on the system, to share information with other VICTORY nodes. The following sections describe the logical standards that are to be implemented on this VDB interface.

The function of the Threat Detection System Interface component type is to acquire threats from various sensors, and report the threats on the VDB in a standard format. It includes a data interface for reporting the threats, and a management interface for managing the service. The back-end interface to existing "current force" threat detection sensors are not standardized. Such existing sensors can be integrated using adapters.

Threat Detection System Data Interface

This interface is used to transport information, including items such as the azimuth, elevation, and distance of a recently detected threat with respect to the vehicle.

Threat Detection System Management Interface

The threat detection system management interface provides functions for configuring, controlling, and managing the health (status and faults) of the threat detection system. This includes configuration parameters. Functionality of the threat detection system component type may be altered via the control functions of the management interface, such as stopping and restarting threat detection services, as well as selecting which VICTORY nodes have access to threat data. The status of the threat detection system component type is available to authorized nodes on the VDB.

9.3.2 Active Protection System

The Active Protection System CT provides high-level interfaces to systems that are designed to automatically react to and in some cases, neutralize a threat, such as an incoming rocket-propelled grenade (RPG). Although these systems are not currently commonly fielded, there are efforts to bring the emerging technologies to the level of maturity that they can be fielded. For example, the U.S. Army is performing research to accelerate the development of new Modular Active Protection Systems (MAPS) technologies for combat vehicles, which may expedite the fielding of these systems.

Active Protection System Data Interface

The data interface for such a system includes the passive reporting of data from the detection sensors (e.g., Radar), whether or not the system has been triggered, and of course detected threats. This concept must be developed further by engaging the MAPS community.

Active Protection System Management Interface

The active protection system management interface enables the health (status and faults) reporting and logging, and system mode and state monitoring, and perhaps sending high-level, low-speed commands to the system.

9.4 Logistics Systems Interfacing

Logistics systems monitor and manage the configuration, health, and usage of the platform. Users of the logistics system and its data include the vehicle operator (e.g., health alerts and supply status), the vehicle maintainer (e.g., trouble-shooting automation, interface to maintenance management system), fleet managers (e.g., storage and off-load of logistics data to support Reliability Centered Maintenance [RCM] analysis and CBM), and the mission commander (e.g., automated combat power reporting). The exchange of data between a platform's logistics systems and networked sustainment systems has been defined through Common Logistics Operating Environment (CLOE) processes and interfaces. These processes include Logistics Common Operating Picture (LCOP), log transactions, reliability, availability, and maintainability, vehicle configuration management (CM), and predictive maintenance.

The logistics systems, their algorithms and implementation are outside of the scope of VICTORY. However, VICTORY may specify network interfaces that enable collection of data and management of systems in a way that supports logistics processes and lessen the work to be done to field a logistics system on a vehicle with a VDB.

The Logistics Systems Interfacing capability group consists of component types that aim to define open interfaces that allow logistics processes to be greatly optimized.

9.4.1 Vehicle Configuration Service

The function of the vehicle configuration service component type is to store and manage the "as designed" and "as maintained" configurations of the vehicle and the systems integrated with it. This service provides interfaces that follow the VICTORY interface pattern, including management, auto-discovery, and health reporting interfaces, but does not include a data interface. The management interface provides functions for configuring, controlling, and managing the health (status and faults) of the service and the configuration data.

<u>Vehicle Configuration Service Management Interface</u>

The vehicle configuration service management interface provides similar functions to other VICTORY CT management interfaces, including the following:

- Retrieving and updating the current vehicle configuration data
- Retrieving historical vehicle configuration data on request
- Logging changes to the vehicle configuration data
- Generating reports of government furnished equipment (GFE) and status (e.g., security report for crypto controlled items)

Vehicle configuration includes information such as:

- As designed configuration (bill of materials)
    - Crew seating configuration, and role locations
    - Product structure to replacement parts
- As maintained configuration (what you actually have)
    - Model

- Serial number and/or bumper number
- Engine and transmission serial number
- A-kit options
- GFE and theater provided equipment (TPE) that has been added

9.4.2 Vehicle Health and Usage Monitoring Service

The function of the Vehicle Health and Usage Monitoring Service (VHUMS) component type is to track, document, and otherwise manage the health and usage of the vehicle and its integrated systems. This service provides interfaces that follow the VICTORY interface pattern, including data, management, auto-discovery, and health reporting interfaces.
The functions implemented by the VHUMS component type include the following:

- Receiving messages from other VICTORY component types, such as the automotive system, power distribution system, etc.
- Interpreting the contents of those messages to "calculate" vehicle health and usage reports
  - This behavior is vehicle / implementation dependent, so cannot be standardized
- Publishing VHUMS reports to the VDB
- Generating a VHUMS report on request
  - E.g., 5988 maintenance and 5987 dispatch reports

Vehicle Health and Usage Monitoring Service Data Interface

The VHUMS data interface publishes vehicle health and usage reports on the VDB.

