AD

AD-E404 336

Technical Report ARWSE-TR-20005

THE COMMON SYNC FRAMEWORK: A CROSS-CUTTING SOFTWARE SERVICE FOR THE COMMON OPERATING ENVIRONMENT

Mitchell Burger
Ross Arnold

December 2021

U.S. ARMY COMBAT CAPABILITIES DEVELOPMENT COMMAND ARMAMENTS CENTER

Weapons and Software Engineering Center

Picatinny Arsenal, New Jersey

The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

The citation in this report of the names of commercial firms or commercially available products or services does not constitute official endorsement by or approval of the U.S. Government.

Destroy by any means possible to prevent disclosure of contents or reconstruction of the document. Do not return to the originator.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-01-0188*

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (*From - To*)
December 2021	Final	

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
The Common Sync Framework: A Cross-Cutting Software Service for the Common Operating Environment	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHORS Mitchell Burger and Ross Arnold	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
U.S. Army DEVCOM AC, WSEC Fire Control Systems and Technology Directorate (FCDD-ACW-FM) Picatinny Arsenal, NJ 07806-5000	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
U.S. Army DEVCOM AC, ESIC Knowledge & Process Management Office (FCDD-ACE-K) Picatinny Arsenal, NJ 07806-5000	11. SPONSOR/MONITOR'S REPORT NUMBER(S) Technical Report ARWSE-TR-20005

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; Distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Common Operating Environment, Version 3 (COEv3) is a set of technologies and standards introduced by the U.S. Army. It is a part of a broad effort to standardize disparate computer systems across different environments, platforms, use cases, and programs. Many branching sub-efforts support the broader COEv3 effort. Mission Command Modernization (MCM) is one of these sub-efforts. The goal of MCM is to standardize computing systems across the Mission Command domain of operation. The MCM strategy calls for the development or acquisition of several critical software components that fit together to form a computing infrastructure. The Mission Command software systems will use this infrastructure for communications and interoperability. This report describes a central and key component within the MCM computing infrastructure called the Common Sync Framework (CSF). It will emphasize how the CSF fits into the infrastructure. Since it can be difficult to visualize how the various MCM infrastructure components fit together from an outside (non-development) perspective, a complete and correct description of the CSF will help facilitate broader understanding and success of the CSF development effort.

15. SUBJECT TERMS
Army software Common Operating Environment COE Common Sync Framework CSF Command Post Client CPC Command Post Computing Environment CPCE Command and Control Infrastructure UltraLight C2IUL Mission command Mission Command Modernization MCM Tactical applications Tacapps

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Ross Arnold
U	U	U	SAR	17	19b. TELEPHONE NUMBER (Include area code) (973) 724-8618

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

CONTENTS

FIGURES

INTRODUCTION

In October 2010, the U.S. Army's Chief Information Office/G-6 and the Assistant Secretary of the Army for Acquisition, Logistics and Technology [ASA (ALT)] approved the Common Operating Environment (COE) Architecture for the Army Enterprise Network. The COE is a set of computing technologies and standards that enables secure and interoperable applications to be developed rapidly and executed across a variety of computing environments, devices, and platforms. The COE architecture is intended to drastically reduce the time it takes to deliver relevant applications to the warfighter. The COE augments previous efforts to standardize end-user environments and software development kits. It establishes streamlined enterprise software processes that rely on common pre-certified reusable software components and it also assists in developing deployment strategies that allow users to directly access new capabilities. The benefits of a COE architecture include lower costs, improved interoperability, and easier system maintenance.

As of Fiscal Year 2017, the Common Operating Environment Version 3 (COEv3) was the latest implementation of the COE. Mission Command Modernization (MCM) is an effort within the COEv3 to standardize and modernize systems across the Mission Command domain. A key MCM strategy is to develop and/or acquire a common software infrastructure (see fig. 1). Mission Command software systems use this infrastructure for communications, collaboration, and interoperability. The infrastructure consists of several software components that work together to form a complete system.

