

SDK & C++ API Training

Test & Training Enabling Architecture (TENA)
Software Development (SDK) and Application Programming Interface (API) training
using Object Models (OM) from the VOICES User Group (VUG)



Schedule overview

- Monday, 12 December
 - TENA SDK overview, installation and build environment verification
 - API code overview using auto-generated example applications
 - Creating a barebones TENA app that joins an execution
 - Creating an SDO
- Tuesday, 13 December
 - Updating an SDO
 - Subscribing to SDOs
 - Implementing SDO remote methods (from a publisher app)
 - Invoking (calling) SDO remote methods (from a subscribing app)
- Wednesday, 14 December
 - Sending TENA messages
 - Subscribing to TENA message types
 - A closer look at TENA TSPI
 - Simulation time with TENA
 - OM design best practices
 - Special topics – CMake & VS Code setup
- Backup topics
 - TDCS & DataView
 - Middleware configuration
 - Advanced filtering
 - Middleware IDs
 - Middleware metadata
 - Handling exceptions
 - Sending alerts
 - Log files
 - TRMC website services



Prerequisites

- Authorization to access TENA website(s)
- Basic knowledge of TENA & related concepts
 - Distributed engineering & TENA specific terminology
 - Executions and starting an Execution Manager (EM)
 - Middleware
 - Object Models (SDOs, Messages, Local Classes, etc.)
 - Covered previously, in TENA intro/Q&A
 - Standard OMs
- TENA SDK – acquisition, installation, contents
 - Covered previously, in TENA SDK overview
- Background with C++ programming language



Notes on VUG OM examples

- Exercises in this custom training use some of the SDOs and messages from the VUG OM
- Exercises are intended to exercise various TENA API concepts
- Exercises do not attempt to illustrate operational realism using the VUG OM, but should serve as a starting point for better understanding how to publish and subscribe to TENA SDOs and messages in the VUG OM (or any other OM)

Firewalls

- Before running network apps, it's a good idea to understand how firewalls might impact connections
- TENA apps will listen on local endpoints
 - Usually just one
 - e.g., “-listenEndpoints 10.1.10.102:55100/portspan=10” (bind to the first available TCP port – starting at 55100 and ending at 55109)
- TENA apps will connect to remote EM endpoints
 - Usually just one
 - e.g., “-emEndpoints 10.1.10.101:55100” (connect to the EM at 10.1.10.101 on TCP port 55100)
- For executions configured to allow “best effort” communication, UDP multicast addresses and ports will be used as well
- Any firewalls (including host-based firewalls) need to allow this TCP “Reliable” (and UDP/Multicast “Best Effort” if configured) communication to occur



Some VS-Code pointers

- Microsoft Visual Studio (VS) Code is not required to build TENA apps
 - However, we'll provide a few pointers for folks that use VS-Code
- VS-Code with C++ Tools for IntelliSense, etc.
 - Tell VS-Code where to find TENA header files
 - Set TENA_PLATFORM preprocessor directive



C/C++ for VS Code, IntelliSense

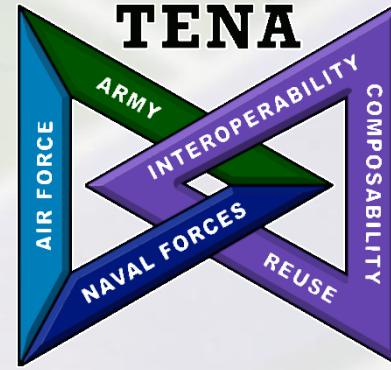


- In settings.json...
 - Or project specific settings file (.vscode/c_cpp_properties.json)
 - Tell IntelliSense where to look for header files

```
"C_Cpp.default.includePath":  
[  
    "${TENA_HOME}/${TENA_VERSION}/include/**",  
    "${workspaceFolder}/**"  
]
```

- Tell IntelliSense what “TENA_PLATFORM” to use

```
"C_Cpp.defaultdefines":  
[  
    "TENA_PLATFORM=${TENA_PLATFORM}"  
]
```



Lesson 1

TENA SDK overview, installation and C++ build environment verification



Topics

- Acquiring the SDK
- Installing the SDK
- SDK contents
- Checking (and changing) environment variables
- Starting an EM
 - From the TENA Console
- Building/running auto-generated C++ example apps
 - *(to verify the build environment is working)*



SDK contents – quick peek

- Execution Manager (EM)
- TENA Console
- TENA Canary
- Environment scripts
- Middleware C++ header files & library
- Standard OMs & Example OM
 - TDL files
 - C++ header files & libraries
 - (*C# bindings, Java bindings, etc. acquired via repository*)
 - Example application source code
 - SDO & Message – Publishers & Subscribers



Acquiring the SDK

- Log into the TRMC TENA website
 - <https://www.trmc.osd.mil/> for US Government & US Government Contractors
 - <https://www.tena-sda.org/> for all others
- Follow “Middleware Downloads” link
 - <https://www.trmc.osd.mil/wiki/display/MW/Home>
 - <https://www.tena-sda.org/wiki/display/MW/Home>
- Use the platform selection form to choose OS/compiler
- Download the selected SDK



Navigate to TENA space on TRMC website

This website is for UNCLASSIFIED USE ONLY. Do not discuss, enter, transfer, process, or transmit any CLASSIFIED information.

Spaces Calendars ... ?

TRMC JMetc CHEETAS TENA NCRC

Pages ...

Test Resource Management Center

Joint Mission Environment Test Capability (JMetc)

The JMetc mission is to provide a persistent capability for linking distributed facilities, enabling DoD customers to develop and test warfighting capabilities in a Joint Context. JMetc provides a test infrastructure consisting of the components necessary to conduct Joint distributed test events by cost-effectively integrating live, virtual, and constructive (LVC) test resources that are configured to support the users' needs. The JMetc program provides its customers a support team to assist with JMetc products and the conduct of

Test and Training Enabling Architecture (TENA)

The United States Office of the Secretary of Defense Test Resource Management Center (TRMC) has developed a common architecture to support effective integration and reuse of testing, training, and simulation capabilities that require real-time collaboration between distributed computer systems operating within diverse testing and training environments. Through the establishment of the Test and Training Enabling Architecture, the interoperability and reuse of range assets are tremendously improved, thereby the reducing

National Cyber Range Complex (NCRC)

The National Cyber Range Complex is a holistic, integrated cyber range capability comprising the National Cyber Range (NCR), Regional Service Delivery Points (RSDP), and the Joint Mission Environment Test Capability (JMetc) Multiple Independent Levels of Security (MILS) Network (JMN). The NCRC provides Department of Defense (DoD) and other government and industry users with representative environments, a distributed network infrastructure, and other tools and services to support cybersecurity testing and training requirements.



Click the “TENA Middleware” link



TENA-team members can add news items on the [News](#) wiki page.



TENA Introductory Material

Papers, presentations, and fact sheets concerning the TENA project & products.



TENA Project Information

Access TENA project and project related information.



TENA Training

Obtain information concerning TENA training.



TENA Helpdesk

Access the TENA Helpdesk to submit issues or search for solutions.



Contact Information

Access contact information concerning the TENA project and website services.



TENA Repository

Access the TENA repository for browsing and building object models.



TENA Middleware

Obtain information concerning the TENA Middleware.



Object Models

Obtain information about TENA Object Models.



TENA User Profile

Review and update your user profile.



User Groups

Learn about TENA user groups.



Click the “Middleware Downloads” link



TENA Middleware

ⓘ Middleware 6.0.9 is Available

- Refer to [Middleware 6.0.9 Release Notes](#)

The TENA Middleware provides a common software infrastructure for supporting realtime distributed events. Documentation associated with the TENA Middleware is provided below, along with a link for downloading the software.

Questions, problems, or suggestions associated with the TENA Middleware and related products should be submitted to the [TENA Helpdesk](#).



Overview Documentation

Introductory information concerning the TENA Middleware.



Release Notes

Notes on release enhancements, changes, and known issues.



Installation Guide

Information associated with installing the TENA Middleware.



Bugs/Feature Requests

Report a problem or suggest improvements with the middleware.



Programmer's Guide

Detailed information concerning the proper use of the TENA Middleware.



API Guide

Middleware Application Programmer Interface documentation.



Performance

Performance information for typical operational configurations.



Middleware Downloads

Repository Middleware SDK Distributions.





Click the latest Middleware SDK version link

TENA Website Repository Home Components Help

Browse | Search

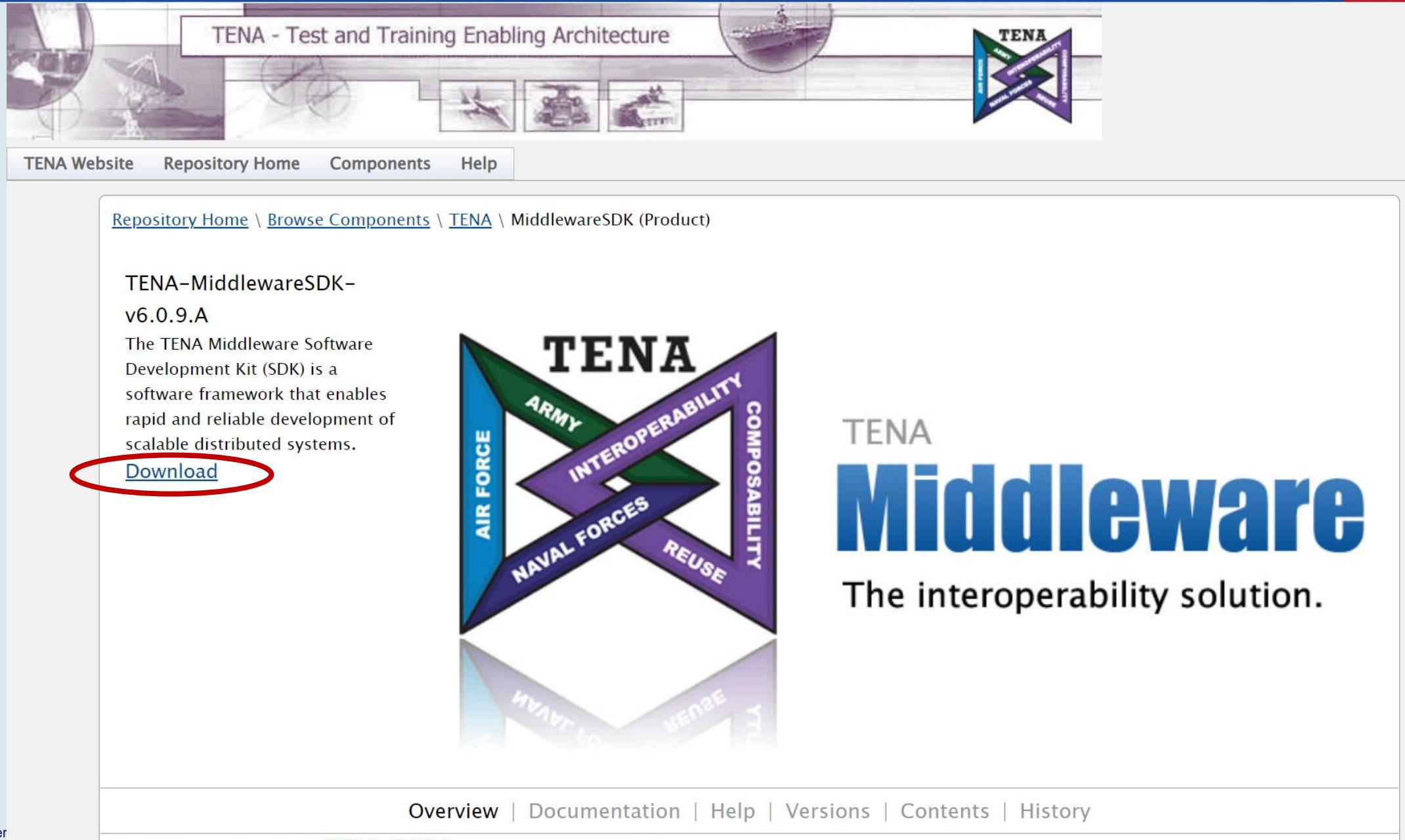
name:MiddlewareSDK OR [?](#) [X](#)

Show archived

Found: 6 Showing: 6

Name	Description	Type	Attributes
TENA-MiddlewareSDK-v6.0.5.1.A	The TENA Middleware Software Development Kit (SDK) is a software framework that enables rapid and reliable development of scalable distributed systems.	Product	Released Public
TENA-MiddlewareSDK-v6.0.5.A	The TENA Middleware Software Development Kit (SDK) is a software framework that enables rapid and reliable development of scalable distributed systems.	Product	Released Public
TENA-MiddlewareSDK-v6.0.6.A	The TENA Middleware Software Development Kit (SDK) is a software framework that enables rapid and reliable development of scalable distributed systems.	Product	Released Public
TENA-MiddlewareSDK-v6.0.7.B	The TENA Middleware Software Development Kit (SDK) is a software framework that enables rapid and reliable development of scalable distributed systems.	Product	Released Public
TENA-MiddlewareSDK-v6.0.8.B	The TENA Middleware Software Development Kit (SDK) is a software framework that enables rapid and reliable development of scalable distributed systems.	Product	Released Public
TENA-MiddlewareSDK-v6.0.9.A	The TENA Middleware Software Development Kit (SDK) is a software framework that enables rapid and reliable development of scalable distributed systems.	Product	Released Public

Click the “Download” link



The screenshot shows the TENA Test and Training Enabling Architecture website. At the top, there's a banner with a satellite dish, a compass, and three small images of military aircraft. Below the banner, the navigation menu includes "TENA Website", "Repository Home", "Components", and "Help". The current page path is "Repository Home \ Browse Components \ TENA \ MiddlewareSDK (Product)". On the left, there's a section for "TENA-MiddlewareSDK-v6.0.9.A" with a description of the software framework and a red oval highlighting the "Download" link. To the right, there's a large graphic of a purple X with the word "TENA" at the top. The arms of the X contain the words "ARMY", "INTEROPERABILITY", "COMPOSABILITY", "NAVAL FORCES", and "REUSE". Below this graphic, the text "TENA Middleware The interoperability solution." is displayed. At the bottom, there are links for "Overview | Documentation | Help | Versions | Contents | History".

TENA - Test and Training Enabling Architecture

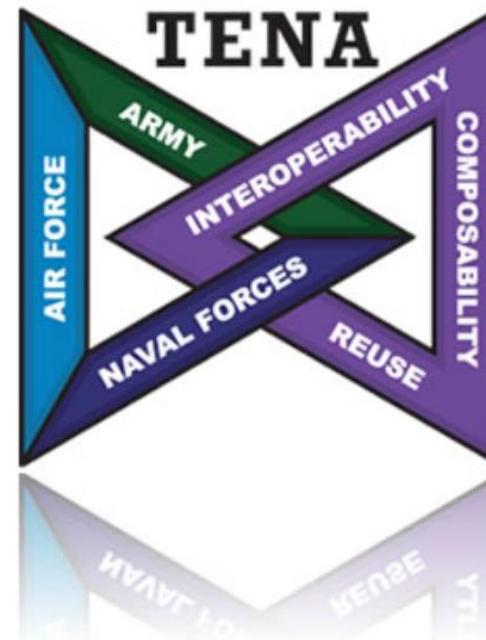
TENA Website Repository Home Components Help

[Repository Home](#) \ [Browse Components](#) \ [TENA](#) \ [MiddlewareSDK \(Product\)](#)

TENA-MiddlewareSDK-
v6.0.9.A

The TENA Middleware Software Development Kit (SDK) is a software framework that enables rapid and reliable development of scalable distributed systems.

[Download](#)



TENA
INTEROPERABILITY
COMPOSABILITY
ARMY
NAVAL FORCES
REUSE
AIR FORCE

TENA
Middleware
The interoperability solution.

Overview | Documentation | Help | Versions | Contents | History



Notes on the SDK download

- *Before we download the SDK...*
- What is a “platform”?
- Dissecting a platform string



What is a “platform”?

- A TENA “platform” is the combination of
 - Operating System (e.g. Windows 10)
 - Compiler/IDE/architecture (e.g. Visual Studio 2019, x86 64-bit)
 - Build type (release/optimized or debug)
- TENA supports platforms requested by DoD organizations
 - To request new platform support, start a helpdesk case at <https://www.trmc.osd.mil/helpdesk>
- *Not to be confused with a “TENA-Platform” SDO*
 - *The TENA-Platform SDO will be replaced by the LVC-Entity SDO*



Dissecting a platform string

- For this overview, we'll use "u2004-gcc9-64"
 - u2004: Ubuntu 20.04
 - gcc9: “GNU’s Not Unix” Compiler Collection 9
 - 64: 64-bit architectures
 - *Absence of “64” implies 32-bit architecture*
 - *Absence of “-d” on the end implies a release build*
 - *Appending “-d” implies a debug build*
- Now back to downloading...



