2018 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
VEHICLE ELECTRONICS AND ARCHITECTURE (VEA) AND GROUND SYSTEMS CYBER ENGINEERING (GSCE) TECHNICAL SESSION
AUGUST 7-9, 2018 - NOVI, MICHIGAN

ENABLING IMPLEMENTATION AND DEPLOYMENT UTILIZING VICTORY TOOLS

Joshua Klein
Southwest Research Institute
San Antonio, TX

Adam Thornton
Southwest Research Institute
San Antonio, TX

Leonard Elliott
U.S. Army Tank Automotive Research Development and Engineering Center (TARDEC)
Warren, MI

ABSTRACT

The VICTORY Service Toolkit (VSTK) and libVictory are software products developed to simplify the implementation and deployment of Vehicular Integration for Command, Control, Communications, and Computers (C4), Intelligence, Surveillance, and Reconnaissance (ISR) Electronic Warfare (EW) Interoperability (VICTORY) Standard compliant components. The libVictory software library provides a well-defined application programming interface (API) that can reduce time and costs for developing VICTORY enabled software and hardware products. The VSTK wraps libVictory with a plugin architecture and streamlines deployment by providing a VICTORY service management application. Together these government-owned tools enable deployment of configurable VICTORY applications on platforms without the need for new software development.

INTRODUCTION

The adoption of new standards and protocols is often met with resistance due to the cost of designing, developing, and testing the new software and hardware required to implement them. For Vehicular Integration for Command, Control, Communications, and Computers (C4), Intelligence Surveillance and Reconnaissance (ISR) Electronic Warfare (EW) Interoperability (VICTORY), reusable software tools, such as libVictory and the VICTORY Service Toolkit (VSTK), have been developed to lessen the effort required to achieve compliance and benefit from the interoperable infrastructure that VICTORY provides. An approach using libVictory and the VSTK can be utilized to adapt between legacy, fielded equipment such that minimal modification of the equipment is needed. The software runs on a shared processing unit (SPU) that is connected to equipment through legacy physical interfaces such as RS-232, RS-422, RS-485, or MIL-STD-1553. These tools act as a software bridge between the data received and transmitted over legacy connections to VICTORY compliant messaging and command and control over Ethernet.

BACKGROUND

VICTORY

The VICTORY initiative is led by the VICTORY Standards Support Office (VSSO), which is hosted by the U.S. Army Program Executive Office for Ground Combat Systems (PEO GCS) Systems Engineering & Integration (SE&I) organization. The VICTORY standard is designed to increase interoperability between systems on Army ground vehicles.

Electronics systems on Army ground vehicles are traditionally installed with a “bolt-on” approach where each subsystem brings its own collection of hardware. This approach often results in redundant sensors, computers, displays, and cabling, which increases the size, weight, and power (SWaP) impact on the vehicle. VICTORY strives to replace this approach with an environment where data can be shared between systems over the VICTORY Data-Bus (VDB), a communication infrastructure based on ubiquitous Ethernet technology. This eliminates unnecessary duplication of equipment and processing resources.

VICTORY defines various “component types” that provide specific data and interact with specific types of equipment.

For example, the VICTORY GPS Receiver component type publishes GPS data that it receives by interacting with a GPS device, such as a Defense Advanced GPS (Global Positioning System) Receiver (DAGR). Supporting a VICTORY component type allows data sharing and remote control over the corresponding system or device.

Because of the need to share data with other systems on the network, VICTORY has well-defined data interfaces. Data is encapsulated in VICTORY data messages (VDM) and published on the VDB using multicast user datagram protocol (UDP). The process to create and publish VDMs is very similar across all VICTORY component types so this functionality can easily be implemented in a software library where it can be reused for multiple component types and by any developer.

VICTORY also defines a way to control component types, and in some cases the devices with which they interact, via the VDB. For most cases this control is accomplished through sending Simple Object Access Protocol (SOAP) requests to the component type. While the content of the requests varies between component types, the mechanisms to send these requests are the same, which allows for a common software library. In addition, the structure of these requests for a specific component type will be identical so that a software library or class can be used for each instantiation of that component type.

Another major portion of implementing a component type is the configuration description and handling of configuration management requests. Handling configuration management requests is one of the more complicated tasks required for most VICTORY component types. Unlike other pieces of the management interface, the configuration description was created to be more flexible so its schema is less constrained and not well-suited for validation by commonly available schema validation tools. Since this task does not benefit as much from code generated from the Web Services Description Language (WSDL) and schema, more software development is needed. This undertaking can benefit greatly from having a shared library.