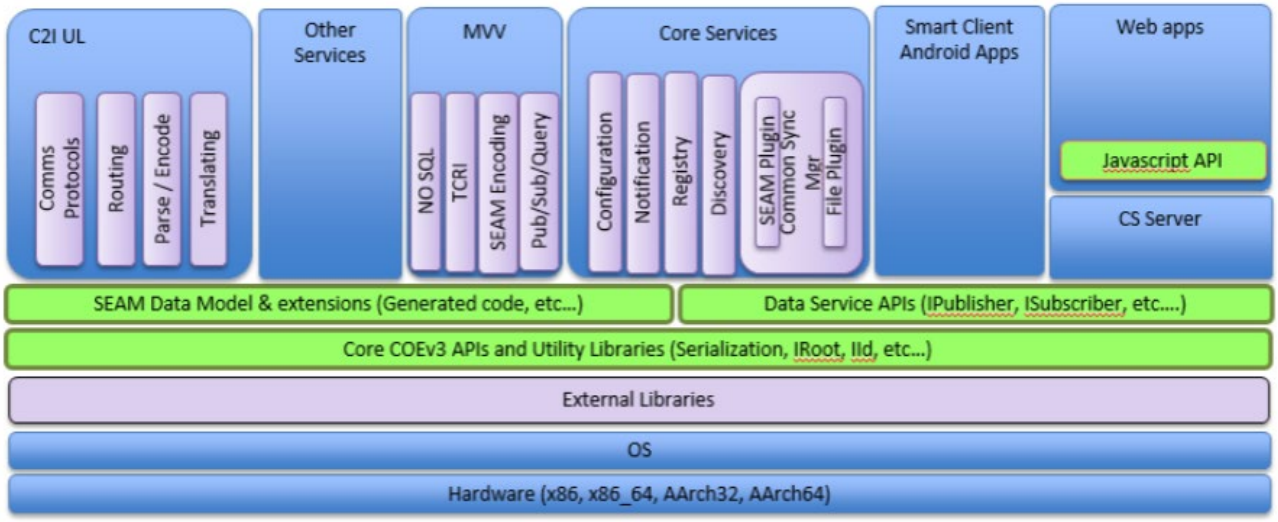

Figure 1
COEv3 data infrastructure

One of the components in this infrastructure is called the Common Sync Framework (CSF). The CSF is an application service within the MCM COEv3 data infrastructure. It provides a converged, secure, reliable, and bandwidth-sensitive dissemination of current and historical COEv3 data objects to other instances of the COEv3 data infrastructure across tactical networks. The CSF is intended to replace duplicative methods and components in the infrastructure baselines for communication of common data, as used previously in Command Post Computing environments (CPCE) and Mounted Computing environments (MCE). The CSF is designed to make appropriate decisions for disconnected, intermittent, and latent (DIL) networks and still be scalable for large-scale, high-bandwidth solutions. The architecture of the CSF allows third-party plugins to transport various data over point-to-point and multipoint connections.

The CSF provides the following operational benefits:

1. Acts as a single place to send native data across tactical networks
2. Offers encryption and compression of all connections
3. Handles setup for all connections

GO PROGRAMMING LANGUAGE

The CSF is largely written in the open source Go programming language (also called Golang) developed by Google. Go was selected to support the initially short (less than 6 months) CSF development timeline. It has a comprehensive standard library that is cross-platform (runs on Android, Linux, and Windows). It also includes an extensive set of code tools that are used for profiling, code coverage, race detection, and unit testing. Go natively provides many of the software capabilities needed by the CSF. By using Go, it removes the need to develop software packages in-house or acquire additional third-party components. This results in reduced development time and cost.

COMMON SYNC FRAMEWORK ARCHITECTURE

The CSF architecture is shown in figure 2. The components that reside above the black line are not part of the CSF. These components (which are applications and services) provide data for the CSF to replicate over a tactical network. Components that reside below the black line are part of the CSF.