Select your platform details, click “Next”, and save installer

The screenshot shows a 'Platform Selection' dialog box overlaid on a website page. The dialog box contains the following information:

- Component: TENA-MiddlewareSDK-v6.0.9.A
- Middleware Version: 6.0.9
- Please choose your operating system:
Linux (selected)
- Which flavor of the operating system?:
Ubuntu 20.04
- Which compiler?:
GCC 9
- 64-bits?
- Mode:
Optimized Version

At the bottom of the dialog box, there are four buttons: Back, Next, Cancel, and Finish. The 'Next' button is highlighted with a red circle.

In the background, the website page displays the TENA Middleware v6.0.9.A download page, featuring sections for 'TENA Website', 'Repository Home', 'TENA Applications', and 'TENA Tools'. A large banner on the right side of the page reads 'Middleware solution.'

What is the TENA Middleware?



Installing the Middleware SDK

- From TENA 6.0.9 forward
 - TENA uses native OS installation system, e.g.,
 - MSI files for Windows
 - RPM files for Red Hat
 - DEB files for Ubuntu
 - Refer to individual OS documentation for details on installation
- From TENA 6.0.8 backward
 - TENA used customer installers
 - EXE files for Windows
 - BIN files for Linux and Mac
 - Installer needs to be set for execution to run
 - Run installer, select installation location



TENA Execution Manager (EM) review

- TENA Execution
 - A collection of TENA-enabled apps that interact and share information
- TENA Execution Manager (EM)
 - Application that manages a TENA execution
 - TENA-enabled apps must connect to an EM
 - Contains info about an execution, including app membership
 - Only used when apps join, resign, change subscriptions, or send alerts
 - Not used for direct OM data distribution
 - Only one EM is needed per execution, but additional EMs may be used to provide fault tolerance
- We can start an EM
 - From a TENA Console (most common way)
 - From the command line
 - As a service (EM automatically starts when the computer is booted)

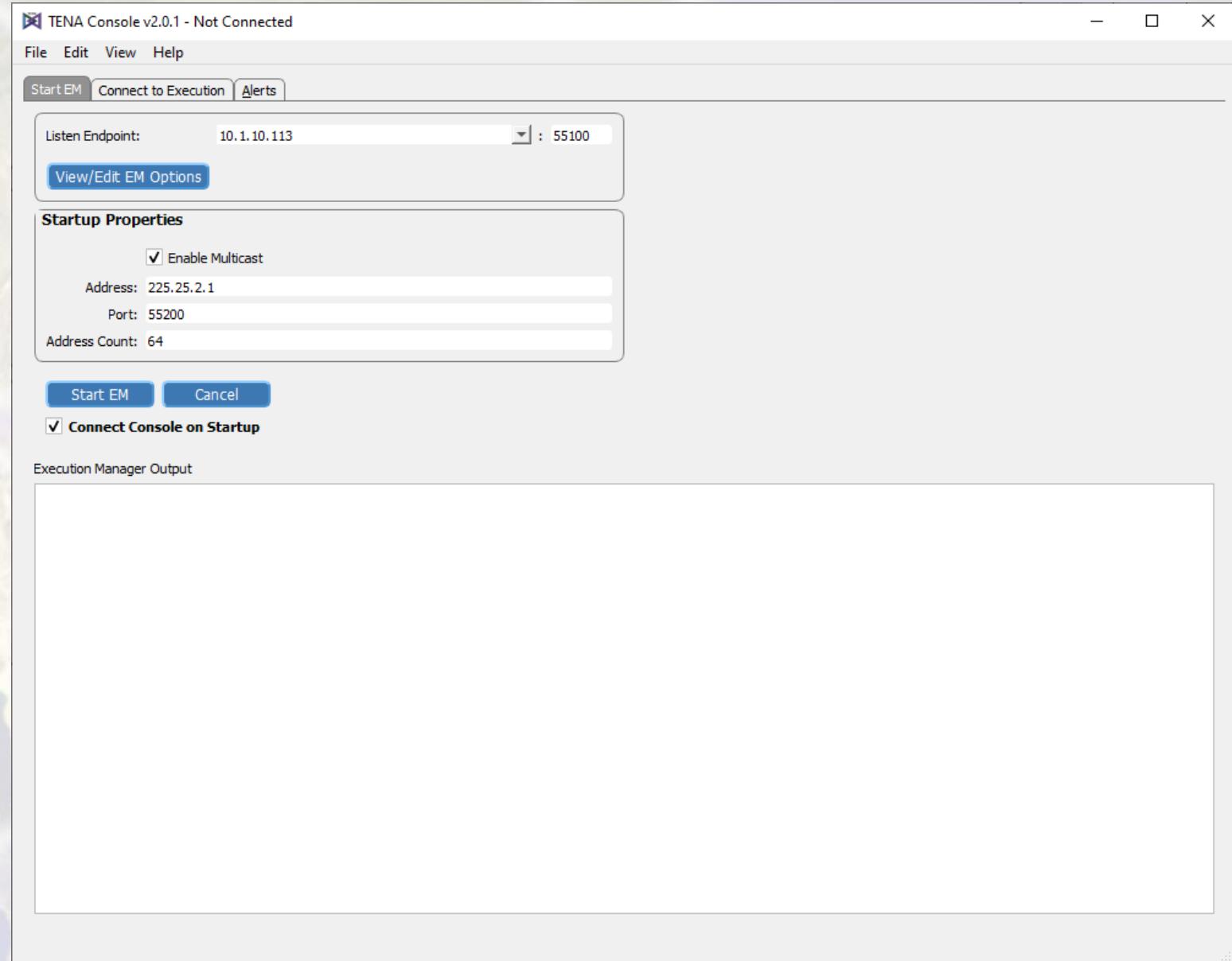


Starting an EM from the TENA Console

- Start TENA Console from desktop icon or “start script” (e.g., “/opt/TENA/Console-v2.0.1/start.sh &”)
- A new EM will be started from the “Start New EM” tab
- To allow example apps to join, we’ll have to allow arbitrary values in the execution
- TENA Console will need to join the EM after it is started
 - Monitors status of EM(s), apps, other Consoles, and alerts
 - Perform network tests as needed or desired



Starting an EM from a TENA Console





EM running from a TENA Console

TENA Console v2.0.1 - Connected to EM at 10.1.10.113:55100

File Edit View Help

EM Running Execution Execution OM Stats Applications Network Monitoring Execution Managers Consoles Alerts

Listen Endpoint: 10.1.10.113 : 55100

[View EM Options](#)

Startup Properties

Enable Multicast

Address: 225.25.2.1

Port: 55200

Address Count: 64

[EM Running](#) [Stop EM](#)

Connect Console on Startup

Execution Manager Output

```
executionManager Version: 6.0.9 untagged version 0
Middleware Version: 6.0.9
TENA Platform: w10-vs2022-64

To join this execution, use one of the following endpoints:
{[10.1.10.113:55100]}

For example: -emEndpoints 10.1.10.113:55100

Enter 'help' to display this list of commands.
Enter 'show version' to display the TENA Middleware version and the version of this EM.
Enter 'show config' to display the supplied command line and full configuration options.
Enter 'show endpoints' to display the endpoints of the execution manager.
Enter 'show allowables' to display the supplied configuration of allowable participants.
Enter 'show om impl rules' to display the supplied configuration of OM Implementation rules.
Enter 'set passphrase [<phrase>]' to set, or without a phrase to clear.
Enter 'force rollover' to forcibly rollover to a new primary execution manager.
Enter 'forcequit' or 'forceexit' to forcibly shutdown the execution manager.
Enter 'quit' or 'exit' to shutdown the execution manager.

Accepting requests ...

>
NOTE: EM ID 0: Attaching a console application with ID 1 listening on endpoint set {[10.1.10.113:55101]} [2022-10-06T15:11:05.203660Z]
>
```



EM running from command line interface

```
Command Prompt - C:\Users\cj\AppData\Local\Programs\TENA\executionManager-v6.0.9\start.bat -listenEndpoints 10.1.10.113:55100 -multicastProperties 225.25.2.1:55200:64
(c) Microsoft Corporation. All rights reserved.

C:\Users\cj>%TENA_HOME%\executionManager-v6.0.9\start.bat -listenEndpoints 10.1.10.113:55100 -multicastProperties 225.25.2.1:55200:64

This executionManager (EM) has been configured to prevent applications using ArbitraryValue instances, that are provided by default in auto-generated example source code, from joining the execution. Anywhere the auto-generated example application must supply an attribute value, an ArbitraryValue is used to emphasize that the value is just a "dummy" value. The configuration parameter "-allowArbitraryValue" can be used to allow applications that may be using an ArbitraryValue to join the execution. Consult the TENA Middleware Programmer's Guide for more information.

executionManager Version: 6.0.9 untagged version 0
Middleware Version: 6.0.9
TENA Platform: w10-vs2022-64

To join this execution, use one of the following endpoints:
{[10.1.10.113:55100]}

For example: -emEndpoints 10.1.10.113:55100

Enter 'help' to display this list of commands.
Enter 'show version' to display the TENA Middleware version and the version of this EM.
Enter 'show config' to display the supplied command line and full configuration options.
Enter 'show endpoints' to display the endpoints of the execution manager.
Enter 'show allowables' to display the supplied configuration of allowable participants.
Enter 'show om impl rules' to display the supplied configuration of OM Implementation rules.
Enter 'set passphrase [<phrase>]' to set, or without a phrase to clear.
Enter 'force rollover' to forcibly rollover to a new primary execution manager.
Enter 'forcequit' or 'forceexit' to forcibly shutdown the execution manager.
Enter 'quit' or 'exit' to shutdown the execution manager.

Accepting requests ...

>
```



Environment variables

- To build TENA apps, environment variables are needed
 - TENA_HOME
 - TENA_VERSION
 - TENA_PLATFORM
- The Windows installer will attempt to set these for you
- On Mac/Linux, you'll need to run a script
- Scripts can be used to switch between environments

Setting environment variables in Linux

- “Source” the tenaenv script for your environment, for example

```
source /opt/TENA/6.0.9/scripts/tenaenv-u2004-gcc9-64-v6.0.9.sh
```

- If you don’t want to source the tenaenv environment every time you need it, add the command to your .bashrc file (or equivalent)

```
echo "source /opt/TENA/6.0.9/scripts/tenaenv-u2004-gcc9-64-v6.0.9.sh &> /dev/null" >> ~/.bashrc
```

(redirecting output to /dev/null avoids some issues with programs like ssh clients connecting to your computer)

- You may then want to “re-source” your .bashrc file for the current shell (to test)

```
source ~/.bashrc
```

(However, for some programs this may require logging back out, then back in to take effect)

- Notes

- "source" is a built-in shell command that executes the contents of a file (a set of commands) passed as an argument in the current shell
 - You may also see the period (.) character used instead of “source”



Setting environment variables in Windows

- Run the tenaenv bat script
- Navigate to TENA SDK installation location, e.g.,
 - %HOMEPATH%\AppData\Local\Programs\TENA
- Navigate to a TENA version subdirectory, e.g.,
 - 6.0.9
- Navigate to the scripts subdirectory
- Run the Windows bat script, e.g.,
 - tenaenv-w11-vs2022-64-v6.0.9.bat



Checking your TENA environment variables

- Linux command: `env | grep TENA_`

- Output example:

`TENA_VERSION=6.0.9`

`TENA_HOME=/opt/TENA`

`TENA_PLATFORM=u2004-gcc9-64`

- Windows command: `set TENA_`

- Output example:

`TENA_HOME=C:\Users\cj\AppData\Local\Programs\TENA\`

`TENA_PLATFORM=w11-vs2022-64`

`TENA_VERSION=6.0.9`



Building/running auto-generated example apps



- C++ programming language support for Standard OMs is included in the SDK
 - For other OMs, download C++ bindings from Repository
- *Download .NET or Java bindings for OMs in Repository*
- C++ example apps are located at
 - Windows: %TENA_HOME%\%TENA_VERSION%\src
 - Linux/Mac: \$TENA_HOME\$\\$TENA_VERSION \src
- TENA API training for C++.NET, and Java is available upon request
- We will build and run one of the included example apps to verify our build environment

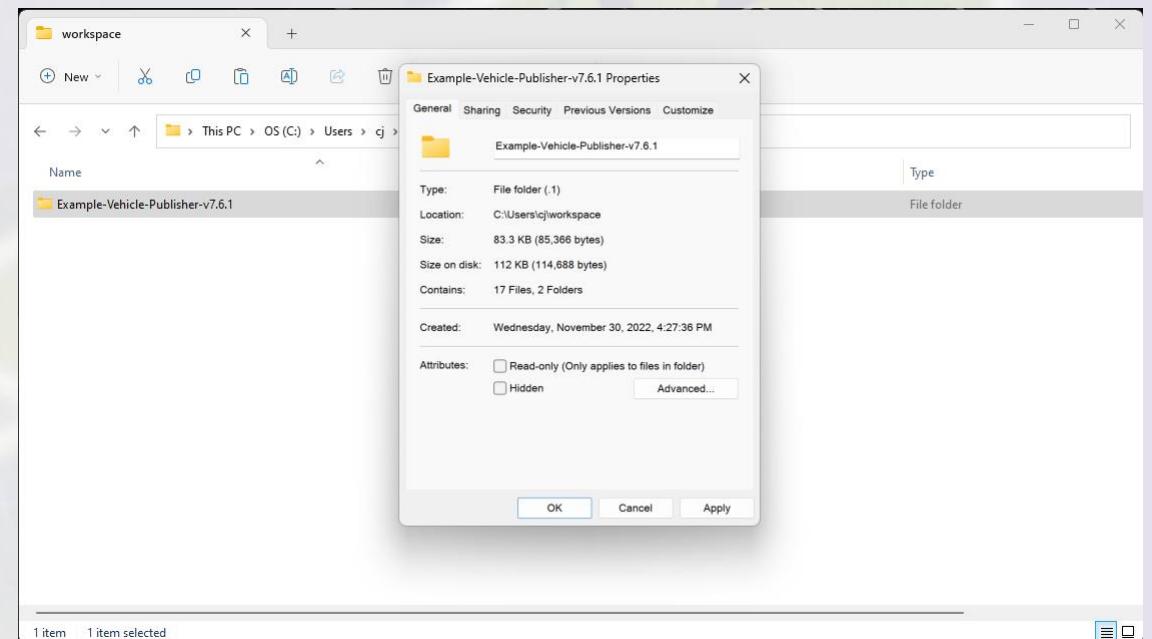


Build Example-Vehicle-v1 OM Vehicle SDO publisher

- Ensure build environment is configured correctly
 - Environmental variables set
 - Supported platform configuration with compiler/IDE
- Navigate to example apps root directory at `TENA_HOME/TENA_VERSION/src`
- Navigate to Example-Vehicle-v1 OM subdirectory
- Copy the Example-Vehicle-Publisher-v7.6.1 subdirectory to a workspace in your home directory
- Make the files in Example-Vehicle-Publisher-v7.6.1 writable
- We will show examples in Windows and Linux
 - w11-vs2022-64
 - u2004-gcc9-64

Windows

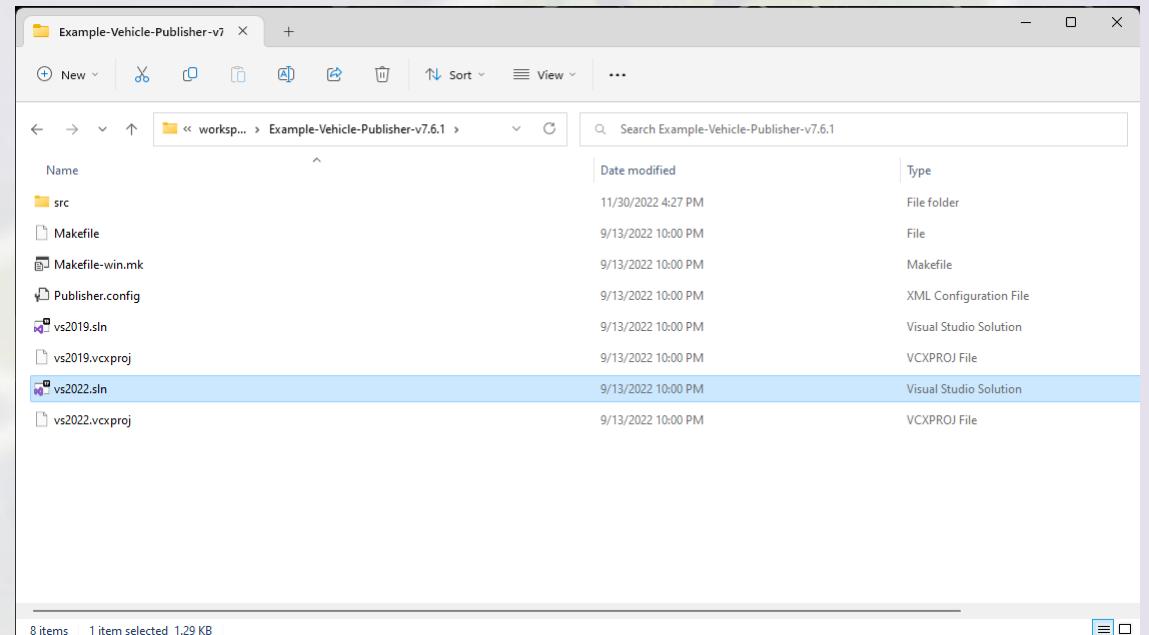
- Right-click on copy of project folder
- Go to properties in the menu
- Uncheck “Read-only” option
- Click OK





Windows

- Open the “sln” (solution) file with Visual Studio





Build the solution

The screenshot shows the Microsoft Visual Studio 2022 interface. The top navigation bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. A red circle highlights the 'Release' dropdown in the Build menu, which is currently set to 'x64'. The bottom left corner shows the Output window with tabs for Error List and Output.

