CDRL A005

Software User’s Manual (SUM) for

Advanced Modular Manikin Project

Phase II Program

Contract # W81XWH-14-C-0101



Date: February 3, 2020

Revision: Rev - 2

Unclassified

PRINCIPAL INVESTIGATOR: ROBERT M. SWEET, MD, FACS

REGENTS OF THE UNIVERSITY OF MINNESOTA

OFFICE OF SPONSORED PROJECTS

200 OAK ST SE

MINNEAPOLIS MN 55455-2009

This document is unclassified, and APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

Revision History

|  |  |
| --- | --- |
| Date of Revision | Description of Changes |
| 24 Dec 2018 | Draft Document Submitted |
| 25 Oct 2019 | Document Submitted |
| 03 Feb 2020 | Distribution Updated |

**Distribution Statement**

This document was created under contract funding W81XWH-14-C-0101 from:

**DEPARTMENT OF THE ARMY**

**US ARMY MEDICAL RESEARCH ACQUISITION ACTIVITY**

**820 CHANDLER STREET**

**FORT DETRICK MD 21701-5014**

It contains no proprietary information, trade secrets, copy righted material or classified information. This document is unclassified, and APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

**Table of Contents**

[1 Scope 4](#_Toc22906696)

[1.1 Identification 4](#_Toc22906697)

[1.2 System overview 5](#_Toc22906698)

[1.3 Document Overview 5](#_Toc22906699)

[2 Referenced Documents 5](#_Toc22906700)

[2.1 Industry Documents 5](#_Toc22906701)

[2.2 Government Documents 6](#_Toc22906702)

[2.3 Related Contract Documents 6](#_Toc22906703)

[3 Software Summary 6](#_Toc22906704)

[3.1 Software Application 6](#_Toc22906705)

[3.2 Software Inventory 9](#_Toc22906706)

[3.3 Software Environment 9](#_Toc22906707)

[3.4 Software Organization and Overview of Operation 10](#_Toc22906708)

[3.5 Assistance and Problem Reporting 10](#_Toc22906709)

[4 Access to Software 10](#_Toc22906710)

[4.1 First Time user of Software 11](#_Toc22906711)

[4.1.1 Equipment Familiarization 11](#_Toc22906712)

[4.1.2 Installation and Setup 11](#_Toc22906713)

[4.2 Initiating a Session 11](#_Toc22906714)

[4.3 Stopping and Suspending Work 12](#_Toc22906715)

[5 Processing Reference Guide 12](#_Toc22906716)

[***Figure 1: Functional Overview of AMM Platform. 4***](#_Toc22906738)

[***Figure 2: Modular Manager. 7***](#_Toc22906739)

[***Figure 3: Web Application Flow. 8***](#_Toc22906740)

[***Figure 4: Example web interface. 11***](#_Toc22906741)

# Scope

This document defines the standards for 1.0 release of the Advanced Modular Manikin (AMM) platform and its formal deliverables. The formal deliverables consist of the platform specification, an open source\* Reference Implementation (RI) of the Computer Software Configuration Items (CSCIs), a reference implementation of the Universal Segment Connector (USC) and other hardware defined by the Hardware Configuration Items (HWCIs), the data models that ensure interoperability between the core and modules, and the documents that describe their design, operation, and extensibility through the addition of AMM Modules. Modules are defined as independent building blocks that provide incremental capabilities to the core or provide training opportunities for different medical and trauma related conditions. The focus of this specification is on the platform, a much broader definition than a physical manikin, as illustrated in Figure 1, and on how it can be extended by medical simulation developers by adding:

* Modules that provide incremental capabilities to the core, including authoring tools, after action review tools, different physiology engines.
* Modules that add training opportunities, including IV/IO arms, intubation heads, laparotomy abdomens, virtual stethoscopes. These can be physical, virtual, or hybrid part task trainers.



**Figure 1: Functional Overview of AMM Platform.**

## Identification

This is the Advanced Modular Manikin (AMM) Software User’s Manual (SUM) CDRL Item A005 of Contract # W81XWH-14-C-0101.

This CDRL is formatted to the requirements of Data Item Description Number DI-IPSC-81443A as required. This document is a Software User Manual which identifies the necessary information for a module supplier to successfully interface to software in the manikin.

## System overview

The AMM platform is a modular, distributed, interoperable system that enables physical, virtual, augmented and hybrid modules to work together as an integrated system. The traditional “core”, i.e. computer and state engine, can be in any one of the traditional manikin segments, i.e. torso, leg etc., or external to the human form, as it would be if the system is only running a virtual instance or if the targeted scenario, i.e. patient case, does not allow them to be internal due to the set of interventions that have to be performed on the body. The platform is architected as a system of systems that allow modules to function either as part of an integrated, whole body simulation or as autonomous part task trainers.

The published AMM standards guide the development and integration of AMM compatible modules. The reference designs provided for the final demo including electronics and central supplies were created to demonstrate the operation of the platform and are published as a developer’s tool kit with sources to acquire them from.