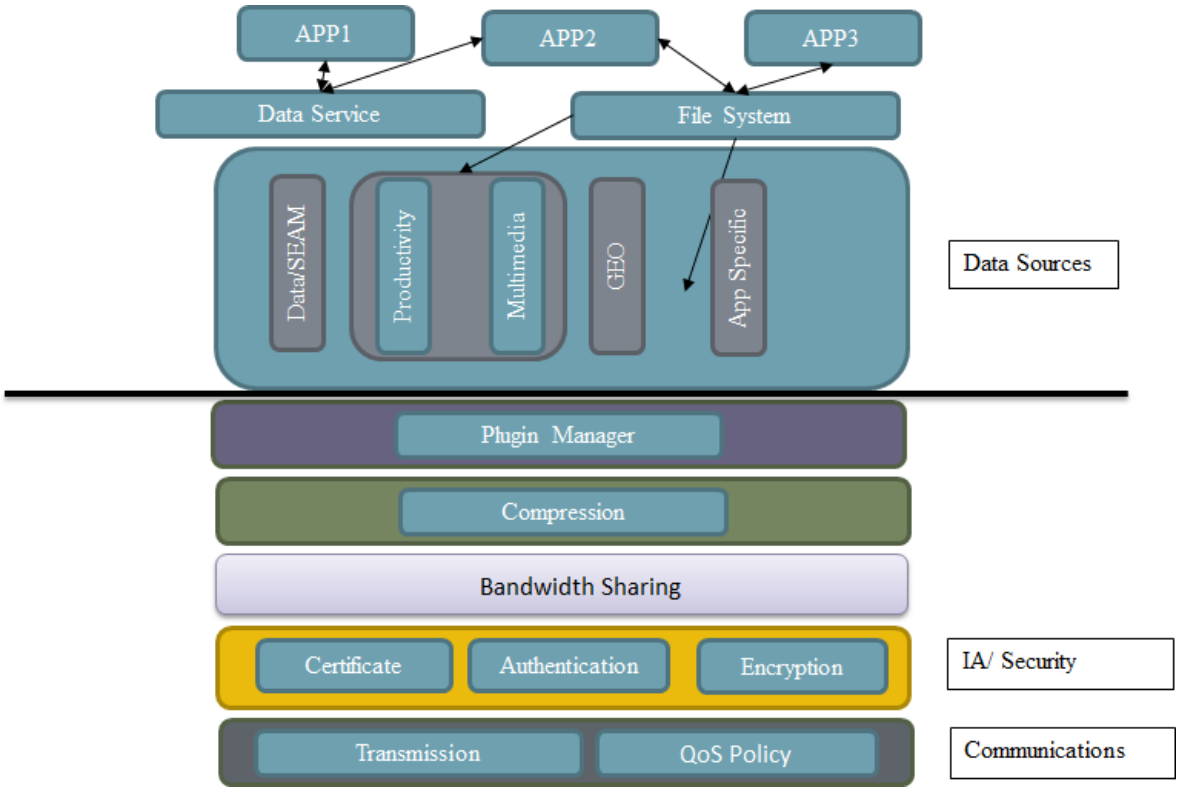

Figure 2
Common Sync Framework architecture

The Inter-Process Communication (IPC) mechanism used by the CSF is implemented in both the data plugin libraries and the framework service code. The plugin libraries hide the implementation details from the data plugins.

The compression layer provides lossless compression on the plugin data to minimize bandwidth needs. For simplicity and performance, compression plugins run inside the CSF process. Go provides implementations of gzip, zlib, and lzw in its standard library.

The bandwidth management layer in the CSF is essentially responsible for sharing limited outbound remote connection bandwidth among multiple data plugins. For MCM Engineering Release (ER) 1.1, the CSF configuration file defines the minimum percentage of the available connection bandwidth that can be used by each data plugin. If additional bandwidth is available, it is allocated evenly among all the data plugins that are ready to send data.

The Information Assurance (IA)/Security layer handles application layer communication hardening between two or more framework services. For simplicity and performance, IA/Security plugins run inside the CSF process. Security is nuanced and requires several combinations of processes. For ER 1.1, the framework uses the Transport Layer Security (TLS) protocol, which is a standard approach that incorporates the necessary components in a tried and true manner.

The communications layer handles network transmission and reception of data between two or more framework services. For simplicity and performance, the communications plugins run inside the CSF process.

COMMON SYNC FRAMEWORK WITHIN THE DATA INFRASTRUCTURE ARCHITECTURE

The architecture of the COEv3 MCM data infrastructure is shown in figure 3. Third-party developers developed the data plugins that provide data for the CSF. That data is pulled and sent to a remote instance of a CSF over a tactical network. The CSF does not receive data directly from any application, service, or the bus on the data infrastructure.

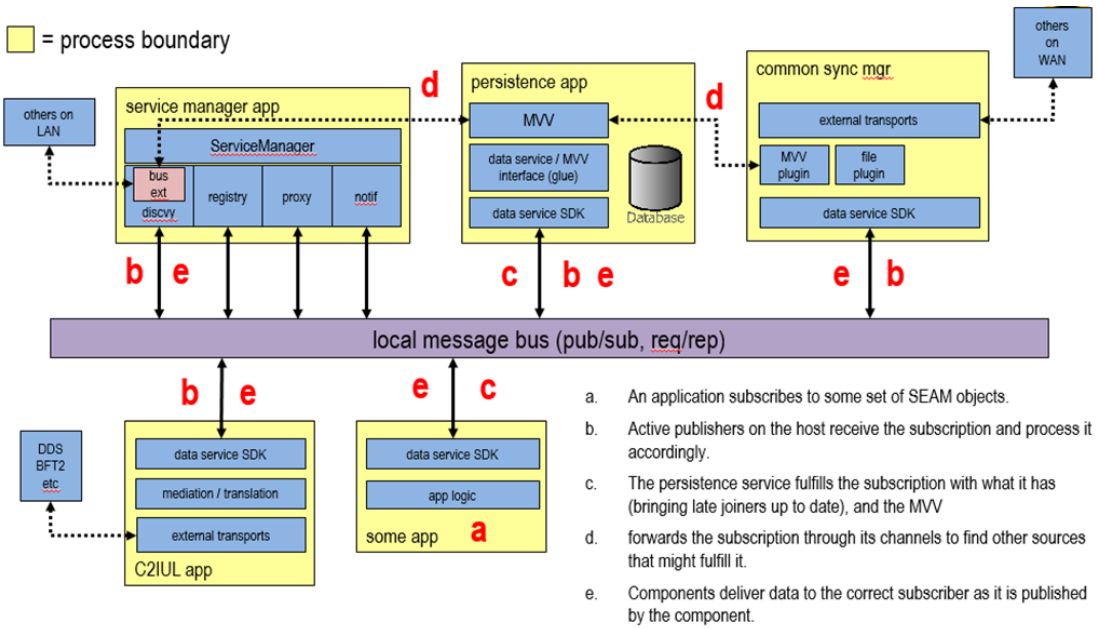

Figure 3
COEv3 data infrastructure architecture

COMMON SYNC FRAMEWORK CONFIGURATION

The CSF creates communication pathways. Data transformation is based on the instructions found in a JavaScript Object Notation (JSON)-formatted configuration file that is provided to the CSF at service start-up. This file is intended to be readable and editable by any person.

The JSON configuration file includes the following five top-level items.