The right side of the screen displays the Solution Explorer window, which lists the 'vs2022' solution containing one project. A red circle highlights the 'Build Solution' option in the context menu for the solution node. The Solution Explorer also shows properties for the solution, including the path 'C:\Users\cj\workspace\Example' and startup project 'vs2022'.

- Select the solution configuration that corresponds with the TENA SDK
 - e.g., “Release” for optimized builds of the SDK
- Right click on the solution in the solution explorer and click “Build Solution”
 - Or use the F7 key



Verify the solution built succeeded

The screenshot shows the Microsoft Visual Studio 2022 interface. The title bar reads "vs2022". The menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar "Search (Ctrl+Q)". The toolbar has icons for file operations like Open, Save, and Print. The status bar at the bottom shows "8 December 2022 Build succeeded".

The main area contains three windows:

- Solution Explorer**: Shows a single project named "vs2022".
- Properties**: Shows the properties for the "vs2022" solution, including the name as "vs2022", active configuration as "Release|x64", path as "C:\Users\cj\workspace\Example", and startup project as "vs2022".
- Output**: Shows the build log:

```
Build started...
1>----- Build started: Project: vs2022, Configuration: Release x64 -----
1>createServant.cpp
1>updateServant.cpp
1>ApplicationConfiguration.cpp
1>main.cpp
1>printFunctions.cpp
1>vs2022.vcxproj -> C:\Users\cj\workspace\Example-Vehicle-Publisher-v7.6.1\Publisher.exe
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

A red oval highlights the line "===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====".



Build example project in Linux

```
# Copy example app  
cp -fr /opt/TENA/6.0.9/src/Example-Vehicle-v1/Example-Vehicle-Publisher-v7.6.1 .  
  
# Make files writable  
chmod u+w -R Example-Vehicle-Publisher-v7.6.1  
  
# Change directories to example app copy  
cd Example-Vehicle-Publisher-v7.6.1  
  
# Build example app  
make  
  
# Run example app  
../Publisher -listenEndpoints 127.0.0.1 -emEndpoints 127.0.0.1:55100
```



Notes on “arbitrary values”

- Example apps use an “arbitrary” value class
- The EM can help ensure that non-arbitrary values are set in production environments
- When starting from a C++ example app to build a production app
 - Remove the preprocessor directive
`TENA_MIDDLEWARE_ALLOW_ARBITRARY_VALUE`
 - Remove the header files for arbitrary values
 - Set “real” values in your publisher (search for the word “arbitrary” in the example apps to find arbitrary values that need to be replaced)

File Edit Selection View Go Run Terminal Help



main.cpp

Makefile



```
101 # The name of the generated application
102 #
103 MAIN_FULL_NAME := Publisher
104 #
105 #
106 # Default target
107 #
108 all: dep $(MAIN_FULL_NAME)
109
110
111 PGM_SOURCES := \
112     src/TENA_LVC_Entity/RemoteMethodsImpl.cpp \
113     src/TENA_LVC_Entity/remoteMethods.h \
114     src/TENA_LVC_Entity/updateServant.cpp \
115     src/ApplicationConfig.h \
116     src/main.cpp \
117     src/printFunctions.cpp \
118
119 DEPS += $(patsubst %,$(GEN_DIR)/.dpnd/%.dpnd,$(PGM_SOURCES:.cpp=))
120 PGM_OBJS := $(patsubst %.cpp, $(OUT_OBJ_DIR)/%.o, $(PGM_SOURCES))
121
122 # Set to allow compilation of code that uses ArbitraryValues (i.e. Publishers)
123 #EXTRA_DEFINES += -DTENA_MIDDLEWARE_ALLOW_ARBITRARY_VALUE
124
125 DEFINES := -DTENA_PLATFORM=$(TENA_PLATFORM) $(CFG_DEFINES) $(EXTRA_DEFINES)
126
127 # Rule to generate $(DEPS) dependencies is defined in platform-configuration.mk
128 dep: $(DEPS)
129
130 ifneq (clean,$(MAKECMDGOALS))
131 -include $(DEPS)
132 endif
133
```

**Remove setting that defines “TENA_MIDDLEWARE_ALLOW_ARBITRARY_VALUE”
(here, we have “commented out” the setting in the Makefile)**



① 0 △ 0

Ln 132, Col 6 Tab Size: 4 UTF-8 LF Makefile



Removing “Arbitrary Values”

- Look for this line and remove it:
 - `#include <TENA/Middleware/ArbitraryValue.h>`
- It may appear in several places
- Search for the word “arbitrary” and replace any “arbitrary values” that are set



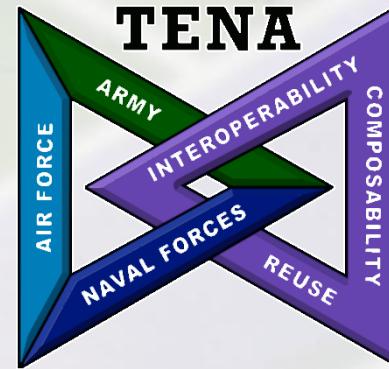
Help with example apps

- Run example apps with the “-help” option for a list of required and optional application configuration items
 - e.g., “./Publisher -help”



Demonstration

- Installing the SDK
- Navigating the SDK directory structure – location of:
 - TENA Execution Manager (EM) and TENA Console
 - TENA environment scripts
 - Middleware C++ header files & library
 - Standard OMs & Example OM
 - TDL files
 - C++ header files & libraries
 - (*C# bindings, Java bindings, etc. acquired via repository*)
 - Example application source code
 - SDO & Message – Publishers & Subscribers
- Verifying the build environment is configured to build and run TENA-enabled applications
 - By building/running TENA example applications



Exercise 1

TENA SDK installation and C++ build environment verification



Exercise 1 overview

- Acquire and install a TENA Middleware SDK
 - Includes example OM SDKs
- Configure and verify TENA related environment variables
- Start a TENA Console and local EM
- Build example app(s)
- Join local execution with example app(s)
- Join class execution
 - *(EM run by instructor with the Console presented via class projector)*



Exercise 1 Ubuntu 20.04, installing dependencies

- `sudo apt install g++ make libxcb-xinerama0`
 - sudo: for admin privileges
 - apt install: installation from repositories
 - g++: C++ compiler
 - libxcb-xinerama0: part of a Qt platform dependency for TENA Console



Exercise 1 Ubuntu 20.04, installing Middleware SDK

- Download the Middleware SDK from the repository and install it

```
sudo dpkg -i --force-overwrite \
TENA-MiddlewareSDK-v6.0.9.A@Product@u2004-gcc9-64-va3bc16dc.deb
```





Exercise 1 Linux, TENA environment variables



```
# "source" TENA environment script from .bashrc file
echo "source /opt/TENA/6.0.9/scripts/tenaenv-u2004-gcc9-64-v6.0.9.sh &>/dev/null" >> ~/.bashrc

# Re-source .bashrc in current terminal
source ~/.bashrc

# Verify TENA variables were set
env | grep TENA_
```



Exercise 1 Linux, start a Console from command line

/opt/TENA/Console-v2.0.1/start.sh &



Exercise 1 Linux, start an EM from command line

- You'll probably want to start an EM from a Console (GUI) instead, but you can start an EM from the command line like this (show with some options as examples):

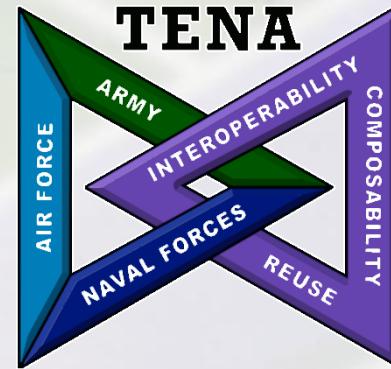
```
/opt/TENA/executionManager-v6.0.9/start.sh \
    -listenEndpoints 127.0.0.1:55100 \
    -allowArbitraryValue \
    -multicastProperties 225.25.2.1:55200:64 \
    -disableRecovery \
    -allowImplementationMismatch
```



Exercise 1 Linux, build/run example apps

```
cd ~  
cp -fr /opt/TENA/6.0.9/src/Example-Vehicle-v1/Example-Vehicle-Publisher-v7.6.1 .  
chmod u+w -R Example-Vehicle-Publisher-v7.6.1  
cd Example-Vehicle-Publisher-v7.6.1  
make  
../Publisher -listenEndpoints 127.0.0.1 -emEndpoints 127.0.0.1:55100
```

```
cd ~  
cp -fr /opt/TENA/6.0.9/src/Example-Vehicle-v1/Example-Vehicle-Subscriber-v7.6.1 .  
chmod u+w -R Example-Vehicle-Subscriber-v7.6.1  
cd Example-Vehicle-Subscriber-v7.6.1  
make  
../Subscriber -listenEndpoints 127.0.0.1 -emEndpoints 127.0.0.1:55100
```



Lesson 2

API code overview using auto-generated example applications
&
Creating a barebones TENA app that joins an execution



API code overview using auto-generated example applications



- Download OM from repository
- Install OM
- Start from SDO example apps
- Build/run SDO publisher and subscriber example apps



Downloading OM - Navigate to TRMC repository

This website is for UNCLASSIFIED USE ONLY. Do not discuss, enter, transfer, process, or transmit any CLASSIFIED information.

Spaces Calendars Create ... Search ?

TRMC Website (Confluence)

- Website Dashboard
- Group Information
- Group Space Directory
- Team Calendars
- TRMC Helpdesk (Jira)
- TRMC Bitbucket
- TRMC Artifacts

TRMC Repository

JMETC **CHEETAS** **TENA** **NCRC**

Edit **Save for later** **Watch** ...

Resource Management Center

Test and Training Enabling Architecture (TENA)

The United States Office of the Secretary of Defense Test Resource Management Center (TRMC) has developed a common architecture to support effective integration and reuse of testing, training, and simulation capabilities that require real-time collaboration between distributed computer systems operating within diverse testing and training environments. The architecture links test and training assets across the DoD and industry users.

National Cyber Range Complex (NCRC)

The National Cyber Range Complex is a holistic, integrated cyber range capability comprising the National Cyber Range (NCR), Regional Service Delivery Points (RSDP), and the Joint Mission Environment Test Capability (JMECTC) Multiple Independent Levels of Security (MILS) Network (JMN). The NCRC provides Department of Defense (DoD) and other government and industry users with

Website Profile

Capacity (JMECTC)

The JMECTC mission is to provide a persistent capability for linking distributed facilities, enabling DoD customers to develop and test warfighting capabilities in a Joint Context. JMECTC provides a test infrastructure consisting of the components necessary to conduct Joint distributed test events by cost-effectively integrating live, virtual, and constructive

Test and Training Enabling Architecture (TENA)

The United States Office of the Secretary of Defense Test Resource Management Center (TRMC) has developed a common architecture to support effective integration and reuse of testing, training, and simulation capabilities that require real-time collaboration between distributed computer systems operating within diverse testing and training environments. The architecture links test and training assets across the DoD and industry users.

National Cyber Range Complex (NCRC)

The National Cyber Range Complex is a holistic, integrated cyber range capability comprising the National Cyber Range (NCR), Regional Service Delivery Points (RSDP), and the Joint Mission Environment Test Capability (JMECTC) Multiple Independent Levels of Security (MILS) Network (JMN). The NCRC provides Department of Defense (DoD) and other government and industry users with

Downloading OM - All Products and Object Models

Welcome

The TENA Repository is a web-based storehouse for TENA interoperability solutions.



TENA Repository

Discover, reuse and share
TENA components



Getting Started

Quick Start Guide

Information describing how to use the TENA Repository.

All Products and Object Models

Browse products and object models from all groups

TENA Standard Object Models

A Collection of C++ SDKs for the TENA Standard Object Models

TENA Canary

The TENA Canary is a simple application that can connect to an execution to help test the network.

Featured Products and Object Models

Featured Products and Object Models

Submit Object Model

Submit or revise an object model

TENA Middleware

The TENA Middleware is a software framework that enables rapid and reliable development of scalable

TENA Console

The TENA Console is a tool that assists in the troubleshooting and monitoring of TENA executions.



Downloading OM - VUG group

Products and Object Models in OASUW

PRITEC

Products and Object Models in PRITEC

RTC

Products and Object Models in RTC

RelayNode

Products and Object Models in RelayNode

SIMDIS

Products and Object Models in SIMDIS

SUMS

Products and Object Models in SUMS

TA

Products and Object Models in TA

TDCS

Products and Object Models in TDCS

TSPIL

Products and Object Models in TSPIL

VENGEANCE

Products and Object Models in VENGEANCE

WSMR

Products and Object Models in WSMR

tmp

Products and Object Models in tmp

Products and Object Models in OneSAR

ProLogic

Products and Object Models in ProLogic

RTTC

Products and Object Models in RTTC

SAIC

Products and Object Models in SAIC

SPAWAR

Products and Object Models in SPAWAR

SimShield

Products and Object Models in SimShield

TACE

Products and Object Models in TACE

TENA

Products and Object Models in TENA

TVDS

Products and Object Models in TVDS

VPEF

Products and Object Models in VPEF

Weibel

Products and Object Models in Weibel

Products and Object Models in PMKRF

RMAST

Products and Object Models in RMAST

Reap2ir

Products and Object Models in Reap2ir

SATIN

Products and Object Models in SATIN

STAT

Products and Object Models in STAT

Simmetry

Products and Object Models in Simmetry

TCS

Products and Object Models in TCS

TRCE

Products and Object Models in TRCE

USNTTR

Products and Object Models in USNTTR

VUG

Products and Object Models in VUG

YTC

Products and Object Models in YTC



Downloading OM - VUG-Vehicle OM

VUG-Entity-v0.1.0	ObjectModel	Released Public
VUG-Pedestrian-v0.2.0	ObjectModel	Released Public
VUG-Scenario-v0.5.0	ObjectModel	Released Public
VUG-TrafficControl-v0.4.0	ObjectModel	Released Public
VUG-TrafficLight-v0.3.0	ObjectModel	Released Public
VUG-VOICES-Combined-v0.12.0	ObjectModel	Released Public
VUG-Vehicle-v0.6.0	ObjectModel	Released Public



Downloading OM - Download C++ OM SDK

TENA - Test and Training Enabling Architecture

TENA Website Repository Home Components Help

Repository Home \ Browse Components \ VUG \ Vehicle (ObjectModel)

 **VUG-Vehicle-v0.6.0** Versions

Download (highlighted with a red box)

TDL

C++ Binding (highlighted with a red box)

Java Binding

.NET Binding

Data Collection System

Web Binding

RelayNode

OM Spreadsheets

Dependencies: Public Imports: TENA-Geospatial-PointOfInterest-v1.0.0, TENA-Geospatial-TSPlcovariance-v1.0.0, TENA-Hardware-System-v1.0.0, (...6 more...) Create Helpdesk Case Middleware: 6.0.9, 6.0.8, 6.0.7

Superseded By: None
Supersedes: None

TDL | Documentation | History