LIBVICTORY

libVictory is a software library developed by the U.S. Army Tank Automotive Research Development and Engineering Center (TARDEC) that can dramatically decrease the effort required to instantiate VICTORY clients and services. The libVictory software is written in C++ and provides an API written in C, which supports bindings to other programming languages such as Java. libVictory is government open-source software and is licensed under the DoD Community Source Usage Agreement version 1.1 (Distribution C with Export Control Restrictions). libVictory 1.0 was released in 2015 and there have been approximately 10 releases supporting the VICTORY 1.6.1 and 1.6.2 specifications. A listing of libVictory-supported VICTORY component types is shown in Table 1 and, prior to release, these software components are implemented with service stubs and tested for compliance using the VICTORY Compliance Test Tool (CTT). The libVictory software project is hosted on the Defense Intelligence Information Enterprise site (DI2E) [1] and releases can be obtained through the VICTORY portal [2].

Component Type	Service	Client
Authentication		✓
Automotive System	✓	✓
Camera Gimbal	✓	✓
Data Logger	✓	✓
Direction-of-Travel	✓	✓
GPS Receiver	✓	✓
Intercom	✓	✓
Orientation	✓	✓
Policy Enforcement		✓
Position	✓	✓
Power Distribution System	✓	✓
Remote Weapon Station	✓	✓
Shared Processing Unit	✓	✓
Threat Detection and Reporting	✓	✓
VDB Management	✓	✓
Vehicle Configuration	✓	✓
Video or Image Encoder	✓	✓
Video or Image Sensor	✓	✓

Table 1: libVictory Supported VICTORY Component Types

A significant portion of the work required to build VICTORY services and clients is related to implementing SOAP request/response communications and interpretations of the VICTORY Configuration Language Instance Document. libVictory implements the network sockets, serialization/deserialization, and processing logic and exposes callbacks and data through the libVictory API. This allows users/developers to focus on implementing their business logic and shields them from complexities of implementing the VICTORY protocol stack as shown in Figure 1.

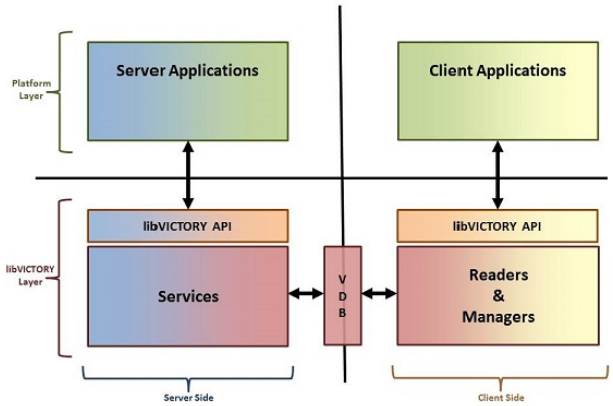

Figure 1: libVictory Software Design View [1]

Security features supported by libVictory include the ability to configure encryption and signing via Transport Layer Security (TLS) and XML digital-signing, and advanced security configurations can make use of the VICTORY Access Control Framework, which greatly increases the granularity of security policies that can be implemented within a system.

libVictory is under active development with increases in regression test coverage, bugfixes, and new improvements being regularly added. The newest release of libVictory will be version 2.5.0, which will include support for the VICTORY 1.7 Active Protection System Service, and is slated for release in August 2018.

VICTORY Service Toolkit

Software Adapter Overview

While libVictory is aligned to become the standard VICTORY web services library that can be used directly by hardware vendors to provide future equipment procurements with a native VICTORY interface, modifying existing hardware tends to be cost prohibitive. One of the current approaches of VICTORY enabling current force equipment is to install the VICTORY services on an SPU to which the existing hardware connects through legacy interfaces. Software running on the SPU converts the legacy interfaces to the appropriate VDMs and provides the health and management interfaces. Figure 2 illustrates an example design for a VICTORY software adapter.

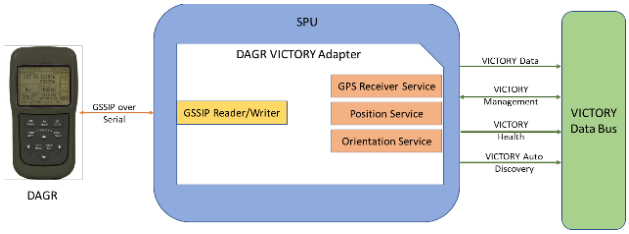

Figure 2: Example VICTORY Software Adapter

This example includes the DAGR for which the deployed units number in the thousands. The DAGR connects to the SPU through a serial connection. An application on the SPU reads the GPS Standard Serial Interface Protocol (GSSIP) messages from serial and converts them to VICTORY compliant data messages. It also provides access to the DAGR's command and control functions through standard VICTORY management interfaces.