1. **A universally unique identifier (UUID)** - The service identifier is an optional UUID string that uniquely identifies the service instance. However, the CSF will automatically create the service identifier if one is not created.
2. **Network interfaces** - Each network interface is identified by a name such as “en0”. A network interface may still be used even if its configuration information is not included in the configuration file explicitly.
3. **Supported security protocols** - When a new connection is received, the CSF allows the connection to proceed as long as it uses one of the supported security protocols enumerated in its configuration. Security protocol settings are configured in this section of the configuration file. Since security protocols have their own configuration requirements, there isn’t a universal format specified for the individual security protocol settings. A new convention will be specified for each supported security protocol. Each supported security protocol is identified by a unique string that must match a security protocol identifier on the remote CSF. The security protocol identifier is a value that contains a JSON object with universal and custom parameters.
4. **Remote connections** - Each remote connection is uniquely identified by its service identifier.
5. **Data plugins** - Each data plugin specifies a collection of capabilities and is identified by a name such as “Sync”. These capabilities are string identifiers that indicate a particular class of data supported by that plugin. Data plugins must specify at least one capability.

COMMON SYNC FRAMEWORK DOCUMENTATION

For each ER, a set of contract deliverable (CDRL) documents are released with the source code and a Linux “rpm" package manager installation file. Each ER contains a “snapshot” of the CDRLs, which are updated throughout the development process by software and systems engineers. The CDRLs include an interface control document (ICD), a software development kit (SDK), a software test description (STD) document, a system/subsystem (component) design document (SSDD), a system (component) installation plan (SIP), a system (component) version description (SVD) document, and a component requirements specification (CRS) document. These items are included in the Infrastructure CSF deliverables. Other documentation that may not be delivered in an ER such as a concept of operations (CONOPS) document, functional specification document, component design document, software development and test document, integration and verification document, and third-party integration document. These items can be found in the Infrastructure Common Sync Framework Confluence page on the Defense Intelligence Information Enterprise (DI2E) environment.

The SDK is designed for anyone who needs to develop a third-party plugin that uses the CSF to send data over a tactical network, or anyone who wants to learn more about the architecture of the CSF and how it was implemented. The SDK describes the components of the CSF architecture and it provides a high-level overview of how the data plugins work with the CSF. The SDK also describes which development languages are used and how they factor into the development of the CSF. Another section in the SDK describes in detail how a third-party developer can create a data plugin for use with application programming interfaces (API) and the interfaces that need to be developed for a third-party data plugin. Additionally, sample code (that includes a link to a detailed example) is provided to assist a developer in developing a data plugin.

COMMON SYNC FRAMEWORK DATA FLOWS

The data flow of the mission command data within the MCM architecture is shown in figure 4. Command and Control Infrastructure UltraLight (C2IUL) is used where mediation to legacy formats such as Variable Message Format (VMF) via Blue Force Tracking 2 (BFT2) is necessary. Sending data using a tactical network such as Warfighter Information Network – Tactical (WIN-T) is handled through the CSF.