The developers of the platform have agreed to publish the AMM platform under the following open source licensing option:

\* *Creative Commons Attribution 4.0 International (CC BY 4.0)* [*https://creativecommons.org/licenses/by/4.0/deed.ast*](https://creativecommons.org/licenses/by/4.0/deed.ast)*.*

*Share — copy and redistribute the material in any medium or format*

*Adapt — remix, transform, and build upon the material for any purpose, even commercially.*

*The licensor cannot revoke these freedoms as long as you follow the license terms.*

This document does not cover modules that were created under separate funding and by other entities to demonstrate the functionality of the AMM Platform under separate funding and are not part of the Open Source agreement.

## Document Overview

This is the final version of the Software User’s Manual as required by CDRL A005. The outline and subject matter content are based on DID DI-IPSC-81443A as required by the contract. The DID has been tailored as appropriate. This document is unclassified and contains no proprietary information, trade secrets, copy righted material or classified information. Unlimited distribution.

# Referenced Documents

## Industry Documents

|  |  |
| --- | --- |
| Doc. No. | Title |
| 802.3x | IEEE Standard for Ethernet  |
|  | Data Distribution Services (DDS) |

## Government Documents

|  |  |
| --- | --- |
| Document Number | Title |
| W81XWH-14-C-0101 | AMM Phase II Contract, DOD |
| DI-IPSC-81443A | Data Item Description |
|  |  |

## Related Contract Documents

|  |  |
| --- | --- |
| Document Number | Title |
| CDRL A001 | Software Design Description  |
| CRDL A007 | Interface Design Description (IDD) |

# Software Summary

## Software Application

The AMM CORE software is provided as a series of required, optional and reference modules.

**AMM Standard Library**

The AMM Standard Library has been developed as a reference implementation of the AMM Data Model definition, exposing APIs to allow developers to quickly develop AMM-compliant modules without having to work directly with the data bus itself. The Standard Library has been developed utilizing eProsima FastRTPS for DDS middleware.

The standard library offers API interfaces to:

* Connect to an AMM network utilizing the ‘standards-compliant’ handshake, including publishing capabilities, receiving configuration and publishing operational status.
* Subscribe to AMM Data Model topics
* Publish to AMM Data Model topics
* Publish diagnostic logging information
* Resolve Event Fragments and Event Fragment Amendment Requests

**AMM CORE Module Design – Required Modules**

1. Module Manager

The Module Manager is a core software component that coordinates the participation, initialization, configuration, and termination of AMM modules during the educational encounter. All AMM compliant modules must perform an appropriate handshake procedure which includes information about the module, the capabilities it provides, and the configuration data needed to provide those capabilities. The Module Manager is also responsible for loading scenarios and publishing the configuration data specific to each module to enable the capabilities required by the scenario. The Modules must validate their configuration and report the operational status of each of their enabled capabilities. The Module Manager will aggregate the operational statuses of all the modules to determine if all the required capabilities of a scenario are available and operational. Figure 2 details this process.



**Figure 2: Modular Manager.**

In the reference implementation of the module manager we have included a data logger which collects both AMM-specific data and out-of-band diagnostic data and places it into persistent storage for analysis and review.

1. **Simulation Manager**
The Simulation Manager is a core software module that drives the simulation by publishing simulation ticks at a frequency of 50 Hz. It operates as a simulation state engine, allowing for control of time-related functions and loading scenarios for execution.
2. **Physiology Engine Manager**
The Physiology Engine Manager is a core software module that connects to a physiology engine. At this time, BioGears is the only supported physiology engine. A wrapper module is provided as an exemplar and guide for how other physiology engines might be connected to AMM.

**AMM CORE Module Design – Optional Modules**

1. **REST Adapter**
The REST Adapter is a web service that converts HTTP Representational State Transfer (REST) requests to DDS messages in support of web browser-based modules like the Dashboard. A web application can be built on top of AMM utilizing the REST services. See Figure 3 below.



**Figure 3: Web Application Flow.**

1. **TCP Bridge**
The TCP Bridge is a socket server that handles TCP communications with socket clients and converts them to DDS messages in support of modules which cannot natively implement DDS but can open network sockets such as the virtual patient and virtual equipment. The TCP bridge has been used to connect Unity games as well as hardware modules that could not easily support a full DDS stack.

**AMM CORE Module Design – Reference Modules**

The following reference modules are included as a guide for developers to develop their own AMM modules. Examples are given of how to subscribe and publish data, transmit capabilities to the module manager and set operational status.

1. **Virtual Equipment**
The Virtual Equipment simulator subscribes to physiology messages and prints them to standard output.
2. **Command Executor**
The Command executor takes arbitrary commands from standard input and publishes them to the DDS bus. This can be used for debugging and testing, or simply as an example for how to publish data that other modules can consume.
3. **“Kitchen Sink” Example Module**

## Software Inventory

All software source is available on the AMM GitHub repository <http://github.com/AdvancedModularManikin>.