```
Vehicle.tdl
1 /**
2  * This file contains the definition of and documentation for the Vehicle and Track::BSM object model
3  * for VOICES. The current content is based off of the SAE J2735 Basic Safety Message and the
4  * following messages from CARMA:
5  * https://github.com/usdot-fhwa-stol/carma-msgs/blob/develop/cav_msgs/msg/ExternalObject.msg
6  * https://github.com/usdot-fhwa-stol/carma-msgs/blob/develop/cav_msgs/msg/DriverStatus.msg
7  * http://docs.ros.org/en/jade/api/geometry_msgs/html/msg/Twist.html
8 */
```

Downloading OM - Platform selection

TENA Website Repository Home

Repository Home \ Browse

 **VUG-Vehicle**

Download

State: Released Restricted

Updated:

Point of Contact: [VUG](#)

Component: VUG-Vehicle-v0.6.0
Distribution: C++ Binding
Middleware Version: 6.0.9
Please choose your operating system:

Which flavor of the operating system?:

Which compiler?:

64-bits?

Mode:

1 `/// \file VUG-V`
2 `///`
3 `/// This file c`
4 `/// for VOICES.`
5 `/// following m`
6 `/// https://git`
7 `/// https://git`
8 `/// http://docs`
9 `/// http://docs.ros.org/en/jade/api/geometry_msgs/html/msg/Pose.html`
10 `///`
11 `/// The VUG::Entities::ControllableVehicle extends the base VUG::Entities::Vehicle class to add the`
12 `/// basics for requesting control of a Vehicle. Generally this class will be used as a base class`

Back **Next** Cancel Finish

Downloading OM - Save file

TENA Website Repository Home

[Repository Home \ Browse](#)

 **VUG-Vehicle**

[Download](#)

State: Released Restrictions:

Updated:

Point of Contact: [VUG](#)

1 /// \file VUG-Vehicle
2 ///
3 /// This file contains API documentation for VOICES.
4 /// for VOICES.
5 /// following message types:
6 /// <https://gitlab.com/voicessys/vug-vehicle>
7 /// <https://gitlab.com/voicessys/vug-vehicle/blob/main/doc>
8 /// http://docs.ros.org/en/jade/api/geometry_msgs/html/msg/Pose.html
9 /// http://docs.ros.org/en/jade/api/geometry_msgs/html/msg/Pose.html
10 ///
11 /// The VUG::Entities::ControllableVehicle extends the base VUG::Entities::Vehicle class to add the
12 /// basic functionality for requesting control of a Vehicle. Generally this class will be used as a base class.

Download VUG-Vehicle-Distribution-v0.6.0 Product (u2004-gcc9-64)
VUG-Vehicle-Distribution-v0.6.0 Product (u2004-gcc9-64) is ready for download.

Your download should begin automatically. The download size is 54.21 MB (56853240 bytes). Please verify the size of the downloaded file. [Click for manual download](#).

Click 'Finish' to exit.

Save As

Downloads

No items match your search.

File name: VUG-Vehicle-Distribution-v0.6.0@Product@u2004-gcc9-64-v7309e963.deb

Save as type: DEB File (*.deb)

Save Cancel

6.0.9
6.0.8
6.0.7

Back Next Cancel Finish

8 December 2023

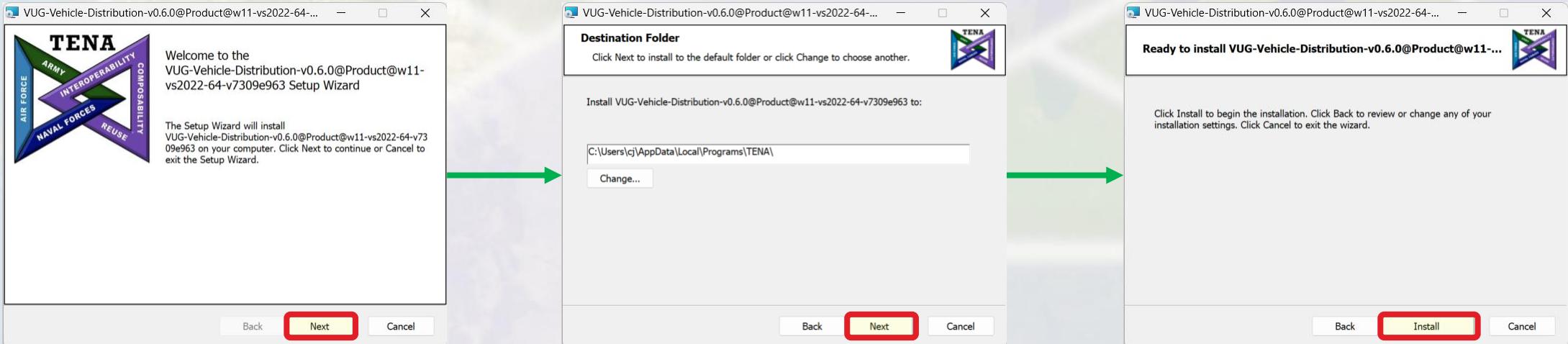


Installing OM SDKs

- From TENA 6.0.9 forward
 - TENA uses native OS installation system, e.g.,
 - MSI files for Windows
 - RPM files for Red Hat
 - DEB files for Ubuntu
 - Refer to individual OS documentation for details on installation
- From TENA 6.0.8 backward
 - TENA used customer installers
 - EXE files for Windows
 - BIN files for Linux and Mac
 - Installer needs to be set for execution to run
 - Run installer, select installation location



Windows OM SDK installer



- Installer file: VUG-Vehicle-Distribution-v0.6.0@Product@w11-vs2022-64-v7309e963.msi
- Note the installation for future reference:
- Default: AppData\Local\Programs\TENA in your home folder
 - AppData is a “hidden” folder

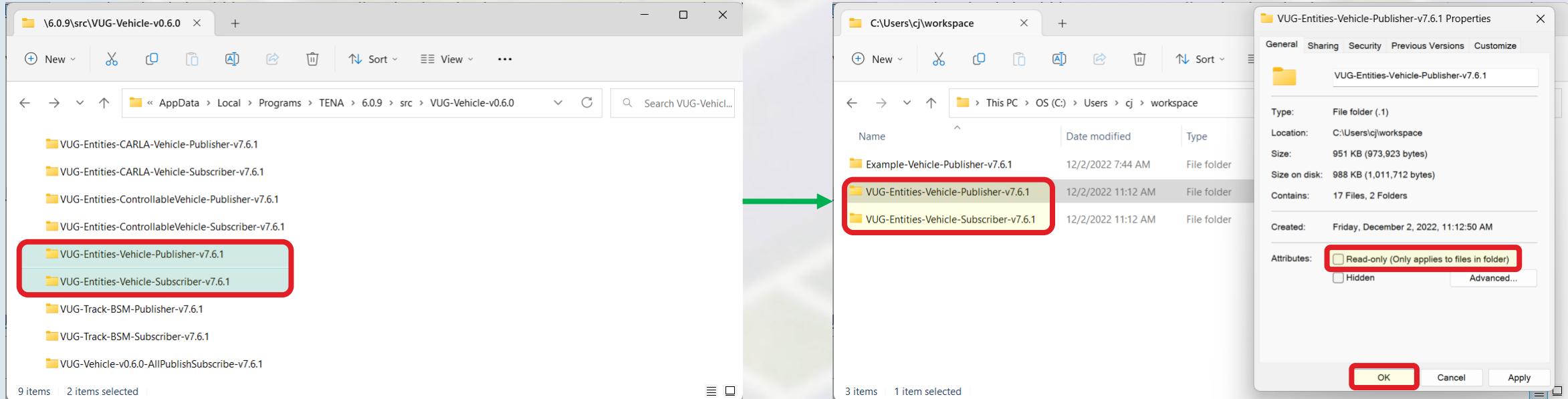


Ubuntu Linux OM SDK installer

```
sudo dpkg \  
  -i \  
  --force-overwrite \  
  VUG-Vehicle-Distribution-v0.6.0@Product@u2004-gcc9-64-v7309e963.deb
```

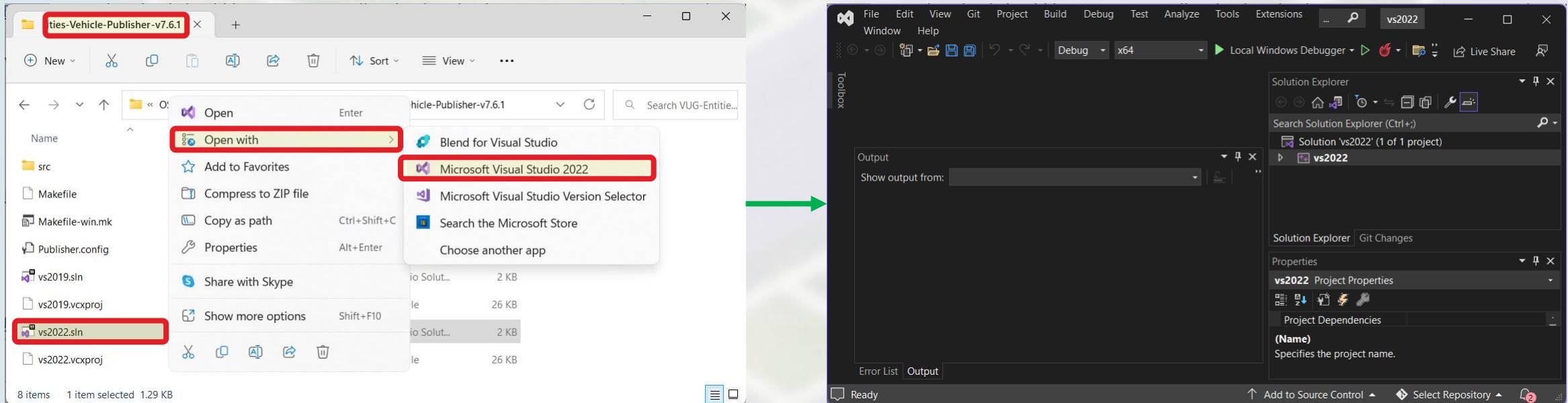
- “sudo”: needed to run dpkg as root
- “-i”: for installation
- “--force-overwrite”: needed to deal with file location conflicts
 - *Known issue in TENA 6.0.9 Ubuntu packages*

Windows – copy example apps from OM SDK installation location



- OM SDK example apps default location:
 - Windows: AppData\Local\Programs\TENA\6.0.9\src (in home path)
 - Linux: /opt/TENA/6.0.9/src
- OM subdirectory for our examples: VUG-Vehicle-v0.6.0
- Copy the following subdirectories to another location
 - VUG-Entities-Vehicle-Publisher-v7.6.1
 - VUG-Entities-Vehicle-Subscriber-v7.6.1
- Make the folders (and subfolders/files) writable

Windows – open, build, and run the solutions



- In VUG-Entities-Vehicle-Publisher-v7.6.1
 - Open “sln” (solution) file in Visual Studio



Windows – build the solution

The screenshot shows the Microsoft Visual Studio 2022 IDE. The title bar says "vs2022". The menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. The toolbar has icons for file operations like Open, Save, and Print. The status bar at the bottom shows "Build succeeded" and the date "8 December 2022".

Solution Explorer: Shows the solution 'vs2022' with one project 'vs2022' selected.

Properties: Shows the properties for the 'vs2022' solution. Under the 'Misc' section, the 'Name' is set to 'vs2022', 'Active config' is 'Release|x64', and 'Path' is 'C:\Users\cj\workspace\VUG-Entities-Vehicle-Publisher-v7.6.1'. The 'Startup project' is also set to 'vs2022'.

Output: Shows the build log:

```
Output  
Show output from: Build  
Build started...  
1>----- Build started: Project: vs2022, Configuration: Release x64 -----  
1>createServant.cpp  
1>updateServant.cpp  
1>ApplicationConfiguration.cpp  
1>main.cpp  
1>printFunctions.cpp  
1>vs2022.vcxproj -> C:\Users\cj\workspace\VUG-Entities-Vehicle-Publisher-v7.6.1\Publisher.exe  
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ======
```

Bottom Status Bar: Add to Source Control, Select Repository, and a refresh icon.

- Select the solution configuration that corresponds with the TENA SDK
 - e.g., “Release” for optimized builds of the SDK
- Right click on the solution in the solution explorer and click “Build Solution”
 - Or use the F7 key
- The output should indicate that everything built successfully
 - All succeeded
 - None failed



Demonstration

- Downloading an OM from the repository
- Installing an OM
- Building and running example apps from the OM
- Example app design and structure overview
 - Build file(s)
 - Windows: Visual Studio files
 - Linux: Makefile
 - Required TENA Middleware and OM
 - “Include” directories
 - Shared libraries directory
 - Shared library files
 - Preprocessor directives
 - Compiler configuration
- Common “main()” function behavior
 - Application configuration
 - TENA initialization, runtime, execution, and session(s)
- Specific example app structure and behavior for
 - SDO publication (and RMI implementation when applicable) – servant creation and updating
 - SDO subscription – observers, events, evoking callbacks, and working with proxy lists
 - Message publication – message senders and messages
 - Message receiving – observers, events, and evoking callbacks



Creating a barebones TENA app that joins an execution



- C++ project files
 - Linux Makefile



C++ project files

- Auto-generated example app project files
 - Linux/Mac: “Makefile” examples provided
 - Windows: Visual Studio project file examples provided
 - Can be modified for a new app if you are “starting from scratch”, or...
 - Can be used as an example for integrating TENA into an existing app
 - Does not need to be based on Visual Studio or Makefile
- Project files configure...
 - “Include” directories for header files
 - For the Middleware and OM dependencies
 - Linked libraries and their directories
 - For the Middleware and OM dependencies
 - Preprocessor definitions:
 - `TENA_PLATFORM=$(TENA_PLATFORM)`
 - Windows only:
 - Runtime library: Multithreaded DLL (Release or Debug)
 - etc.



Linux Makefile

- Minimal g++ compiling configuration
 - `-std=c++17`: Use C++ 17 standard
 - `-pthread`: Adds support for multithreading with the pthreads library (sets flags for both the preprocessor and linker)
 - `-I$(TENA_HOME)/$(TENA_VERSION)/include`: location of the Middleware C++ header files
 - *Additionally, you will need to include header file locations for any OM libraries you use (and for OM implementation libraries for OMs with custom constructors or methods)*
 - `-DTENA_PLATFORM=$(TENA_PLATFORM)`: Determines various options in how the Middleware is compiled for specific platforms
 - `-D_REENTRANT`: legacy definition intended to ensure g++ use thread safe versions of certain functions
- Minimal g++ linking configuration
 - `-std=c++17`
 - `-pthread`
 - `-L$(TENA_HOME)/lib`: location of Middleware (and OM) libraries
 - `-LTENA_Middleware-$(TENA_PLATFORM)-v$(TENA_VERSION)`: TENA Middleware library
 - *Additionally, you will need to link against any OM libraries you use (and OM implementation libraries for OMs with custom constructors or methods)*



Minimal Makefile example (just Middleware, no OMs)

all:

```
# Compile  
g++ -std=c++17 -pthread -I$(TENA_HOME)/$(TENA_VERSION)/include \  
-DTENA_PLATFORM=$(TENA_PLATFORM) -D_REENTRANT -c app1.cpp -o app1.o  
  
# Link  
g++ -std=c++17 -pthread app1.o -L$(TENA_HOME)/lib \  
-lTENA_Middleware-$(TENA_PLATFORM)-v$(TENA_VERSION) -o app1  
  
rm app1.o
```

clean:

```
rm -f app1 app1.o
```



Minimal C++ file - to just join a TENA execution

- Include the TENA/Middleware/config.h header file
- Create a `TENA::Middleware::ApplicationConfiguration`
- Initialize the Middleware and obtain its runtime pointer
- Join the execution from the runtime pointer



Include the TENA/Middleware/config.h header file

- `#include <TENA/Middleware/config.h>`
- In exercise 2, we will also leverage some other TENA Middleware header files, and header files from the C++ standard template library (STL)
 - `#include <TENA/Middleware/ApplicationConfiguration.h>`
 - `#include <TENA/Middleware/Utils/executablePathname.h>`
 - `#include <TENA/Middleware/Utils/ExitDetector.h>`
 - `#include <TENA/Middleware/Utils/Debug/TENAostream.h>`
 - `#include <TENA/Middleware/RuntimePtr.h>`
 - `#include <iostream>`
 - `#include <chrono>`
 - `#include <thread>`
 - `#include <vector>`



Create a TENA::Middleware::ApplicationConfiguration

- There are several ways to configure the Middleware, which are covered in the TENA programmers guide
 - The example apps create a custom extension of ApplicationConfiguration that parses the command line, configuration file, and environment variables
 - We will use a more basic approach to methodically (tediously?) construct an ApplicationConfiguration in exercise 2
 - Create a TENA::Middleware::Utils::BasicConfiguration::KeyValueList
 - Create a TENA::Middleware::Utils::BasicConfiguration::KeyValuePair for listenEndpoints
 - Add the KeyValuePair listenEndpoints to the KeyValueList
 - Create a TENA::Middleware::ApplicationConfiguration from the KeyValueList



Create a TENA::Middleware::ApplicationConfiguration from a KeyValueList

using namespace **TENA::Middleware;**

```
// Create a KeyValueList  
Utils::BasicConfiguration::KeyValueList keyValueList;
```

```
// Create a KeyValuePair for ListenEndpoints  
Utils::BasicConfiguration::KeyValuePair listenEndpoints(  
    "listenEndpoints", "127.0.0.1");
```

```
// Add the KeyValuePair ListenEndpoints to the KeyValueList  
keyValueList.push_back(listenEndpoints);
```

```
// Create an ApplicationConfiguration from the KeyValueList  
ApplicationConfiguration config(keyValueList, "");
```



Initialize the Middleware and obtain its runtime pointer

- Using the ApplicationConfiguration object we created (and named “config”), call its `tenaConfiguration()` method, which returns a `TENA::Middleware::Configuration` object
 - Pass the `TENA::Middleware::Configuration` to the `TENA::Middleware::init()` API call, which initializes the Middleware and returns a `TENA::Middleware::RuntimePtr` object
- Or, in short, create a `RunTimePtr` from the config...