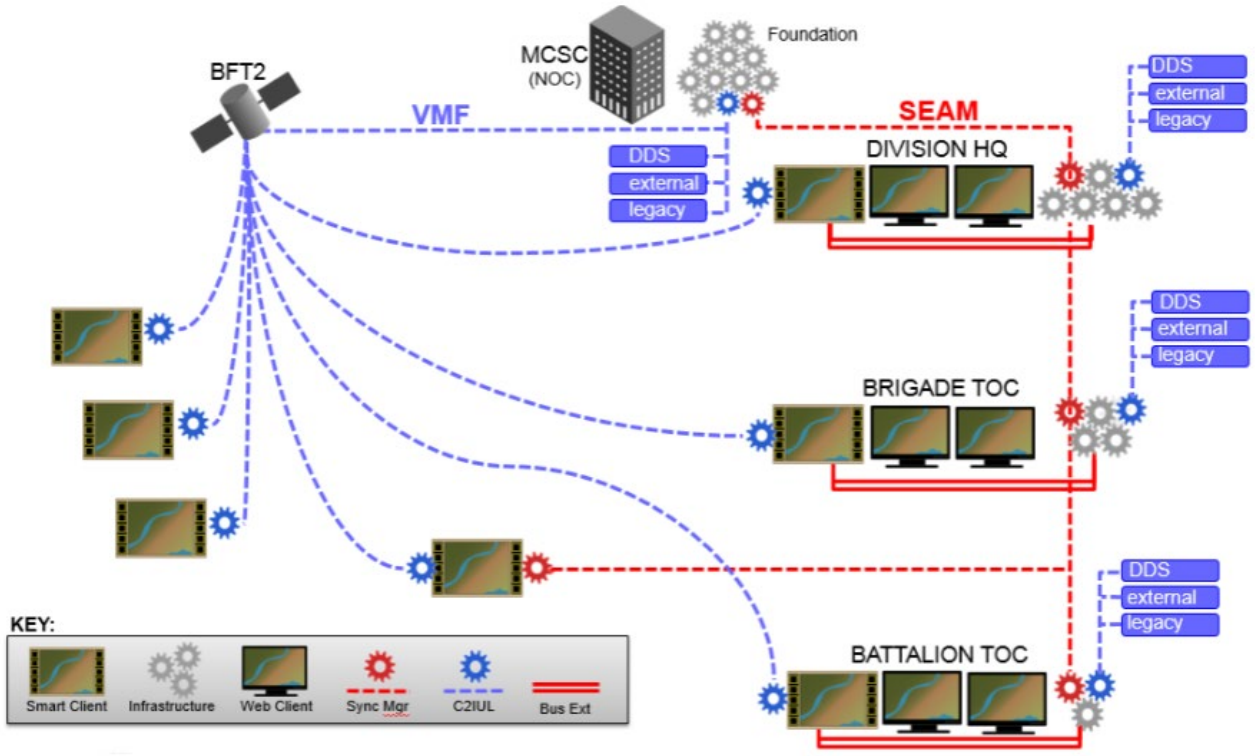

Figure 4
COEv3 mission command data flow

The CSF data flow is shown in figure 5. The CSF sends/receives data to/from a remote instance of the CSF or a local data plugin. After the data plugin is registered with the CSF, the CSF pulls the data provided by the data plugin. The CSF (optionally) compresses the data and splits it into smaller packets. Then, the packetized data is (optionally) encrypted and transmitted over a tactical network to a remote instance of a CSF. If the data was encrypted, the remote CSF decrypts the data and re-assembles it. If the data was compressed, it is decompressed and transmitted to the corresponding remote registered plugin.

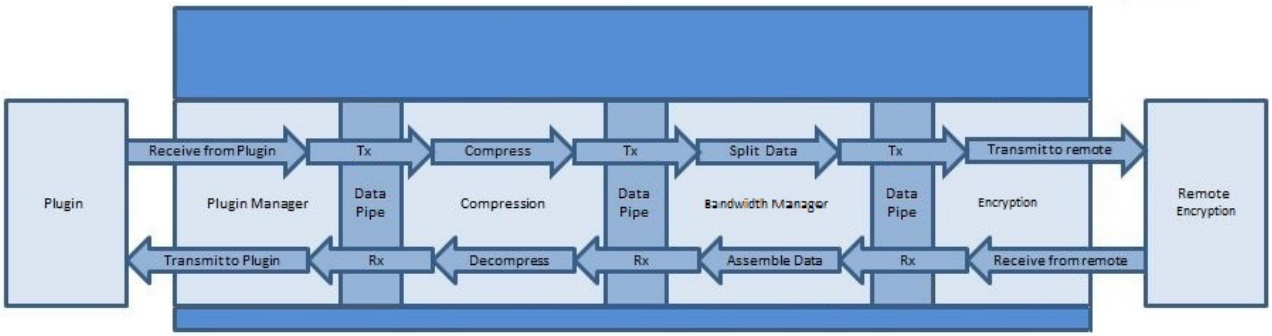

Figure 5
Common Sync Framework data flow

COMMON SYNC FRAMEWORK PERFORMANCE METRICS

The bandwidth performance of a simulated instance of the CSF is shown in figure 6. For ER1, a goal was set up to ascertain the maximum throughput of the CSF service itself, independent of any external network constraints. This was established as a baseline for future performance enhancements. Figure 6 shows the *n* CSF instances connected to a single CSF instance. Each instance ran through a single data plugin on a laptop and it sent a total of 2*n 5kB messages simultaneously between data plugins. Encryption and compression are both enabled for all connections.

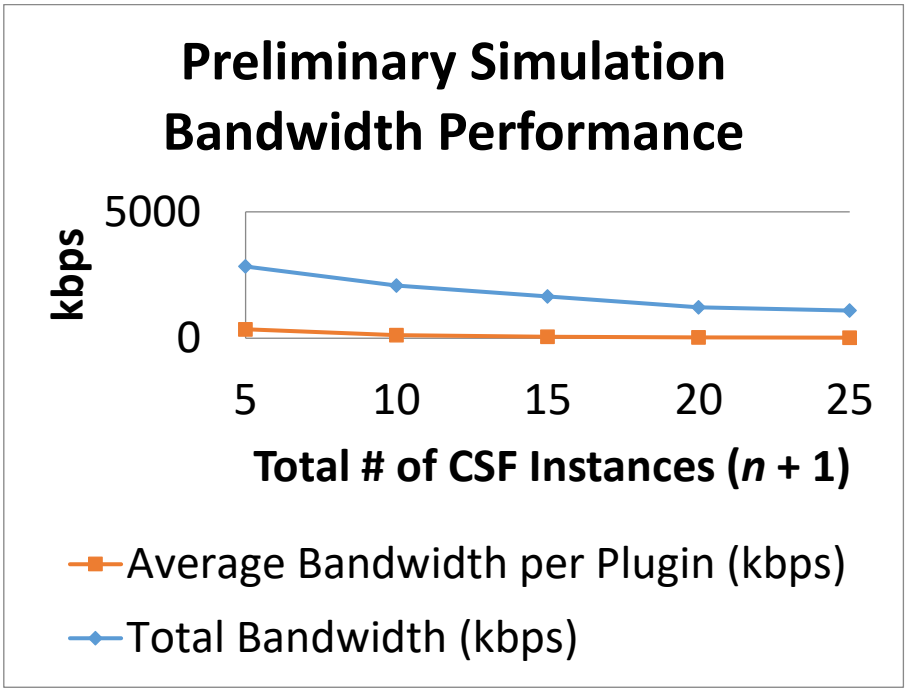

Figure 6
Common Sync Framework simulation bandwidth performance

For ER1, a goal was set up to ascertain the latency added by the CSF service itself, independent of any external network latency. This was established as a baseline for future performance enhancements. Figure 7 shows the *n* CSF instances connected to a single CSF instance. Each instance ran through a single data plugin on a laptop and it sent a total of 2*n 5kB messages simultaneously between data plugins. Encryption and compression are both enabled for all connections.