Vehicle Health and Usage Monitoring Service Management Interface

The VHUMS management interface provides functions for configuring, controlling, and managing the health (status and faults) of the service and the underlying data.
The concept of setting up conditions that generate alarms (events) related to vehicle health and usage has been discussed, but is not currently supported by the architecture.

9.5 Training Systems Interfacing

The Training Systems Interfacing capability group consists of CTs that aim to allow the training community to leverage existing sensors and systems on the vehicles, and to use the VDB for integrating their test systems, thus making their operations more efficient.

9.5.1 Live Training Transformation Adapter

Live training transformation (LT2) is a training community initiative for developing an architecture to integrate training systems, including devices such as indicator panels, effects controllers, instrumentation devices, and tactical engagement simulation (TES) systems and devices. These devices are integrated by training networks, protocols and/or standards, but it may be advantageous for those systems to provide data to and/or receive data from the in-vehicle network integrated with VICTORY Standard Interfaces.

The LT2 Adapter component type is intended to define the interfaces through which LT2 systems may interoperate with C4ISR/EW and platform systems integrated with a VDB using VICTORY standard interfaces.

LT2 encompasses several training systems / devices including, but not limited to:

- Visual indicators
- Laser detectors
- Lasing systems: Laser Engagement Transmitters/Lasing Systems (already covered in Lethality System types)
- Laser sensors and repeaters (e.g., MILES optical repeater and emitter unit)
- Effects generators/actuators (also known as signature devices), including:
    - Illumination actuators (e.g., vehicle kill indicator, flash weapons effects simulation system [WESS])
    - Pyrotechnic effects generators (direct fire/indirect fire CUE [DIFCUE] and other pyrotechnic systems such as WESS)
    - Haptic effects generators (e.g., for providing tactile cues to the crew in/on vehicle)
- Instrumentation systems (e.g., training data that is sent off the platform via training system radios)
- Battle damage assessment systems (e.g., vehicle kill status, casualty assessment)
- Weapon sensors (e.g., breach, hatch, weapon orientation, or trigger sensors)
- Vehicle sensors
- Surrogate rounds
- Proximity sensors
- TES-specific user interface functions
    - Loader interface
    - Combat engineer interface simulated equipment repair
    - Training observer/controller functions (e.g., observer/controller tablet)

The LT2 adapter component type interfaces support transport of existing LT2 system messages (e.g., personal area network (PAN)) on the VDB. The LT2 adapter does not attempt to convert the LT2 system messages from and to other VICTORY message types, but instead encapsulates and transports the messages, acting as a delivery mechanism as opposed to a message conversion service.

The functions implemented by the LT2 adapter component type include the following:

- Allow existing training systems to leverage the VDB network to transport and deliver LT2 messages between LT2 components
    - Interface with existing wired and wireless training system protocols to receive LT2 messages from sources
    - Encapsulate LT2 messages and routing them from sources to destinations
    - De-encapsulating the LT2 messages at the destinations
    - Interface with the training system protocols to deliver to the appropriate destinations
- Retrieving and updating the LT2 Adapter configuration

- Controlling and monitoring LT2 TES devices and systems

The LT2 adapter component type receives data from training system component output interfaces attached to the VDB in the native training system messaging protocol, encapsulates the training messages into special VDMs, and publishes the encapsulated training data on the VDB. It also subscribes to training system VDMs, extracts training messages from the VDMs, and provides the messages in the native training messaging protocols to training system component input interfaces. This allows the training systems to leverage the VDB for transport of messages between its components.

The LT2 adapter component type includes discovery, data and management interfaces.

Live Training Transformation Adapter Data Interface

The LT2 adapter data interface publishes and subscribes to training VDMs on the VDB, routing the messages from source to destination in the LT2 system.

Live Training Transformation Adapter Management Interface

The LT2 adapter management interface provides functions for configuring, controlling, and managing the health (status and faults) of the LT2 adapter.