```
TENA::Middleware::RuntimePtr runtime(
    TENA::Middleware::init(
        config.tenaConfiguration()));
```



TENA Middleware Runtime

- TENA apps must use “TENA::Middleware::init()” to initialize the Middleware
 - Only means to obtain a valid runtime pointer
 - Every TENA app must contain exactly one instance of a runtime
 - Invoking init() while a runtime exists will result in a “RuntimeAlreadyExists” exception



TENA::Middleware::ApplicationConfiguration

- Used in the auto-generated example apps (not in our barebones app here)
- Helper class to parse options from
 - Command line
 - Configuration file
 - Environment variables
- Provides a TENA::Middleware::Configuration object that can be used to initialize the Middleware and create a TENA Middleware runtime pointer

Join the execution from the runtime pointer

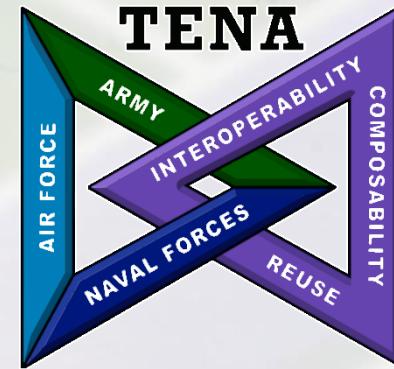
- For this, we need to know the execution endpoint(s)
 - As a minimal example, we can “hard code” the EM endpoints like this:
 - `vector<TENA::Middleware::Utils::Endpoint> emEndpoints;`
 - `TENA::Middleware::parseEndpointString("127.0.0.1:55100", emEndpoints);`
 - The example apps illustrate another way to parse the EM endpoints from parameters provided by the app user
- From the RuntimePtr object we created previously (and named “runtime”), we can call its `joinExecution()` method, passing in the endpoint vector that we prepared – which will attempt to join the execution and return an ExecutionPtr object:

```
TENA::Middleware::ExecutionPtr execution(  
    runtime->joinExecution(emEndpoints));
```



A note on exercises 2 through 9

- Each exercise includes a “start” and a “solution” folder
- Attendees should start in the start folder to work on and complete the exercise
 - The start folders will contain projects at various levels of completion, depending on the goals of the exercise
 - Feel free to refer to various sources to complete the exercises
 - TDL file(s) for the corresponding OM in \$TENA_HOME/tdl
 - Auto-generated example apps for SDOs and messages in \$TENA_HOME/\$TENA_VERSION/src
 - These slides (included in the class pack files)
 - Especially the notes slides following the main exercise slide
 - “solution” folder contains “a” solution (not the only possible solution)
 - It may be helpful to compare the start and solution (e.g., “diff -bur start/ solution/”)
 - Also, feel free to ask instructors for help



Exercise 2

Create a barebones TENA app that joins an execution



Exercise 2

- Start from the exercise-02\start folder
- Modify the Makefile to build a TENA app, linked against the Middleware
- Modify the C++ source code to build an app that joins a TENA execution
- Verify your app joins the class execution



Exercise 2 Linux, unzipping the “class pack” files

- Download the classpack zip file and unzip it to your home directory

<https://www.trmc.osd.mil/wiki/display/TRAINING/vug22>

```
cp ~/Downloads/classpack-* .zip ~/  
cd ~  
unzip classpack-* .zip
```





Exercise 2 Linux, start in the exercise-02/start directory

```
cd ~/classpack/exercises/exercise-02/start
```



Exercise 2 Linux, Makefile g++ lines

```
# Compile  
g++ -std=c++17 -pthread  
    -I$(TENA_HOME)/$(TENA_VERSION)/include \  
    -DTENA_PLATFORM=$(TENA_PLATFORM) -D_REENTRANT \  
    -c app1.cpp -o app1.o
```

```
# Link  
g++ -std=c++17 -pthread app1.o -L$(TENA_HOME)/lib \  
    -lTENA_Middleware-$(TENA_PLATFORM)-v$(TENA_VERSION) \  
    -o app1
```



Exercise 2 app1.cpp

- Added to app1.cpp

```
# Create a runtime
TENA::Middleware::RuntimePtr runtime(
    TENA::Middleware::init(config.tenaConfiguration()));

# Join an execution
TENA::Middleware::ExecutionPtr execution(
    runtime->joinExecution(emEndpoints));

# Leave the execution by explicitly destroying the execution pointer
execution.reset();

# Explicitly destroy the runtime pointer
runtime.reset();
```

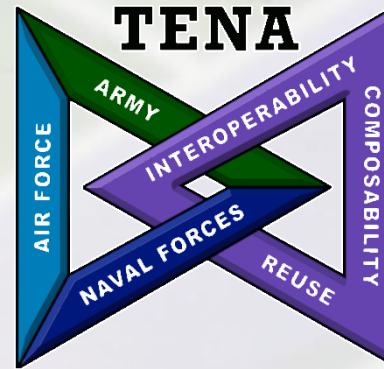


Exercise 2 Linux, build and run

```
make
```

```
./app1
```

```
# verify that app joined execution!
```



Lessons 3 through 9

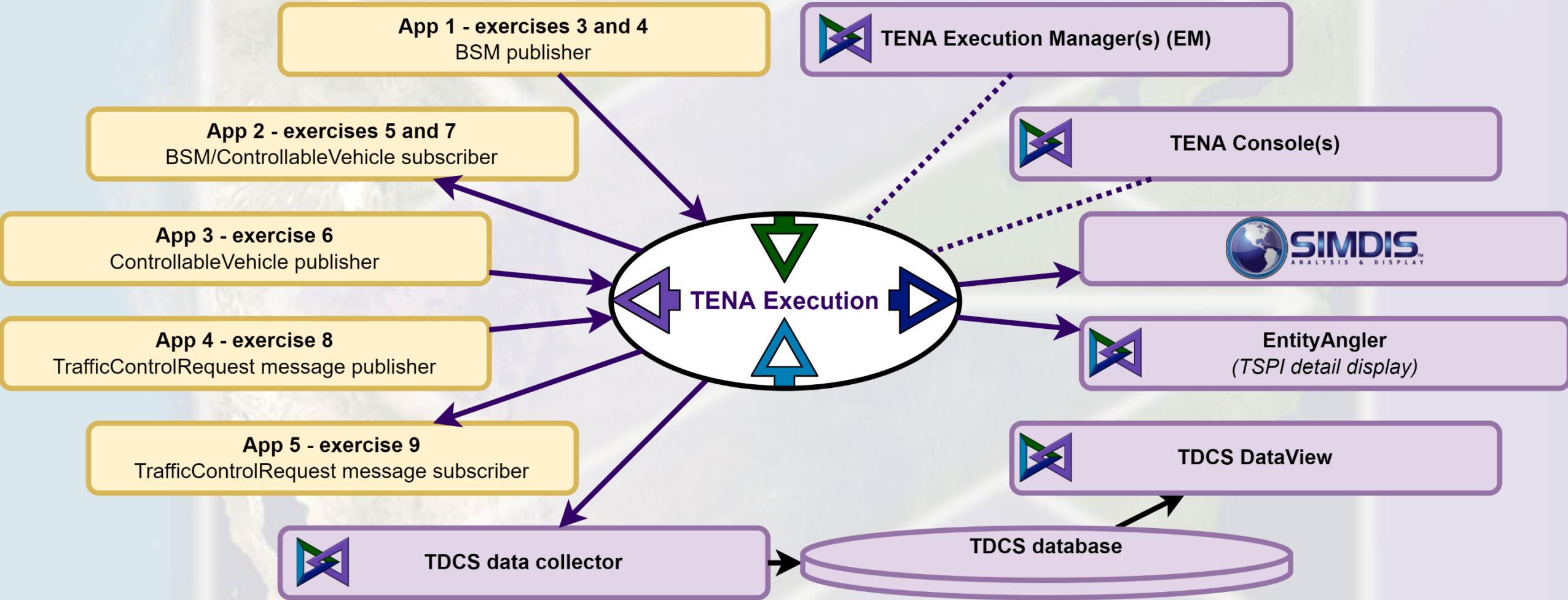
Overview

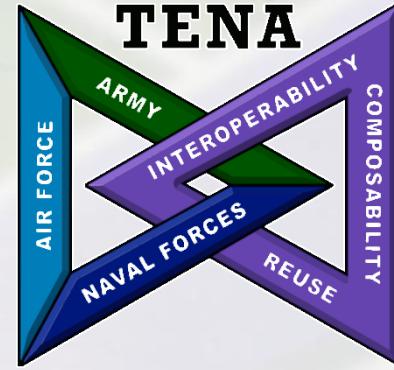


Exercise 3 through 9

- We will build 5 apps using
 - VUG-Vehicle-v0.6.0 OM: BSM & ControllableVehicle SDOs
 - App1: BSM SDO publisher
 - Exercise 3: SDO creation
 - Exercise 4: SDO updating
 - App2: BSM & ControllableVehicle subscriber
 - Exercise 5: SDO subscription
 - Exercise 7: Calling remote methods
 - App3: ControllableVehicle SDO publisher
 - Exercise 6: Implementing remote methods
 - VUG-TrafficControl-v0.4.0 OM: TrafficControlRequest message
 - App4: TrafficControlRequest message publisher
 - Exercise 8: Message publication
 - App5: TrafficControlRequest message subscriber
 - Exercise 8: Message subscription

Exercise 3 through 9 apps





Lesson 3

Creating an SDO

VUG OM SDOs and messages

- In the demo, we found the VUG-Vehicle-v0.6.0 OM in the repo, downloaded its C++ distro, installed it, and built the publisher and subscriber for Vehicle SDOs
- For the current exercises we will focus on the **BSM SDO** – which as no remote method to implement



Dissecting the BSM SDO

- Starting with class BSM, we would like to know more about its class structure and methods, including what it inherits



VUG-Vehicle OM

[Repository Home](#) \ [Browse Components](#) \ [VUG](#) \ Vehicle (ObjectModel)



VUG-Vehicle-v0.6.0

[Versions](#)

[Download](#)

[State:](#) Released

[Restrictions:](#) Public

[Imports:](#) [TENA-Geospatial-PointOfInterest-v1.0.0](#)

[Create Helpdesk Case](#)

[Middleware:](#) 6.0.9

[TENA-Geospatial-TSPlcovariance-v1.0.0](#)

6.0.8

[TENA-Hardware-System-v1.0.0](#)

6.0.7

[\(...6 more...\)](#)

[Updated:](#)

[Superseded By:](#) None

[Point of Contact:](#) [VUG](#)

[Supersedes:](#) None

[TDL](#) | [Documentation](#) | [History](#)

```
1  /// \file VUG-Vehicle.tdl
2  /**
3  * This file contains the definition of and documentation for the Vehicle and Track::BSM object model
4  * for VOICES. The current content is based off of the SAE J2735 Basic Safety Message and the
5  * following messages from CARMA:
6  * https://github.com/usdot-fhwa-stol/carma-msgs/blob/develop/cav\_msgs/msg/ExternalObject.msg
7  * https://github.com/usdot-fhwa-stol/carma-msgs/blob/develop/cav\_msgs/msg/DriverStatus.msg
8  * http://docs.ros.org/en/jade/api/geometry\_msgs/html/msg/Twist.html
9  * http://docs.ros.org/en/jade/api/geometry\_msgs/html/msg/Pose.html
10 /**
11 * The VUG::Entities::ControllableVehicle extends the base VUG::Entities::Vehicle class to add the
12 * basics for requesting control of a Vehicle. Generally this class will be used as a base class
13 * for a class that is specific to a particular simulation's API, such as the VUG::Entities::CARLA::Vehicle.
14 /**
15 * The VUG::Entities::CARLA::Vehicle is an extension of the VUG::Entities::ControllableVehicle class and
16 * uses the CARLA Vehicle class reference:
17 * http://carla.org/Doxygen/html/d9/dc6/classcarla\_1\_1client\_1\_1Vehicle.html#a278374757fcc52dbb06a84a04c5189c7
18 * specific to CARLA to the Vehicle. The carla::client::Vehicle ApplyControl was used as
```



BSM SDO in VUG-Vehicle OM

```
208  
209 package Track {  
210  
211     class BSM : extends TENA::Instrumentation::Track::State3D {  
212  
213         const LVCindicator lvcIndicator;      ///< Indicates if the entity is live, virtual, or constructive  
214  
215         const vector<octet> basicSafetyMessageIdentifier;    ///< bsm id is of the form [0xff,0xff,0xff,0xff]  
216  
217         boolean validLatitude;    ///< A BSM can signify that the position data in the message is not valid. This  
218         boolean validLongitude;   ///< flag represents that state. For the initial virtual/constructive BSM Track  
219         boolean validElevation;   ///< publishers this will always be true. Other TSPI attributes will be unset  
220             ///< if BSM indicated they are invalid.  
221  
222         int8 messageCount;       // A sequence number within a stream of messages from the same sender. A sender may initialize  
223             // this value to any value in the range 0..127 when sending the first message. It may also  
224             // be reinitialized if the sender changes identity. Depending on the application, the sequence  
225             // number may increment by one message or remain constant during a stream of messages when the  
226             // content within each message has not changed from the prior message sent. The value after  
227             // 127 is reserved.  
228         uint16 msWithinMinute;    // The VDX record expressed in this data element consists of integer values from 0..60999,  
229             // representing the milliseconds within a minute. A leap second is represented by the value  
230             // 61000. The values 61000..565534 are reserved.  
231  
232  
233         VUG::J2735::TransmissionState transmissionState;    ///< From BSM Core Data  
234  
235         optional float32 steeringWheelAngle;  ///< Angle of the drivers steering wheel Defined in BSM Core Data.  
236             ///< LSB units of 1.5 degrees, a range of -189 to +189 degrees.  
237             ///< Value of +127 is used for unavailable value. From BSM Core Data  
238         VUG::J2735::BrakeSystemStatus brakeSystemStatus;    ///< From BSM Core Data  
239  
240         const float32 vehicleWidthInMeters;  
241         const float32 vehicleLengthInMeters;  
242         optional const float32 vehicleHeightInMeters;  
243  
244         optional VUG::J2735::VehicleEventFlags vehicleEventFlags;  ///< Indicates potential states of the vehicle based on  
245             ///< J2735 VehicleSafetyExtensions. This attribute should  
246             ///< not be set if all the flags are false.  
247
```

VUG-Vehicle imported OM

[Repository Home](#) \ [Browse Components](#) \ [VUG](#) \ Vehicle (ObjectModel)



VUG-Vehicle-v0.6.0

[Versions](#)

[Download](#)

[State:](#) Released

[Restrictions:](#) Public

[Imports:](#) [TENA-Geospatial-PointOfInterest-v1.0.0](#)

[Create Helpdesk Case](#)

[Middleware:](#) 6.0.9

6.0.8

6.0.7

[TENA-Geospatial-TSPLcovariance-v1.0.0](#)

[TENA-Hardware-System-v1.0.0](#)

(...6 more...)

[Updated:](#)

Superseded By: None

[TENA-Instrumentation-Track-State3D-v1.0.0](#)

Point of Contact: [VUG](#)

Supersedes: None

[TENA-TSPI-v5](#)

[TENA-Time-v2](#)

[TENA-exceptions-v1.0.0](#)

[VUG-Entity-v0.1.0](#)

[VUG-TrafficLight-v0.3.0](#)