The following pieces of software have been provided:

* AMM Standard Library – this is the basis for all AMM modules. It provides methods and APIs for communicating with the DDS bus using the AMM data model. <https://github.com/AdvancedModularManikin/amm-library>
* AMM CORE Software
	+ Module Manager
	+ Simulation Manager
	+ Physiology Manager
* AMM Extended CORE Software
	+ TCP Bridge
	+ REST Adapter
* AMM Example Modules
	+ Virtual equipment
	+ Command executor
	+ “Kitchen Sink” example module

## Software Environment

Since the AMM CORE software is provided as source, it can be compiled for a number of different environments and platforms. The following prerequisites must be installed (possibly built from source) in order to build all facets of the AMM CORE and Extended CORE:

* eProsima FastRTPS, FastCDR - <https://github.com/eProsima/Fast-RTPS>
* TinyXML2 - <http://www.grinninglizard.com/tinyxml2/>
* RapidJSON (REST adapter)
* Pistache (REST adapter)
* BioGears - <https://github.com/BioGearsEngine/core>
* C++ Boost libraries

Each of these may have their own dependencies.

As of this document, the AMM CORE modules have been built and utilized on the both Linux and Windows operating systems. The AMM CORE modules have been built and executed on the following hardware architectures:

* 64-bit ARM (AARCH64, ARMv8)
* 32-bit ARM (ARMHF, ARMv7)
* 64-bit Windows
* 32-bit Windows
* 64-bit Linux (x64)

## Software Organization and Overview of Operation

All AMM software, including the Standard Library and reference implementation modules, are published on GitHub for open source development: <https://github.com/advancedmodularmanikin>

The software is organized into several different repositories, which are outlined here:

* **AMM Specification**: the textual specifications for AMM as well as required XML schemas for expressing module capabilities and configuration information.
Repository: <https://github.com/AdvancedModularManikin/specification>
* **AMM Standard Library**: a reference implementation API to be used to create an AMM-compliant module. The standard library has a number of helper methods to interact with the AMM data bus via DDS/FastRTPS.
Repository: <https://github.com/AdvancedModularManikin/amm-library>
* **AMM CORE Software**: reference implementations of the Simulation Manager, Module Manager and Physiology Engine Manager.
* **AMM CORE Extended Software**: reference implementations of the TCP Bridge and REST Adapter which allow for non-DDS communication with the AMM data bus and connected modules.
* **AMM Example Modules**: modules showing various functionality and how to use the AMM Standard Library. A number of tutorials and how-to’s are included with the example modules to help understand how to develop a module from scratch or integrate existing hardware as a compliant module.

In general, each code repository includes a root-level CMake configuration file and a source directory (named src or Source) which contains specific source for that module. Reference implementation modules also contain a ‘config’ directory which includes files required to properly configure FastRTPS for the module as well as publish its operational description/capabilities and configuration.

## Assistance and Problem Reporting

Assistance and troubleshooting support is provided through GitHub Issues, a publicly available issue tracking system that is integrated with the AMM CORE software module repository.

# Access to Software

Access to all AMM CORE software modules is through a public, open source GitHub repository. It is available at <http://github.com/AdvancedModularManikin>.

The software’s source code can be browsed through this link or downloaded to compile and execute.

Procedures for building the software are provided in the repository. A developer building the software for the first time is expected to have some experience with basic software building processes, specifically using CMake.

A comprehensive developer’s guide is also available in the repository to help guide developers in choosing which software to use and how to integrate it to create AMM modules.

## First Time user of Software

### Equipment Familiarization

While the AMM CORE software can be run on many different platforms and architectures, most work has been done on Linux systems. Command-line tools are available for interacting with the AMM CORE modules, although ideally a web interface will be used to handle the actual work-flow of running a simulation.

### Installation and Setup

Installation and compilation of the AMM CORE software must be completed according to the published instructions before a simulation can be run. Before initiating a session, at the least the Simulation Manager, Module Manager and Physiology Engine Manager must be operational.

## Initiating a Session

Session control is all handled through the DDS data bus. Messages are published and modules are configured to receive those messages and perform actions as appropriate. In order to generate these DDS messages, a command line tool such as the AMM Command Executor can be used. In most situations, a user interface will be developed to control session flow through either the web or another application. A simple web-based dashboard has also been released to manage common tasks required for operating an AMM educational encounter or simulation.



**Figure 4: Example web interface.**

## Stopping and Suspending Work

Operation of the simulation can be temporarily paused by publishing the HALT message on the SimulationControl topic, transmitted as described above through either the Command Executor or through a user interface. Details on how the SimulationControl topic can be used to start, pause, resume, halt and save a simulation are given in the published specification docs.

# Processing Reference Guide

Guides to utilizing the reference implementation software are provided as part of the published open source software in the AMM GitHub repositories. Please refer to section 3.4 for descriptions of the repositories – each repository has its own corresponding processing procedures that are documented within the repository.