With this approach, a driver must be developed for each unique device along with an executable application that will instantiate the driver and adapt the device's data to VICTORY standard interfaces, perhaps via libVictory. This means for each device there must be a separate application, driver, and adapter developed. Since there are multiple vendors on a platform, there will be different installation requirements and processes for each software device adapter. This results in the platform maintainers needing to have knowledge on how to deploy and maintain a variety of applications. There may be one application per unique piece of hardware that has a VICTORY software adapter.

VSTK Application

The VSTK fills the gap between libVictory and the platform devices as both an application programming interface (API) for implementing device drivers and an application as shown in Figure 3.

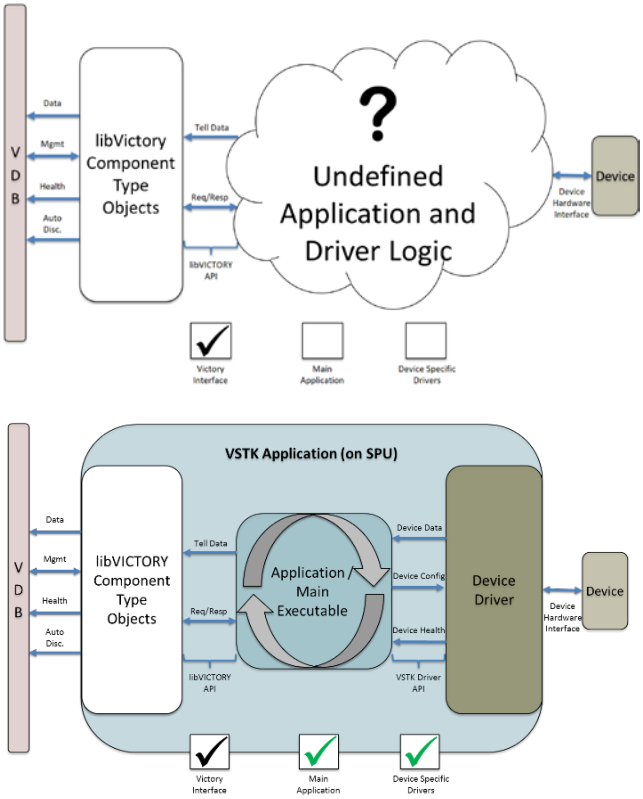

Figure 3: VSTK: Complete Picture

As an API, it defines standard interfaces for the various types of subsystems that are supported by the VICTORY standard, for instance, GPS receivers, EW devices, automotive buses, etc. As an application, it provides a single program from which any number of VICTORY services can be configured and started, thus reducing the training requirements for the platform maintainers.

For any number of device and VICTORY service combinations, the VSTK runs as a single executable yet there is a thread per device driver or VICTORY service adapter. The VSTK has been designed to provide a plugin-like

architecture for both the VICTORY service adapters and the device drivers as portrayed in Figure 4.

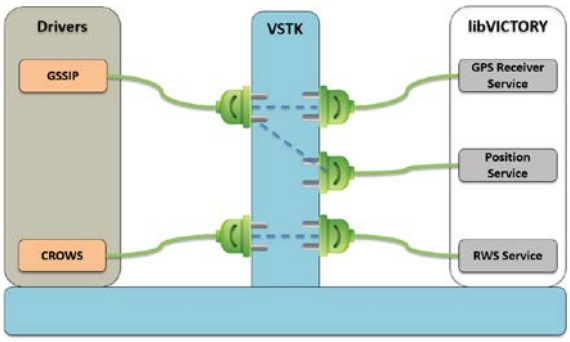

Figure 4: VSTK Plug-in Architecture

The plugin architecture allows the device driver and VICTORY service adapter combinations to be designated at any point without the need to recompile the application. Currently, the shared object (SO) files for the driver and adapters reside in a specific folder in the operating system. At any time, additional device drivers or adapters can be added to this folder at which point recompiling the main application is still not necessary to incorporate the new drivers and adapters into the application. Another benefit of the plugin architecture is that once the main VSTK application has been tested and verified it should not need to be modified, compiled, and thus, re-tested and verified again.