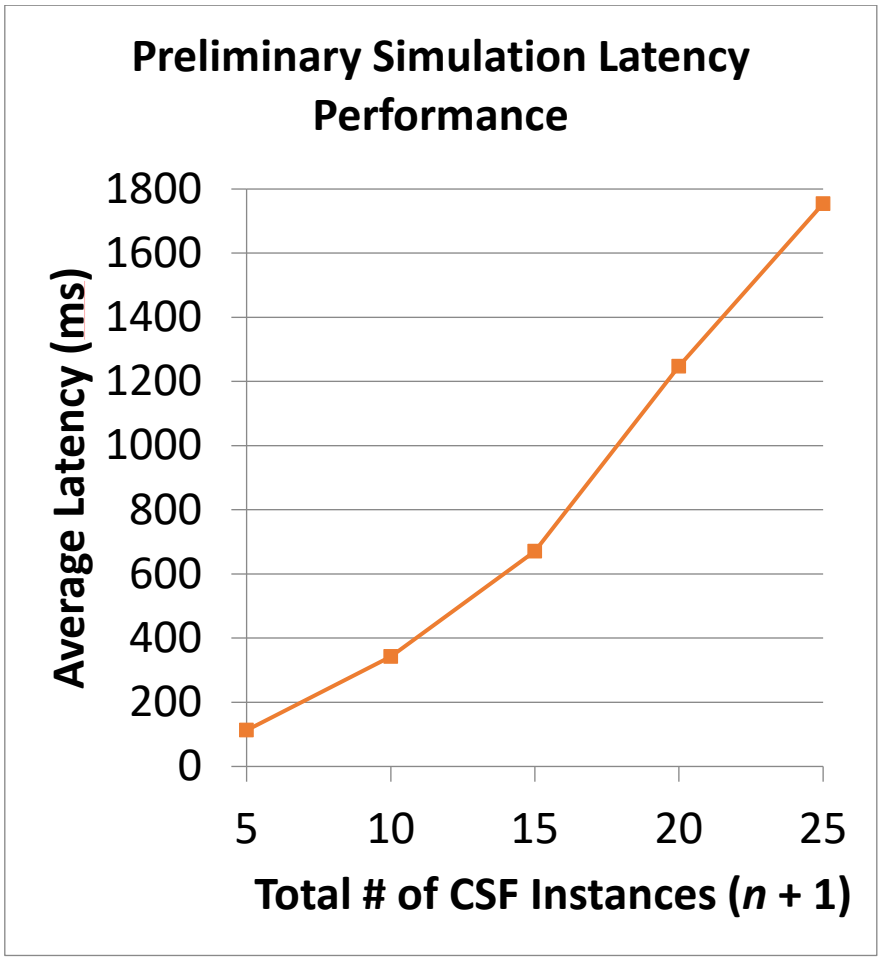

Figure 7
Common Sync Framework simulation latency performance

COMMON SYNC FRAMEWORK SCOPE AND BUILD PLAN

As part of the COEv3 data infrastructure update, four main ERs are planned for the CSF. Each ER in turn has between three and four incremental drops. A complete list of which capabilities of the CSF are to be included in each ER is shown in table 1. For each release of the CSF, the source code, an rpm file, and the associated documentation are provided to the integration team. Then, the released CSF is installed in the Mission Command laboratory at the Aberdeen Proving Ground (APG), MD. Additionally, nightly builds of the CSF are conducted and the output is uploaded to the COEv3 project's Nexus webpage on DI2E.