```
1 //\file VUG-Vehicle.tdl
2 /**
3 /// This file contains the definition of and documentation for the Vehicle and Track:::BSM object model
4 /// for VOICES. The current content is based off of the SAE J2735 Basic Safety Message and the
5 /// following messages from CARMA:
6 /// https://github.com/usdot-fhwa-stol/carma\_msgs/blob/develop/msg/msgExternalObject.msg
7 /// https://github.com/usdot-fhwa-stol/carma\_msgs/blob/develop/msg/msgVehicleLocation.msg
8 /// http://docs.ros.org/en/jade/api/geometry\_msgs/html/msg/Twist.html
9 /// http://docs.ros.org/en/jade/api/geometry\_msgs/html/msg/Pose.html
10 /**
11 /// The VUG::Entities::ControllableVehicle extends the base VUG::Entities::Vehicle class to add the
12 /// basics for requesting control of a vehicle. Generally this class will be used as a base class
13 /// for a class that is specific to a particular simulation's API, such as the VUG::Entities::CARLA::Vehicle.
14 /**
15 /// The VUG::Entities::CARLA::Vehicle is an extension of the VUG::Entities::ControllableVehicle class and
16 /// uses the CARLA Vehicle class reference:
17 /// http://carla.org/Doxygen/html/d9/dc6/classcarla\_1\_1client\_1\_1Vehicle.html#a278374757fcc52dbb06a84a04c5189c7
18 /// to add attributes specific to CARLA to the Vehicle. The carla::client::Vehicle ApplyControl was used as
```

The VUG-Vehicle OM imports the
[TENA-Instrumentation-Track-State3D-v1.0.0 OM](#),
which is where we will find the definition for the
Track3D class



TENA-Instrumentation-Track-State3D-v1.0.0 OM

[Repository Home](#) \ [Browse Components](#) \ [TENA](#) \ Instrumentation-Track-State3D (ObjectModel)



TENA-Instrumentation-Track-State3D-v1.0.0

Versions

[Download](#) [Upload New Version](#)

[State:](#) Released

[Restrictions:](#) Public

Imports: [TENA-Geospatial-PointOfInterest-v1.0.0](#)

[Create Helpdesk Case](#)

[Middleware:](#) 6.0.9

6.0.8

6.0.7

(...3 more...)

[TENA-Geospatial-TSPlcovariance-v1.0.0](#)

[TENA-Hardware-System-v1.0.0](#)

(...3 more...)

Updated:

11/7/18 9:45 PM

Superseded By: None

Point of Contact: [TENA](#)

[Supersedes:](#) [TENA-Instrumentation-Track-State3D-v1.0.0M1135.7](#)

[TDL](#) | [Documentation](#) | [History](#)

```
1 //> \file TENA-Instrumentation-Track-State3D-v1.0.0.tdl
2 /**
3  * This file contains the definition of and documentation for the TENA::Instrumentation::Track::State3D SDO
4  */
5 /**
6  * \attention
7  * This software was written under US Government contracts 1435-04-01-CT-31085, DASG60-02-D-0006-040,
8  * DASG60-02-D-0006-122, and W900KK-09-C-0013; and the US Government retains unlimited rights to the software.
9
10 import <TENA-Geospatial-PointOfInterest-v1.0.0.tdl>
11 import <TENA-Hardware-System-v1.0.0.tdl>
12
13 package TENA {
14     package Instrumentation {
15         package Track {
16
17             /// Enumeration indicating, at a high level, the nature of the corrections that can be made to the track data
18             enum CorrectionMode {
```

State3D class (SDO)

```
47     /// be useful to provide covariance information (perhaps based on statistical processing from the actual sensor).
48     ///
49     /// For a single measurement dimension, it is common to use twice the estimated standard deviation of the
50     /// measurement to represent the uncertainty. This is formally defined to be the "expanded uncertainty" (i.e.,
51     /// estimated standard deviation with a coverage factor k = 2) and it corresponds to ~95.45% confidence. For
52     /// example, the temperature is 100 °F ± 1 °F, with ~95.45% confidence. Using a coverage factor k = 2 is
53     /// consistent with current international practice (see [Guidelines for Evaluating and Expressing the Uncertainty
54     /// of NIST Measurement Results (NIST Technical Note 1297)]
55     /// (http://physics.nist.gov/Pubs/guidelines/TN1297/tn1297s.pdf)).  

56     ///
57     /// For measurements involving multiple dimensions, it is often desirable to support a covariance matrix
58     /// representation of the uncertainty for sophisticated uses. At the same time, it is desirable to also support a
59     /// simpler (single value expanded uncertainty) representation to accommodate publishers that are not capable of
60     /// providing a more sophisticated covariance matrix representation, as well as to accommodate subscribers that
61     /// require only a "general sense" of the uncertainty of the measurement.  

62     class State3D : extends TENA::Geospatial::PointOfInterest {  

63         const TENA::Hardware::System * system;           ///< A pointer to the particular TENA::Hardware::System that  

64                                         // produced the particular State3D  

65         CorrectionMode correctionMode;                ///< Indicates the nature of any corrections made to the data  

66         optional string nameOfOtherCorrectionMode;    ///< To be set if and only if \ref correctionMode is  

67                                         // TENA::Instrumentation::Track::CorrectionMode_Other  

68         boolean isPositionDirectlyMeasured;          ///< True indicates the position was actually measured, not  

69                                         // computed from other measurements  

70         boolean isVelocityDirectlyMeasured;          ///< True indicates the velocity was actually measured, not  

71                                         // computed from other measurements  

72         boolean isAccelerationDirectlyMeasured;       ///< True indicates the acceleration was actually measured, not  

73                                         // computed from other measurements  

74         boolean isOrientationDirectlyMeasured;        ///< True indicates the orientation was actually measured, not  

75                                         // computed from other measurements  

76         boolean isAngularVelocityDirectlyMeasured;   ///< True indicates the angularVelocity was actually measured, not  

77                                         // computed from other measurements  

78         boolean isAngularAccelerationDirectlyMeasured;///< True indicates the angularAcceleration was actually measured,  

79                                         // not computed from other measurements  

80     };  

81
82     }; // package Track
83     }; // package Instrumentation
84     }; // package TENA
```

The TENA-Instrumentation-Track-State3D SDO
(defined in the TENA-Instrumentation-Track-State3D-v1.0.0 OM), not
“extends” the TENA-Geospatial-PointOfInterest SDO
(defined in the TENA-Geospatial-PointOfInterest-v1.0.0 OM)

Dissecting the BSM SDO – continued

- Continuing with the State3D, we would like to know more about its class structure and methods, including what it inherits
- Since a State3D SDO “is a” PointOfInterest SDO, we will look at PointOfInterest next...



TENA-Instrumentation-Track-State3D imported OM

[Repository Home](#) \ [Browse Components](#) \ [TENA](#) \ Instrumentation-Track-State3D (ObjectModel)



TENA-Instrumentation-Track-State3D-v1.0.0

Versions

[Download](#) [Upload New Version](#)

[State:](#) Released

[Restrictions:](#) Public

Imports

[TENA-Geospatial-PointOfInterest-v1.0.0](#)

[Create Helpdesk Case](#)

[Middleware:](#) 6.0.9

6.0.8

6.0.7

(...3 more...)

Updated: 11/7/18 9:45 PM

Superseded By

[TENA-TSPI-v5](#)

Point of Contact: [TENA](#)

[Supersedes:](#)

[TENA-Time-v2](#)

[TENA-exceptions-v1.0.0](#)

-Track-State3D-v1.0.OM1135.7

[TDL](#) | [Documentation](#) | [History](#)

```
1  /// \file TENA-Instrumentation-Track-State3D.tdl
2  ///
3  /// This file contains the definition of the TENA-Instrumentation-Track-State3D SDO
4  ///
5  /// \attention
6  /// This software was written under US Government contract DASG60-02-D-0006-00,
7  /// DASG60-02-D-0006-122, and W900KK-09-C-0013, and the US Government retains unlimited rights to the software.
8
9
10 import <TENA-Geospatial-PointOfInterest-v1.0.0.tdl>
11 import <TENA-Hardware-System-v1.0.0.tdl>
12
13 package TENA {
14     package Instrumentation {
15         package Track {
16
17             /// Enumeration indicating, at a high level, the nature of the corrections that can be made to the track data
18             enum CorrectionMode {
```

The TENA-Instrumentation-Track-State3D OM imports the
TENA-Geospatial-PointOfInterest-v1.0.0 OM,
which is where we will find the definition for the
PointOfInterest class



TENA-Geospatial-PointOfInterest-v1.0.0 OM

[Repository Home](#) \ [Browse Components](#) \ [TENA](#) \ Geospatial-PointOfInterest (ObjectModel)



TENA-Geospatial-PointOfInterest-v1.0.0

Versions

[Download](#) [Upload New Version](#)

[State:](#) Released

[Restrictions:](#) Public

Imports: [TENA-Geospatial-TSPIcovariance-v1.0.0](#)

[Create Helpdesk Case](#)

[Middleware:](#) 6.0.9

6.0.8

6.0.7

(...3 more...)

Updated: 11/7/18 9:17 PM Superseded By: None

Point of Contact: [TENA](#)

[Supersedes:](#) [TENA-Geospatial-PointOfInterest-v1.0.0.M1135.7](#)

[TDL](#) | [Documentation](#) | [History](#)

```
1  /// \file TENA-Geospatial-PointOfInterest-v1.0.0.tdl
2  ///
3  /// This file contains the definition of and documentation for the TENA::Geospatial::PointOfInterest SDO.
4  ///
5  /// \attention
6  /// This software was written under US Government contracts 1435-04-01-CT-31085, DASG60-02-D-0006-040,
7  /// DASG60-02-D-0006-122, and W900KK-09-C-0013; and the US Government retains unlimited rights to the software.
8
9 import <TENA-Geospatial-TSPIcovariance-v1.0.0.tdl>
10 import <TENA-TSPI-v5.tdl>
11
12 package TENA {
13     package Geospatial {
14         /// SDO that represents a point in time and space
15         ///
16         /// The %PointOfInterest SDO is used to provide information about the time and space position information (%TSPI) of
17         /// something (or someplace) of possible interest. This information can be used, for example, within a range
```



PointOfInterest OM

```
24  /// TENA::Instrumentation::Track::State3D SDO is used to provide actual %TSPI measurement data from a sensor that is
25  /// sensing or measuring the %TSPI of something.
26  ///
27  /// Example Use Case: Several radar systems measure the %TSPI of a single missile and each produces a
28  /// TENA::Instrumentation::Track::State3D SDO instance. An impact prediction system subscribes to the
29  /// TENA::Instrumentation::Track::State3D SDOs and uses that information to produce a %PointOfInterest SDO instance
30  /// that predicts where and when the missile is going to make impact. An optical tracking sensor system subscribes
31  /// to the %PointOfInterest SDO and "looks" at the place where the missile is predicted to impact.
32  class PointOfInterest {
33      const string identifier;                                ///< Used to identify or name the %PointOfInterest, e.g., "Missile 4",
34                                              /// "XYZ-123", "Calibration Target 6", "Projected Point Of Impact", etc.
35      optional string designation;                          ///< Mission-specific identification information, e.g., "Nosecone"
36      const string sourceIdentifier;                      ///< Used to identify the system that is publishing the %PointOfInterest,
37                                              /// e.g., "Radar27", "Acme Fusion System 1", etc.
38      optional const string sourceDescription;           ///< Used to provide information about the source, beyond its identifier,
39                                              /// e.g., "FPS-16 Radar", "CRIIS GPS Pod", "Data Playback System", etc.
40
41      /// Contains the time and space position information (%TSPI) for the point of interest. Note well that the time
42      /// value may be in the future, e.g., to represent a prediction about the TENA::TSPI data for the %PointOfInterest
43      /// at that future time (e.g., a predicted location and time of impact). In other cases, a subscribing system may
44      /// choose to extrapolate the TENA::TSPI data for the %PointOfInterest to make its own prediction.
45      TENA::TSPI tspi;
46
47      optional float32 expandedUncertaintyInSeconds;        ///< E.g., ±0.001 s, with ~95.45% confidence
48      optional float32 expandedUncertaintyInMeters;          ///< E.g., ±1 m, with ~95.45% confidence
49      optional float32 expandedUncertaintyInMetersPerSecond;  ///< E.g., ±1 m/s, with ~95.45% confidence
50      optional float32 expandedUncertaintyInMetersPerSecondSquared;  ///< E.g., ±1 m/s2 with ~95.45% confidence
51      optional float32 expandedUncertaintyInOrientation;    ///< E.g., ±0.1 radians, with ~95.45% confidence
52      optional float32 expandedUncertaintyInAngularVelocity;  ///< E.g., ±0.1 radians/s, with ~95.45% confidence
53      optional float32 expandedUncertaintyInAngularAcceleration;  ///< E.g., ±0.1 radians/s2, with ~95.45% confidence
54
55      /// Detailed covariance information about the measurements. When updated, the appropriate expandedUncertainty
56      /// values should also be updated for subscribers only capable of using those uncertainty representations.
57      optional TENA::Geospatial::TSPIcovariance tspiCovariance;
58  };
59
60  }; // package Geospatial
61 
```

The PointOfInterest class is a base class that does not inherit additional attributes or methods



PointOfInterest SDO

```
class PointOfInterest
{
    const string identifier;
    optional string designation;
    const string sourceIdentifier;
    optional const string sourceDescription;
    TENA::TSPI tspi;
    optional float32 expandedUncertaintyInSeconds;
    optional float32 expandedUncertaintyInMeters;
    optional float32 expandedUncertaintyInMetersPerSecond;
    optional float32 expandedUncertaintyInMetersPerSecondSquared;
    optional float32 expandedUncertaintyInOrientation;
    optional float32 expandedUncertaintyInAngularVelocity;
    optional float32 expandedUncertaintyInAngularAcceleration;
    optional TENA::Geospatial::TSPIcovariance tspiCovariance;
};
```



PointOfInterest SDO

```
// A lot of this is optional stuff...
class PointOfInterest
{
    const string identifier;
    optional string designation;
    const string sourceIdentifier;
    optional const string sourceDescription;
    TENA::TSPI tspi;
    optional float32 expandedUncertaintyInSeconds;
    optional float32 expandedUncertaintyInMeters;
    optional float32 expandedUncertaintyInMetersPerSecond;
    optional float32 expandedUncertaintyInMetersPerSecondSquared;
    optional float32 expandedUncertaintyInOrientation;
    optional float32 expandedUncertaintyInAngularVelocity;
    optional float32 expandedUncertaintyInAngularAcceleration;
    optional TENA::Geospatial::TSPICovariance tspiCovariance;
};
```



PointOfInterest SDO

```
// Without all the optional stuff things look a little simpler...
class PointOfInterest
{
    const string identifier;
    const string sourceIdentifier;
    TENA::TSPI tspi;
};
```



Back to State3D SDO

```
class State3D : extends TENA::Geospatial::PointOfInterest
{
    const TENA::Hardware::System * system;
    CorrectionMode correctionMode;
    optional string nameOfOtherCorrectionMode;
    boolean isPositionDirectlyMeasured;
    boolean isVelocityDirectlyMeasured;
    boolean isAccelerationDirectlyMeasured;
    boolean isOrientationDirectlyMeasured;
    boolean isAngularVelocityDirectlyMeasured;
    boolean isAngularAccelerationDirectlyMeasured;
};
```



State3D SDO - optional attribute

```
// Just one optional attribute...
class State3D : extends TENA::Geospatial::PointOfInterest
{
    const TENA::Hardware::System * system;
    CorrectionMode correctionMode;
    optional string nameOfOtherCorrectionMode;
    boolean isPositionDirectlyMeasured;
    boolean isVelocityDirectlyMeasured;
    boolean isAccelerationDirectlyMeasured;
    boolean isOrientationDirectlyMeasured;
    boolean isAngularVelocityDirectlyMeasured;
    boolean isAngularAccelerationDirectlyMeasured;
};
```



PointOfInterest and State3D

```
// Without the optional stuff
class PointOfInterest
{
    const string identifier;
    const string sourceIdentifier;
    TENA::TSPI tspi;
};

class State3D : extends TENA::Geospatial::PointOfInterest
{
    const TENA::Hardware::System * system;
    CorrectionMode correctionMode;
    boolean isPositionDirectlyMeasured;
    boolean isVelocityDirectlyMeasured;
    boolean isAccelerationDirectlyMeasured;
    boolean isOrientationDirectlyMeasured;
    boolean isAngularVelocityDirectlyMeasured;
    boolean isAngularAccelerationDirectlyMeasured;
};
```



Back to BSM SDO

```
class BSM : extends TENA::Instrumentation::Track::State3D
{
    const LVCindicator lvcIndicator;
    const vector<octet> basicSafetyMessageIdentifier;
    boolean validLatitde;
    boolean validLongitude;
    boolean validElevation;
    int8 messageCount;
    uint16 msWithinMinute;
    VUG::J2735::TransmissionState transmissionState;
    optional float32 steeringWheelAngle;
    VUG::J2735::BrakeSystemStatus brakeSystemStatus;
    const float32 vehicleWidthInMeters;
    const float32 vehicleLengthInMeters;
    optional const float32 vehicleHeightInMeters;
    optional VUG::J2735::VehicleEventFlags vehicleEventFlags;
    optional VUG::J2735::PathHistory pathHistory;
    optional VUG::J2735::PathPrediction pathPrediction;
    optional VUG::J2735::ExteriorLights exteriorLights; };
```

Back to BSM SDO – optional attributes