Configuring an installation of the VSTK to "connect" device drivers to VICTORY service adapters is relatively simple. A single ".ini" configuration file allows for customizing which services are connected to which drivers as shown in Figure 5.

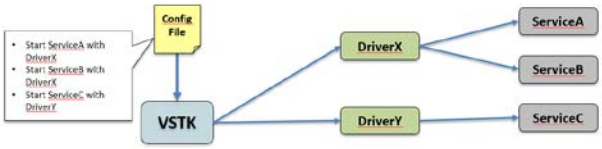

Figure 5: Configuring VSTK

A device driver may be connected to multiple VICTORY service adapters. For instance, a GPS driver can be connected to the GPS Receiver service adapter, the Position service adapter, and the Orientation service adapter. With such an example, there is only one instance of the GPS driver running, but the data output from the driver is shared with three service adapter instances. Internally, this is achieved using a publish/subscribe mechanism. Services are publishers while adapters are subscribers. In the GPS Receiver adapter example, the GPS driver is the publisher. Once data is received from the GPS device and processed, the GPS driver will notify the GPS Receiver, Position and Orientation service adapters of an update to the data structures they are interested in.

VSTK APIs

The VSTK API defines two types of interfaces: device drivers and VICTORY service adapters.

The driver API is the high level abstraction for each type of subsystem on a platform. It defines the data structures to which the device driver instantiations must load the data to later be consumed by one or more VICTORY service adapter instantiations. While the device driver API is delineated by VICTORY component types, a driver must be developed for each unique piece of hardware even if they map to the same VICTORY component type. For instance, while AN/VLQ-12 Counter Remote Controlled Improvised Explosive Device (RCIED) Electronic Warfare (CREW) Duke and a CREW Vehicle Receiver/Jammer (CVRJ) are both devices that map to the EW VICTORY component type, they have different legacy protocols and interfaces. Therefore, each will need to implement a driver specific to that device, yet both extend the EW device driver API.

The service adapter API adapts the driver instantiations to the VICTORY service library API (i.e. libVictory). An adapter instantiation is responsible for mapping and converting the data received from a driver instantiation to the relevant VICTORY interfaces. For instance, when dealing with GPS and position data the information is handled in both radians and degrees. A GPS receiver or position adapter is responsible for converting the data received from a GPS driver to the type that is required by a VICTORY service library's GPS receiver or position service.

The two APIs reinforce the software design principle of separation of concerns. Driver implementations are strictly concerned with connecting to and reading data into data structures provided by the driver API. The adapters take the data from those structures and convert it to the data types and ranges required by the VICTORY service library. Additionally, the adapters perform analysis on the data to determine if any of the data is indicative of fault conditions and notifies the VICTORY service library of said faults, which will in turn generate fault notifications.

VSTK Version 1

Southwest Research Institute® (SwRI®) has developed a fully functional implementation of the main VSTK application, which includes the plugin framework and configuration with an "ini" file. This implementation was developed during a GPS Receiver implementation effort where the DAGR was utilized as the legacy device. The current version includes the device driver API for GSSIP based GPS devices along with the GPS Receiver, Position, and Orientation service adapter APIs.

Future Versions

Future efforts will continue to define the remaining device driver APIs and VICTORY service adapter APIs for the remaining VICTORY component types that have not previously been defined. Since there is no dependency between APIs across the component types, these can be defined individually as the need arises.

Since one goal of the VSTK is to reduce deployment and maintenance needs for VICTORY services, a graphical user interface (GUI) is needed with which a technician can install plugins and graphically establish the device driver to service adapter connection. This removes the need for the technician to know where the plugin SO files are saved and how to edit the file. Additional features of the GUI would include the ability to start and stop VICTORY services on the fly without restarting the main VSTK application.

CONCLUSIONS

The VSTK and libVictory are tools that enable deployment of configurable VICTORY applications on platforms without the need for new software development. They reduce the knowledge gap and costs necessary to design and deploy VICTORY-enabled platforms.

Since the VSTK has not been formally released, it is not currently available for distribution. For more information on the VSTK please contact Adam Thornton at adam.thornton@swri.org.

REFERENCES

[1] TARDEC-VEA, “libVictory Release Notes & API Documentation” [Online] Available: https://confluence.di2e.net/display/LIBVICTORY April 2017.

[2] VSSO, “VICTORY Reusable Software Tools” VSSO, [Online] Available: https://victory-standards.org/index.php/vic-port May 2018.