Table 1
Common Sync Framework scope and build plan

	POR: Scope and Build POC:									ER1 DevAutoIA Complete 27 JUN 16			ER1.1 DevAutoIA Complete 13 OCT 16			ER2 DevAutoIA Complete 4 APR 17				ER3 DevAutoIA Complete 26 SEP 17			
Category	Sub-categor	Applicable Initial TG	Component	Sub-Component	llocated PC	Implementatio TSRV eb	Implementatio Smart Client	Infras SUB-Component Interdependencies	Sub-Component Interdepende	Iteration 1 2FEB-23MAR	Iteration 2 16MAR-3MAY	Iteration 3 27APR-14JUN	Iteration 4 8JUN-26JUL	Iteration 5 20JUL-8SEP	Iteration 6 31AUG-19SEP	Iteration 7 12OCT-30NOV	Iteration 8 23NOV-12JAN	Iteration 9 4JAN-22FEB	Iteration 10 15FEB-	Iteration 11 29MAR-16MAY	Iteration 12 10MAY-	Iteration 13 21JUN-	Iteration 14 9AUG-
Data Architecture	Remote Data Synch	4.3.5	Framework (Sync)	Sync Transport Connection Interface	PdM TMC	X	X			Proto	Partial	Full	Full	Full	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated
				Configuration file management	PdM TMC	X	X			Proto	Partial	Partial	Partial	Full	Full	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated
				Encryption/decryption	PdM TMC	X	X			Proto	Proto	Partial	Partial	Partial	Partial	Full	Full	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated
				Plugin manager	PdM TMC	X	X			Proto	Proto	Partial	Partial	Partial	Full	Full	Full	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated
				Multiple Pipelines/Connections	PdM TMC	X	X			ND	ND	ND	Proto	Partial	Full	Full	Full	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated
				Network connections	PdM TMC	X	X			Proto	Proto	Partial	Partial	Partial	Partial	Full	Full	Full	Integrated	Integrated	Integrated	Integrated	Integrated
Data Architecture	Remote Data Synch	4.3.5	Plugins (Sync)	SEAM transport connection	PdM TMC	X	X	Sync Transport Connection Interface		Proto	Proto	Proto	Partial	Partial	Partial	Full	Full	Full	Full	Integrated	Integrated	Integrated	Integrated
				File transport connection	PdM TMC	X	X	Sync Transport Connection Interface		ND	Proto	Proto	Proto	Partial	Partial	Partial	Full	Full	Full	Full	Integrated	Integrated	Integrated
Data	Remote	4.3.5	NetMon		PdM TMC	X	X			ND	ND	ND	Proto	Proto	Partial	Partial	Partial	Partial	Full	Full	Full	Full	Full
Data Architecture	Remote Data Synch		Protocols (Sync)	Sync Encryption	PdM TMC	X	X			ND	Proto	Proto	Proto										
				Data Compression	PdM TMC	X	X			ND	Proto	Proto	Proto										
				Daemonized Containerable Service	PdM TMC	X	X			ND	Proto	Proto	Proto										
Data Architecture	Remote Data Synch		GoS (Sync)	Prioritizer	PdM TMC	X	X			Proto	Proto	Proto	Partial	Partial	Partial	Partial	Full	Full	Full	Integrated	Integrated	Integrated	Integrated
				Configuration	PdM TMC	X	X			Proto	Proto	Proto	Partial	Partial	Partial	Partial	Full	Full	Full	Integrated	Integrated	Integrated	Integrated
Data Architecture	Remote Data Synch	4.3.5	Config (Sync)	Web client GUI	PdM TMC	X		Configurationfile management, UWStiye Framework, Web Framework - OZP		ND	ND	Proto	Proto	Partial	Partial	Partial	Full	Full	Full	Integrated	Integrated	Integrated	Integrated
				Smart client GUI	PdM TMC		X	Configurationfile management, UWStiye Framework, Smart Client Framework - Android		ND	ND	Proto	Proto	Partial	Partial	Partial	Full	Full	Full	Integrated	Integrated	Integrated	Integrated

CONCLUSIONS

The Common Sync Framework (CSF) provides a "common" method to set policies and filters for all data transport requirements. This removes the "stove pipe" interoperability mechanisms that provide synchronized services currently used by applications in the Command Post Computing environments and Mounted Computing environments. The CSF is designed specifically for disconnected, intermittent, and latent networks. It provides bandwidth-sensitive distribution of data and scaling capabilities for the top notch performance expected at the command post. The CSF operates in a store-and-forward manner that delivers data upon reconnection as specified by configurable policies and filters.

BIBLIOGRAPHY

1. Army Chief Information Officer (CIO), “Common Operating Environment Guidance, Enabling Success for Today and Tomorrow,” U.S. Army CIO/G-6, October 2010.

2. Arnold, R. D. and Wade, J. P., “A Definition of Systems Thinking: A Systems Approach,” Procedia Computer Science, Volume 44, Pages 669-678, 2015. http://doi.org/10.1016/j.procs.2015.03.050

3. Arnold, R. D., Lieb, A. J., Samuel, J. M., and Burger, M. A., “The next generation of command post computing,” Proceedings of the 2015 SPIE Defense Systems + Security Conference, Volume number and publication order: 9456-30, Baltimore, MD, April 2015.

4. Lieb, A. J., Arnold, R. D., Cauvel, J., and Grim, L., “Tactical Applications (TacApps) User Design Workshop Analysis and Findings Report,” Contractor Report ARWSE-CR-16004, U.S. Army ARDEC, Picatinny Arsenal, NJ, November 2017.

5. Arnold, R. D., Dykstra, M., Desilva, K., Faulkner, M., and Ommert, W., “Command Post of the Future BC13.2 Scalability Report,” Contractor Report ARWSE-CR-16005, U.S. Army CCDC AC, Picatinny Arsenal, NJ, September 2019.

6. Arnold, R. D., “Command Post Client: A Prototype Software Application,” Technical Report ARWSE-TR-16040, U.S. Army ARDEC, Picatinny Arsenal, NJ, September 2017.