9.5.2 Lasing System

A lasing system consists of a laser transmitter and other necessary components and functions in order to provide operational capabilities such as laser range finding, and laser illumination to support tactical operations such as cooperative engagements. Lasing systems may also support dual-use functionality, such as a laser rangefinder that can also serve as TES laser transmitter. For example, the small tactical optical rifle mounted (STORM) – micro-laser rangefinder (MLRF) supports both MILES training as well as operational laser range finding.

The function of the lasing system component type is to provide standard network interfaces to a lasing system to promote interoperability between test and training system components. This component type is similar to the targeting system component type in the lethality systems component types group, but is different in that it is more generalized. It is not necessarily part of a weapon system, but may be used in conjunction with a weapon system for test or training operations. This component type provides data, management, auto-discovery, and health reporting interfaces. The data interface publishes information such as the current pointing direction of the laser. The management interface provides functions for configuring, controlling, and managing the health (status and faults) of the lasing system interface and the underlying laser system.

The Lasing System CT includes discovery, data, and management interfaces.

Lasing System Data Interface

The lasing system data interface publishes lasing system reports on the VDB. These reports include the following:

- Lasing action report
  - Action time and duration
  - Pointing direction
  - Lasing configuration parameters

Lasing System Management Interface

The lasing system management interface supports the following functions:

- Managing the lasing system configuration
  - Storing, retrieving, and updating the configuration data
  - Lasing system configuration parameters include:
    - Attachment point on the vehicle or relationship to a weapon
    - Laser power, frequencies, codes, and pulse count settings
- Controlling the lasing system
  - Controlling the power on/off, lase/stop lasing
  - Accepting nominations of locations to point to ("possible targets")
  - Controlling pointing direction ("slew to cue")
- Reporting status and health of the lasing system

9.5.3 Laser Detection System

The function of the Laser Detection System component type is to provide a network interface for publishing data from sensors with laser detection capabilities on the VDB. A laser detection system includes one or more laser detection sensors embedded in the platform to support dual-use functions such as active protection and MILES-based TES. Another example is a set of appended laser detection systems used to support TES-based Live Training or ordnance-free testing.

The network interface is not always integrated with a laser detection system itself, but may be accomplished via a network adapter that interfaces to the laser detection system in its native format, such as serial, universal serial bus (USB), wireless personal area network (WPAN), and raw electrical signals, and publishes data on the VDB. Note that the network adapter could be a software component running on an SPU, with the SPU physically connected to the system through a serial port or other connection.

This component type provides data, management, auto-discovery, and health reporting interfaces. The data interface publishes information such as detection events. The management interface provides functions for configuring, controlling, and managing the health (status and faults) of the laser detector interface, and of and the underlying sensor.

The Laser Detection System CT follows the pattern of other VICTORY CTs, including discovery, data, and management interfaces.

Laser Detection System Data Interface

The function of the laser detection system data interface is to report laser energy detections on the VDB.

Laser Detection System Management Interface

The functions of the laser detection system management interface include:

- Viewing and configuring (updating) sensing and sensor system interface parameters, such as:
    - Data reporting configuration parameters
    - Location on the vehicle, or current position (e.g., for appended detectors)
    - Role of the sensor to a test or training activity
    - Laser sensitivity
    - Operational and logistics status (e.g., hours of use/operation)
    - Faults in the laser detectors

10. VICTORY COMPOSITION GUIDELINES

As described in Section 1.2, an architecture defines a common glossary (well-defined terminology), a type system (modules and interfaces), and system composition guidelines (common structures, patterns, and constraints).

In the case of the VICTORY Architecture, the common glossary consists of the terms, names, and titles defined in this document. The type system is the set of the VICTORY component types and their interfaces. This document provides the terminology and the component types in sufficient detail to provide a conceptual framework for the VICTORY Standards Body to develop technical specifications for the component types and their interfaces.

The third element of an architecture, system composition guidelines, is not completely described in the VICTORY Architecture document. The VICTORY composition guidelines are described in a companion document titled "VICTORY Implementation Guide."

The rationale for documenting implementation guidance separately is to allow the guidelines to be updated asynchronously, and more frequently as needed. Implementations instantiate portions of the architecture based on specific program requirements, and as unique situations arise the lessons learned can be added to the guidelines without changing this document.

The VICTORY Implementation Guide provides additional information to the community as they develop designs, implement artifacts, and integrate systems.