```
// A lot of this is optional stuff...
class BSM : extends TENA::Instrumentation::Track::State3D
{
    const LVCindicator lvcIndicator;
    const vector<octet> basicSafetyMessageIdentifier;
    boolean validLatitde;
    boolean validLongitude;
    boolean validElevation;
    int8 messageCount;
    uint16 msWithinMinute;
    VUG::J2735::TransmissionState transmissionState;
    optional float32 steeringWheelAngle;
    VUG::J2735::BrakeSystemStatus brakeSystemStatus;
    const float32 vehicleWidthInMeters;
    const float32 vehicleLengthInMeters;
    optional const float32 vehicleHeightInMeters;
    optional VUG::J2735::VehicleEventFlags vehicleEventFlags;
    optional VUG::J2735::PathHistory pathHistory;
    optional VUG::J2735::PathPrediction pathPrediction;
    optional VUG::J2735::ExteriorLights exteriorLights; };
```



Back to BSM SDO – optional attributes

```
// Without the optional stuff...
class BSM : extends TENA::Instrumentation::Track::State3D
{
    const LVCindicator lvcIndicator;
    const vector<octet> basicSafetyMessageIdentifier;
    boolean validLatitde;
    boolean validLongitude;
    boolean validElevation;
    int8 messageCount;
    uint16 msWithinMinute;
    VUG::J2735::TransmissionState transmissionState;
    VUG::J2735::BrakeSystemStatus brakeSystemStatus;
    const float32 vehicleWidthInMeters;
    const float32 vehicleLengthInMeters;
};
```

Dissecting an SDO – attributes and remote methods

- From PointOfInterest
 - `const string identifier`
 - `const string sourceIdentifier`
 - `TENA::TSPI tspi`
 - From State3D, which extends PointOfInterest
 - `const TENA::Hardware::System * system`
 - `CorrectionMode correctionMode`
 - `boolean isPositionDirectlyMeasured`
 - `boolean isVelocityDirectlyMeasured`
 - `boolean isAccelerationDirectlyMeasured`
 - `boolean isOrientationDirectlyMeasured`
 - `boolean isAngularVelocityDirectlyMeasured`
 - `boolean isAngularAccelerationDirectlyMeasured`
 - From BSM, which extends State3D
 - `const LVCIndicator lvclIndicator`
 - `const vector<octet> basicSafetyMessageIdentifier`
 - `boolean validLatitude`
 - `boolean validLongitude`
 - `boolean validElevation`
 - `int8 messageCount`
 - `uint16 msWithinMinute`
 - `VUG::J2735::TransmissionState transmissionState`
 - `VUG::J2735::BrakeSystemStatus brakeSystemStatus`
 - `const float32 vehicleWidthInMeters`
 - `const float32 vehicleLengthInMeters`
- 2 levels of inheritance
 - BSM is a State3D
 - State3D is a PointOfInterest
 - 22 attributes total
 - 7 const
(only set in initializer)
 - 2 string
 - 1 SDO pointer
 - 1 enumeration
 - 1 vector of octets
 - 2 float32
 - 15 variable
(set in initializer and can be updated)
 - 2 local class (tspi and brakeSystemStatus)
 - 2 enumeration
 - 9 boolean
 - 1 int8
 - 1 uint16
 - No remote methods



Dissecting an SDO - reorganized view of BSM

Constant (only set in initializer)

- String
 - identifier
 - sourceIdentifier
- SDO pointer
 - TENA::Hardware::System * system
- Enumeration
 - LVCindicator lvclIndicator
- Vector of octets
 - basicSafetyMessageIdentifier
- 32-bit floating point
 - vehicleWidthInMeters
 - vehicleLengthInMeters

Variable (set in initializer and can be updated)

- Local Class
 - TENA::TSPI tspi
 - VUG::J2735::BrakeSystemStatus brakeSystemStatus
- Enumeration
 - CorrectionMode correctionMode
 - VUG::J2735::TransmissionState transmissionState
- Boolean
 - isPositionDirectlyMeasured
 - isVelocityDirectlyMeasured
 - isAccelerationDirectlyMeasured
 - isOrientationDirectlyMeasured
 - isAngularVelocityDirectlyMeasured
 - isAngularAccelerationDirectlyMeasured
 - validLatitde
 - validLongitude
 - validElevation
- 8-bit signed integer
 - messageCount
- 16-bit unsigned integer
 - msWithinMinute



Creating a BSM SDO in a C++ app

- Project file(s) (e.g., Makefile)
 - Add OM (and OM implementation)
 - include directories
 - libraries
- C++ source code
 - Add include statements for OM containing the SDO
 - Add include statements for any OM LC implementations as well
 - Create an SDO servant factory
 - Create an SDO initializer
 - Use the servant factory and initializer to create an SDO
 - We will postpone discussion/exercises related to providing a remote methods implementation
 - Join the class execution and verify the SDO is displayed on the instructor app

Makefile – add OM/impl include directories

- For each OM that is used, including OM dependencies, add directory include options for compilation, e.g., for VUG-Vehicle-v0.6.0:

```
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/VUG-Vehicle-v0.6.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/VUG-Entity-v0.1.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/VUG-TrafficLight-v0.3.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-Time-v2
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-exceptions-v1.0.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-Hardware-System-v1.0.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-TSPI-v5
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-Geospatial-TSPIcovariance-v1.0.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-Geospatial-PointOfInterest-v1.0.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-Instrumentation-Track-State3D-v1.0.0
```

- For each OM that is used and has local class constructors or methods, add directory include options for compilation, e.g.:

```
-I$(TENA_HOME)/$(TENA_VERSION)/include/impls/TENA-Geospatial-TSPIcovariance-v1.0.0-StdImpl-v1.0.1
-I$(TENA_HOME)/$(TENA_VERSION)/include/impls/TENA-Time-v2-StdImpl-v1.1.4
-I$(TENA_HOME)/$(TENA_VERSION)/include/impls/TENA-TSPI-v5-StdImpl-v1.0.4
-I$(TENA_HOME)/$(TENA_VERSION)/include/impls/TENA-Hardware-System-v1.0.0-StdImpl-v1.0.1
```

- The example apps provide good examples of what is required for each SDO and message*

Makefile – add OM/impl libraries

- For each OM that is used, including OM dependencies, add OM libraries, e.g., for VUG-Vehicle-v0.6.0:

```
-lVUG-Vehicle-v0.6.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lVUG-Entity-v0.1.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lVUG-TrafficLight-v0.3.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Time-v2-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-exceptions-v1.0.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Hardware-System-v1.0.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-TSPI-v5-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Geospatial-TSPICovariance-v1.0.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Geospatial-PointOfInterest-v1.0.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Instrumentation-Track-State3D-v1.0.0-$(TENA_PLATFORM)-v$(TENA_VERSION)
```

- For each OM that is used and has local class constructors or methods, add implementation libraries for compilation, e.g.:

```
-lTENA-Time-v2-StdImpl-v1.1.4-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-TSPI-v5-StdImpl-v1.0.4-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Geospatial-TSPICovariance-v1.0.0-StdImpl-v1.0.1-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Hardware-System-v1.0.0-StdImpl-v1.0.1-$(TENA_PLATFORM)-v$(TENA_VERSION)
```

- The example apps provide good examples of what is required for each SDO and message*



Add include statements for OM containing the SDO

- For any SDOs or Messages that you plan to publish or subscribe to, you will need to include its API library header files, e.g.,
 - `#include <VUG/Track/BSM.h>`



Add include statements for OM LC implementations



- Add include files for implementations of local class methods or constructors
 - Example apps will put these in localClassImpls.h

Local class impls for Vehicle OM

// Provided by TSPI-v5-StdImpl

```
#include <TENA/TSPI/Impl.h>
#include <TENA/Position/Impl.h>
#include <TENA/Velocity/Impl.h>
#include <TENA/Acceleration/Impl.h>
#include <TENA/AngularVelocity/Impl.h>
#include <TENA/AngularAcceleration/Impl.h>
#include <TENA/Orientation/Impl.h>
```

```
#include <TENA/GeodeticSRF/Impl.h>
#include <TENA/GeodeticPosition/Impl.h>
```

```
#include <TENA/GeocentricSRF/Impl.h>
#include <TENA/GeocentricPosition/Impl.h>
#include <TENA/GeocentricVelocity/Impl.h>
#include <TENA/GeocentricAcceleration/Impl.h>
#include <TENA/GeocentricAngularVelocity/Impl.h>
#include <TENA/GeocentricAngularAcceleration/Impl.h>
```

```
#include <TENA/LTPenuSRF/Impl.h>
#include <TENA/LTPenuPosition/Impl.h>
#include <TENA/LTPenuVelocity/Impl.h>
#include <TENA/LTPenuAcceleration/Impl.h>
#include <TENA/LTPenuAngularVelocity/Impl.h>
#include <TENA/LTPenuAngularAcceleration/Impl.h>
```

```
#include <TENA/LSTPsrf/Impl.h>
#include <TENA/LSTPposition/Impl.h>
#include <TENA/LSTPvelocity/Impl.h>
#include <TENA/LSTPacceleration/Impl.h>
```

```
#include <TENA/DISorientation/Impl.h>
#include <TENA/FRDwrtGeocentricBodyFixedZYXorientation/Impl.h>
#include <TENA/FRDwrtLTPenuBodyFixedZYXorientation/Impl.h>
#include <TENA/FRDwrtGeocentricQuaternionOrientation/Impl.h>
#include <TENA/FRDwrtLTPenuQuaternionOrientation/Impl.h>
```

// Provided by Time-v2-StdImpl

```
#include <TENA/Time/Impl.h>
#include <TENA/Date/Impl.h>
#include <TENA/YTDtime/Impl.h>
#include <TENA/UTCtime/Impl.h>
#include <TENA/GPStime/Impl.h>
#include <TENA/UnixTime/Impl.h>
```

// Provided by Geospatial-TSPIcovariance-v1.0.0-StdImpl

```
#include <TENA/Geospatial/TSPIcovariance/Impl.h>
#include <TENA/Geospatial/ConfidenceEllipsoid3D/Impl.h>
#include <TENA/Geospatial/ConfidenceEllipseXY/Impl.h>
```

// Provided by Hardware-System-v1.0.0-StdImpl

```
#include <TENA/Hardware/ControllerToken/Impl.h>
```



TENA sessions

- Before we can publish or subscribe to anything, we need to create one or more sessions (with unique names) in the execution, using the execution pointer

```
TENA::Middleware::SessionPtr session(  
execution->createSession("my session"));
```



TENA communication properties

- TENA communication properties
 - Reliable: all communication performed via TCP
 - TCP: One-to-one communication (uses more bandwidth for multiple subscribers)
 - Retransmits data if packets are lost in transit
 - Best effort:
 - SDOs:
 - Creation and destruction performed via TCP
 - Updates performed via UDP multicast
 - Messages:
 - Messages sent via UDP multicast
 - EM must be configured to allow best effort communication, specifying a range of multicast addresses and a port
 - UDP/multicast: One-to-many communication (uses less bandwidth for multiple subscribers)
 - No retransmits if packets are lost in transit
 - WANs must be configured with special protocols to support multicast traffic
- SDOs must be created by a servant factory that specifies SDO servant communication properties when SDOs are created
- Messages must be sent by a message sender that is configured for either “reliable” or “best effort” communication
 - You can create multiple message senders to support some messages being sent reliably and some using best effort



Create an SDO servant factory



- Using a TENA session, create an SDO servant factory for the BSM SDO

```
VUG::Track::BSM::ServantFactoryPtr bsmServantFactory  
VUG::Track::BSM::ServantFactory::create(session));
```



Create a BSM SDO initializer



- SDO publication state initializers are used to ensure that required (non-optional) attribute values set before publication
- Initializers are generated as a C++ std::unique_ptr by an SDO servant factory

```
unique_ptr<VUG::Track::BSM::PublicationStateInitializer>  
initializer(bsmServantFactory->createInitializer());
```



Set values in the initializer (minimally all non-optional attributes)



- Review the TDL to determine which attributes must be set in the initializer
 - The auto-generated TENA example apps demonstrate the attributes that must be set
 - Recall earlier that we dissected the BSN SDO and determined that there were 22 non-optional attributes



PublicationStateInitializer BSM example

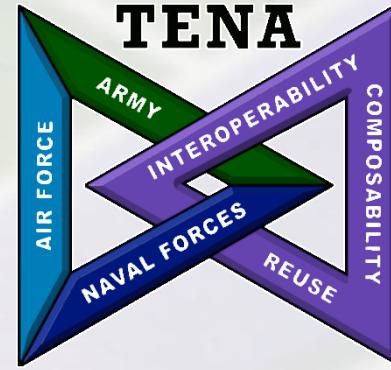
```
initializer->set_identifier("exercise 3");
initializer->set_sourceIdentifier("app1");
initializer->set_tspi(
    TENA::TSPI::create(TENA::Time::create(),
        TENA::Position::create(TENA::GeodeticPosition::create(gdSRF, 40.5, -105.0, 5000.0)));
initializer->set_system(TENA::Hardware::System::SDOpinterPtr());
initializer->set_correctionMode(TENA::Instrumentation::Track::CorrectionMode_None);
initializer->set_isPositionDirectlyMeasured(true);
initializer->set_isVelocityDirectlyMeasured(true);
initializer->set_isAccelerationDirectlyMeasured(true);
initializer->set_isOrientationDirectlyMeasured(true);
initializer->set_isAngularVelocityDirectlyMeasured(true);
initializer->set_isAngularAccelerationDirectlyMeasured(true);
initializer->set_lvcIndicator(VUG::LVCIndicator_Constructive);
initializer->set_basicSafetyMessageIdentifier(vector<TENA::octet>());
initializer->set_validLatitude(true);
initializer->set_validLongitude(true);
initializer->set_validElevation(true);
initializer->set_messageCount(0);
initializer->set_msWithinMinute(0);
initializer->set_transmissionState(VUG::J2735::TransmissionState_Unavailable);
initializer->set_brakeSystemStatus(
    VUG::J2735::BrakeSystemStatus::create(true, true, true, true,
        VUG::J2735::TractionControlStatus_Off, VUG::J2735::AntiLockBrakeStatus_Off,
        VUG::J2735::StabilityControlStatus_Off, VUG::J2735::BrakeBoostApplied_Off,
        VUG::J2735::AuxiliaryBrakeStatus_Off));
initializer->set_vehicleWidthInMeters(5.0);
initializer->set_vehicleLengthInMeters(2.0);
```



Create a servant from the servant factory and initializer

- Now we can use the servant factory to generate a servant from
 - The initializer we just prepared
 - A communication properties option
- When the servant is created, any subscribers that have declared interest in the SDO type will receive a discovery event from the publisher
- Our BSM SDO does not have remote methods
 - *(otherwise, we would also have to provide a remote method implementation at this time)*

```
VUG::Track::BSM::ServantPtr bsmServant =  
bsmServantFactory->createServant(  
*initializer,  
TENA::Middleware::Reliable);
```



Exercise 3

Create an SDO



Exercise 3

- Start from the exercise-03\start folder
 - Review OM (and OM implementation) include directories and libraries that have been added to Makefile
- Add include statements for SDOs that will be published, and for OM implementations
- Create an SDO servant factory
- Create an SDO initializer
- Use the servant factory and initializer to create an SDO
 - Use “reliable” communication properties
- Join the class execution and verify the SDO is displayed on the instructor app



Exercise 3 Ubuntu 20.04, installing VUG-Vehicle OM SDK

- Download the VUG Vehicle OM SDK from the repository and install it

```
sudo dpkg -i --force-overwrite \
VUG-Vehicle-Distribution-v0.6.0@Product@u2004-gcc9-64-v7309e963.deb
```





Exercise 3 app1.cpp add header files

```
#include "localClassImpls.h"  
#include <VUG/Track/BSM.h>
```





Exercise 3 app1.cpp add servant factory



```
VUG::Track::BSM::ServantFactoryPtr bsmServantFactory(  
VUG::Track::BSM::ServantFactory::create(session));
```



Exercise 3 app1.cpp add initializer

```
unique_ptr<VUG::Track::BSM::PublicationStateInitializer>  
initializer(bsmServantFactory->createInitializer());
```



Exercise 3 app1.cpp set initializer attribute values