7. Reid, T. A., Kim, A. B., and Arnold, R. D., “Using Tactical Applications (TacApps) Inside Ozone Widget Framework: A Feasibility Analysis,” Contractor Report ARWSE-CR-16001, U.S. Army CCDC AC, Picatinny Arsenal, NJ, January 2020.

8. Antunes, N. and Arnold, R. D., “An Investigation of the Data-driven Documents JavaScript Library for use in Tactical Applications,” Technical Report ARWSE-TR-16006, U.S. Army ARDEC, Picatinny Arsenal, NJ, November 2016.

9. Sunga, J. and Arnold, R. D., “Tactical Applications (TacApps) Android Development Approach Analysis,” Technical Report ARWSE-TR-16010, U.S. Army ARDEC, Picatinny Arsenal, NJ, January 2017.

10. Reid, T. A., Kim, A. B., and Arnold, R. D., “The Ozone Platform – An Initial Analysis for Tactical Applications (TacApps),” Contractor Report ARWSE-CR-16002, U.S. Army ARDEC, Picatinny Arsenal, NJ, December 2016.

11. Reid, T. A., Klementowski, C., and Arnold, R. D., “Tactical Applications JavaScript Development Tools Recommendations,” Technical Report ARWSE-TR-15041, U.S. Army CCDC AC, Picatinny Arsenal, NJ, January 2020.

12. Burger, M., Kim, A., and Arnold, R. D., “Third-Party Software Dependency Analysis for Tactical Applications (TacApps) Increment 1,” Technical Report ARWSE-TR-15042, U.S. Army ARDEC, Picatinny Arsenal, NJ, November 2017.

13. Reid, T. A., and Arnold, R. D., “Reuse of the Cloud Analytics and Collaboration Environment within Tactical Applications (TacApps): A Feasibility Analysis,” Contractor Report ARWSE-CR-15001, U.S. Army ARDEC, Picatinny Arsenal, NJ, March 2016.

BIBLIOGRAPHY
(continued)

14. Arnold, R. D., “Lightweight Tactical Client: A Capability-Based Approach to Command Post Computing,” Technical Report ARWSE-TR-15015, U.S. Army ARDEC, Picatinny Arsenal, NJ, December 2015.

15. Burger, M., and Arnold, R. D., “Erecting an Ozone Widget Framework Instance for Tactical Applications,” Technical Report ARWSE-TR-15012, U.S. Army ARDEC, Picatinny Arsenal, NJ, November 2015.

16. Burger, M., Klementowski, C., and Arnold, R. D., “A Performance Analysis of the Communications-Electronics Command (CECOM) Software Engineering Center (SEC) Renderer for MIL-STD-2525 Tactical Graphics,” Technical Report ARWSE-TR-15013, U.S. Army ARDEC, Picatinny Arsenal, NJ, October 2015.

DISTRIBUTION LIST

U.S. Army DEVCOM AC
ATTN: FCDD-ACE-K
FCDD-ACW-FM, R. Arnold
FCDD-ACW-NL, M. Burger
Picatinny Arsenal, NJ 07806-5000

Defense Technical Information Center (DTIC)
ATTN: Accessions Division
8725 John J. Kingman Road, Ste 0944
Fort Belvoir, VA 22060-6218

GIDEP Operations Center
P.O. Box 8000
Corona, CA 91718-8000
gidep@gidep.org

REVIEW AND APPROVAL OF ARDEC REPORTS

THIS IS A:

☒ **TECHNICAL REPORT** ☐ **SPECIAL REPORT**

☐ **MEMORANDUM REPORT** ☐ **ARMAMENT GRADUATE SCHOOL REPORT**

FUNDING SOURCE PM funded EMD
[e.g., TEX3; 6.1 (ILIR, FTAS); 6.2; 6.3; PM funded EMD; PM funded Production/ESIP; Other (please identify)]

The Common Sync Framework: A Cross-Cutting Software Service for the Common Operating Environment
Title

Tactical Applications
Project

Mitchell Burger, Ross Arnold
Author/Project Engineer

Report number/Date received (to be completed by LCSD)

8618 31
Extension Building

PCDD-ACW-FM
Author's Office Symbol

PART 1. **Must be signed before the report can be edited.**

a. The draft copy of this report has been reviewed for technical accuracy and is approved for editing.

b. Use Distribution Statement A ✓, B____, C____, D____, E____, or F____ for the reason checked on the continuation of this form. Reason: public release

1. If Statement A is selected, the report will be released to the National Technical Information Service (NTIS) for sale to the general public. Only unclassified reports whose distribution is not limited or controlled in any way are released to NTIS.

2. If Statement B, C, D, E, or F is selected, the report will be released to the Defense Technical Information Center (DTIC) which will limit distribution according to the conditions indicated in the statement.

c. The distribution list for this report has been reviewed for accuracy and completeness.

Tri Lu 9/16/2019
Division Chief (Date)

PART 2. To be signed either when draft report is submitted or after review of reproduction copy.

This report is approved for publication.

Tri Lu 9/16/2019
Division Chief (Date)

RDAR-CIS (Date)

LCSD 49 (1 Sept 16)
supersedes SMCAR Form 49, 20 Dec 06