```
initializer->set_identifier("<lastname>");  
initializer->set_sourceIdentifier("app1");  
initializer->set_tspi(TENA::TSPI::create(TENA::Time::create(),  
TENA::Position::create(TENA::GeodeticPosition::create(  
    TENA::GeodeticSRF::create(), 40.5, -105.0, 5000.0))));  
initializer->set_system(TENA::Hardware::System::SDOPointerPtr());  
initializer->set_correctionMode(TENA::Instrumentation::Track::CorrectionMode_None);  
initializer->set_isPositionDirectlyMeasured(true);  
initializer->set_isVelocityDirectlyMeasured(true);  
initializer->set_isAccelerationDirectlyMeasured(true);  
initializer->set_isOrientationDirectlyMeasured(true);  
initializer->set_isAngularVelocityDirectlyMeasured(true);  
initializer->set_isAngularAccelerationDirectlyMeasured(true);  
initializer->set_lvcIndicator(VUG::LVCindicator_Constructive);  
initializer->set_basicSafetyMessageIdentifier(vector<TENA::octet>());  
initializer->set_validLatitude(true);  
initializer->set_validLongitude(true);  
initializer->set_validElevation(true);  
initializer->set_messageCount(0);  
initializer->set_msWithinMinute(0);  
initializer->set_transmissionState(VUG::J2735::TransmissionState_Unavailable);  
initializer->set_brakeSystemStatus(VUG::J2735::BrakeSystemStatus::create(true, true, true, true,  
    VUG::J2735::TractionControlStatus_Off, VUG::J2735::AntiLockBrakeStatus_Off,  
    VUG::J2735::StabilityControlStatus_Off, VUG::J2735::BrakeBoostApplied_Off,  
    VUG::J2735::AuxiliaryBrakeStatus_Off));  
initializer->set_vehicleWidthInMeters(5.0);  
initializer->set_vehicleLengthInMeters(2.0);
```



Exercise 3 app1.cpp create a servant

```
VUG::Track::BSM::ServantPtr bsmServant =  
bsmServantFactory->createServant(  
*initializer, comProps);
```





Exercise 3 app1.cpp explicitly destroy servant

bsmServant.reset();



Exercise 3 Linux, build and run

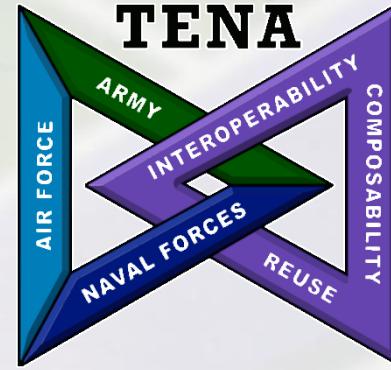
make

```
./app1 -configFile app1.config
```

or

```
./app1 \
    -listenEndpoints 127.0.0.1:55100/portspan=10 \
    -emEndpoints 127.0.0.1:55100
```

verify app joined execution and SDO was created



Lesson 4

Update a published SDO



Lesson 4

- Create an SDO updater using the servant that was created in the previous lesson
- Change state of the updater
 - We'll just update the tspi attribute as an example
- Use SDO servant to commit the updater state



Create an SDO updater using the servant

- Get an updater, which is held in a unique_ptr

```
std::unique_ptr<VUG::Track::BSM::PublicationStateUpdater>  
bsmUpdater(bsmServant->createUpdater());
```

Update attributes in the updater

- Updating TSPI is just an example - we could update other attributes with the updater as well
- We'll just increase the latitude value by 0.001 decimal degrees using namespace **TENA**;

```
// Get the current position
GeodeticPositionPtr gdPosition(
    GeodeticPosition::create(
        GeodeticSRF::create(),
        bsmUpdater->get_tspi()->get_position()));

// Update the latitude value in the position
gdPosition->set_latitudeInDegrees(
    gdPosition->get_latitudeInDegrees() + 0.001);

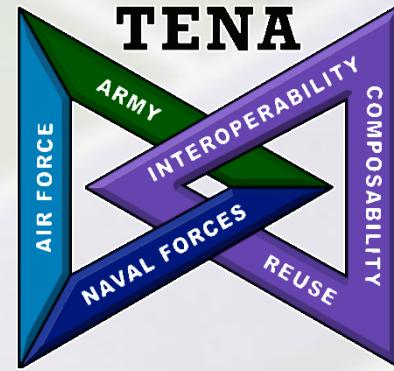
// Set the new position in the updater
bsmUpdater->set_tspi(
    TSPI::create(Time::create(), Position::create(gdPosition)));
```



Use SDO servant to commit the updater state

- Commit the updater to update the servant's state

```
bsmServant->commitUpdater(  
    std::move(bsmUpdater));
```



Exercise 4

Update a published SDO



Exercise 4

- Start from the exercise-04\start folder
- Create an SDO updater using the servant that was created in the previous lesson
- Change state of the updater
 - You can just update the tspi attribute as an example
- Use SDO servant to commit the updater state



Exercise 4 app1.cpp

```
// Re-usable SRF
const TENA::ImmutableGeodeticSRFPtr gdSRF(TENA::GeodeticSRF::create());

// Updater
unique_ptr<VUG::Track::BSM::PublicationStateUpdater> updater(
    bsmServant->createUpdater());

// Get current geodetic position
TENA::GeodeticPositionPtr gdPosition(TENA::GeodeticPosition::create(
    gdSRF, bsmUpdater->get_tspi()->get_position()));

// Update latitude in geodetic position
gdPosition->set_latitudeInDegrees(
    gdPosition->get_latitudeInDegrees() + 0.001);

// Update tspi in updater
bsmUpdater->set_tspi(TENA::TSPI::create(TENA::Time::create(),
    TENA::Position::create(gdPosition)));

// Commit the updater
bsmServant->commitUpdater(move(updater));

// Set identifier to your Lastname for testing exercise 7 later
initializer->set_identifier("<Lastname>");
```



Exercise 4 Linux, build and run

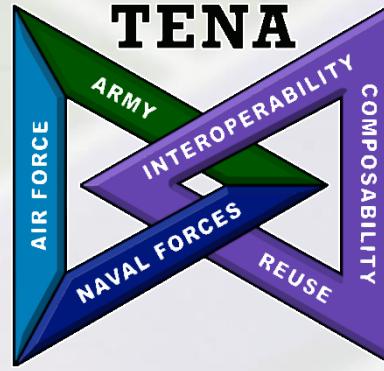
make

```
./app1 -configFile app1.config
```

or

```
./app1 \
    -listenEndpoints 127.0.0.1:55100/portspan=10 \
    -emEndpoints 127.0.0.1:55100
```

```
# verify app joined execution and SDO was created
# and SDO is updating
```



Lesson 5

Subscribing to SDOs



Lesson 5

- Review adding OM (and OM implementation) include directories and libraries to Makefile
- Add include statements for SDO that will be subscribed to (and review adding OM implementation include statements)
- Create a subscription object for the SDO type
- For event-driven behavior:
 - Create SDO observer class(es)
 - Create instances of observer(s)
 - Attach observer(s) to subscription
- Subscribe to SDOs (by type)
- Provide time to the Middleware to evoke subscription callback events
- Access and iterate over the SDO proxy list (for non-event-driven behavior)

Makefile – add OM/impl include directories

- For each OM that is used, including OM dependencies, add directory include options for compilation, e.g., for VUG-Vehicle-v0.6.0:

```
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/VUG-Vehicle-v0.6.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/VUG-Entity-v0.1.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/VUG-TrafficLight-v0.3.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-Time-v2
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-exceptions-v1.0.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-Hardware-System-v1.0.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-TSPI-v5
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-Geospatial-TSPIcovariance-v1.0.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-Geospatial-PointOfInterest-v1.0.0
-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-Instrumentation-Track-State3D-v1.0.0
```

- For each OM that is used and has local class constructors or methods, add directory include options for compilation, e.g.:

```
-I$(TENA_HOME)/$(TENA_VERSION)/include/impls/TENA-Geospatial-TSPIcovariance-v1.0.0-StdImpl-v1.0.1
-I$(TENA_HOME)/$(TENA_VERSION)/include/impls/TENA-Time-v2-StdImpl-v1.1.4
-I$(TENA_HOME)/$(TENA_VERSION)/include/impls/TENA-TSPI-v5-StdImpl-v1.0.4
-I$(TENA_HOME)/$(TENA_VERSION)/include/impls/TENA-Hardware-System-v1.0.0-StdImpl-v1.0.1
```

- The example apps provide good examples of what is required for each SDO and message*

Makefile - add OM/impl libraries

- For each OM that is used, including OM dependencies, add OM libraries, e.g., for VUG-Vehicle-v0.6.0:

```
-lVUG-Vehicle-v0.6.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lVUG-Entity-v0.1.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lVUG-TrafficLight-v0.3.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Time-v2-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-exceptions-v1.0.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Hardware-System-v1.0.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-TSPI-v5-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Geospatial-TSPICovariance-v1.0.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Geospatial-PointOfInterest-v1.0.0-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Instrumentation-Track-State3D-v1.0.0-$(TENA_PLATFORM)-v$(TENA_VERSION)
```

- For each OM that is used and has local class constructors or methods, add implementation libraries for compilation, e.g.:

```
-lTENA-Time-v2-StdImpl-v1.1.4-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-TSPI-v5-StdImpl-v1.0.4-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Geospatial-TSPICovariance-v1.0.0-StdImpl-v1.0.1-$(TENA_PLATFORM)-v$(TENA_VERSION)  
-lTENA-Hardware-System-v1.0.0-StdImpl-v1.0.1-$(TENA_PLATFORM)-v$(TENA_VERSION)
```

- The example apps provide good examples of what is required for each SDO and message*



Add include statements for OM containing the SDO

- For any SDOs or Messages that you plan to publish or subscribe to, you will need to include its API library header files, e.g.,
 - `#include <VUG/Track/BSM.h>`



Add include statements for OM LC implementations

- Add include files for implementations of local class methods or constructors
 - Example apps will put these in localClassImpls.h



Local class impls for Vehicle OM

// Provided by TSPI-v5-StdImpl

```
#include <TENA/TSPI/Impl.h>  
  
#include <TENA/Position/Impl.h>  
#include <TENA/Velocity/Impl.h>  
#include <TENA/Acceleration/Impl.h>  
#include <TENA/AngularVelocity/Impl.h>  
#include <TENA/AngularAcceleration/Impl.h>  
#include <TENA/Orientation/Impl.h>
```

```
#include <TENA/GeodeticSRF/Impl.h>  
#include <TENA/GeodeticPosition/Impl.h>
```

```
#include <TENA/GeocentricSRF/Impl.h>  
#include <TENA/GeocentricPosition/Impl.h>  
#include <TENA/GeocentricVelocity/Impl.h>  
#include <TENA/GeocentricAcceleration/Impl.h>  
#include <TENA/GeocentricAngularVelocity/Impl.h>  
#include <TENA/GeocentricAngularAcceleration/Impl.h>
```

```
#include <TENA/LTPenuSRF/Impl.h>  
#include <TENA/LTPenuPosition/Impl.h>  
#include <TENA/LTPenuVelocity/Impl.h>  
#include <TENA/LTPenuAcceleration/Impl.h>  
#include <TENA/LTPenuAngularVelocity/Impl.h>  
#include <TENA/LTPenuAngularAcceleration/Impl.h>
```

```
#include <TENA/LSTPsrf/Impl.h>  
#include <TENA/LSTPposition/Impl.h>  
#include <TENA/LSTPvelocity/Impl.h>  
#include <TENA/LSTPacceleration/Impl.h>
```

```
#include <TENA/DISorientation/Impl.h>  
#include <TENA/FRDwrtGeocentricBodyFixedZYXorientation/Impl.h>  
#include <TENA/FRDwrtLTPenuBodyFixedZYXorientation/Impl.h>  
#include <TENA/FRDwrtGeocentricQuaternionOrientation/Impl.h>  
#include <TENA/FRDwrtLTPenuQuaternionOrientation/Impl.h>
```

// Provided by Time-v2-StdImpl

```
#include <TENA/Time/Impl.h>  
#include <TENA/Date/Impl.h>  
#include <TENA/YTDtime/Impl.h>  
#include <TENA/UTCtime/Impl.h>  
#include <TENA/GPStime/Impl.h>  
#include <TENA/UnixTime/Impl.h>
```

// Provided by Geospatial-TSPlcovariance-v1.0.0-StdImpl

```
#include <TENA/Geospatial/TSPlcovariance/Impl.h>  
#include <TENA/Geospatial/ConfidenceEllipsoid3D/Impl.h>  
#include <TENA/Geospatial/ConfidenceEllipseXY/Impl.h>
```

// Provided by Hardware-System-v1.0.0-StdImpl

```
#include <TENA/Hardware/ControllerToken/Impl.h>
```



TENA sessions

- Manages publication and subscription state within an app
 - An app may use multiple sessions within an execution
- Before we can publish or subscribe to anything, we need to create one or more sessions (with unique names) in the execution, using the execution pointer

```
TENA::Middleware::SessionPtr session(  
    execution->createSession("my session"));
```

Callback Framework

- Allows the middleware to notify the application to perform processing in response to subscription events
- Callback types
 - SDO related
 - Discovery, State change, and Destruction
 - Entered-scope and left-scope
 - *Notification SDO entered advanced filtering or “best match” interest specification*
 - Message receipt: notification of receiving a message
- Callback evocation methods (return value indicates number of callbacks remaining)
 - `size_t evokeCallback()`
 - Process a single callback if one is pending
 - `size_t evokeCallback(unsigned int waitTimeInMicroseconds)`
 - Process a single callback if one is pending or wait a duration of time for a callback
 - `size_t evokeMultipleCallbacks(unsigned int waitTimeInMicroseconds)`
 - Process any callbacks received within a specified duration of time
 - Most used and most efficient method
 - Ideal for calling in a loop in a subscription thread

Notes on advanced filtering (optional discussion)

- The middleware matches OM type being published with the type of interest for the subscriber
- Advanced filtering extends this type-based subscription to allow publishers and subscribers to provide additional criteria for control of subscription
 - To minimize unwanted information that would be received with a pure type-based filtering system
- Publishers pass a “tag” to the method that creates the SDO or messages
- Subscribers use a “filter” for published SDOs or messages – including wildcards (match all), specific tag values, or no tag (match SDOs/messages that do not have a tag)
- *More info:*
<https://www.trmc.osd.mil/wiki/display/MW/Advanced+Filtering>



Notes on “best-match” subscription (optional discussion)



- Allows apps to subscribe to multiple types within an SDO or Message hierarchy, but only discover the SDO or receive the Message on the “most matching” subscription
- Since subsequent subscriptions can modify the resulting “best match” of an SDO, apps may need to handle object "scope" events
- *More info:*
<https://www.trmc.osd.mil/wiki/display/MW/Best+Match>



State change “pruning”

- “Pruning” prevents the processing of redundant state changes when the app only needs the latest state – not every state change
- Use case examples
 - Data logger that needs to record every state change
 - Set pruning to false
 - Use the publication state passed into the stateChangeEvent method of an “observer” (more on observers later)
 - Display system that just needs the latest state
 - Set pruning to true
 - Obtain the publication state from the proxy during the stateChange method (or after obtaining the proxy list if observers are not used)



Create a subscription object for the SDO type

- Create a BSM subscription that does not prune state changes

```
VUG::Track::BSM::SubscriptionPtr bsmSubscription(  
    std::make_shared<VUG::Track::BSM::Subscription>(false));
```



“Observers” for event-driven subscription

- TENA subscription uses the observer-pattern for event driven subscriptions
 - Used to tell the Middleware what to do when an event occurs
- User-defined class inherits from an abstract observer class to create custom event-driven behavior
- API users
 - Implement observer methods for events (if needed)
 - SDOs events: discovery, state change, destruction, entered scope, & left scope
 - Message event: message received
 - Create instance of observer and attach to a subscription
 - Provide time to the middleware to execute observer methods
- Multiple observers (with same or different behavior) can be attached to the same subscription



Create an SDO observer class

```
class BSMalertingObserver : public VUG::Track::BSM::AbstractObserver {  
public:  
    BSMalertingObserver() {}  
  
    virtual void discoveryEvent(VUG::Track::BSM::ProxyPtr const & proxy,  
        VUG::Track::BSM::PublicationStatePtr const & pubState) {  
        cout << "VUG::Track::BSM discovered" << endl;  
    }  
  
    virtual void stateChangeEvent(VUG::Track::BSM::ProxyPtr const & proxy,  
        VUG::Track::BSM::PublicationStatePtr const & pubState)  
    {  
        cout << "\nVUG::Track::BSM state changed" << endl;  
    }  
  
    virtual void destructionEvent(VUG::Track::BSM::ProxyPtr const & proxy,  
        VUG::Track::BSM::PublicationStatePtr const & pubState) {  
        cout << "\nVUG::Track::BSM destroyed" << endl;  
    }  
  
    BSMalertingObserver & operator=( BSMalertingObserver const & );  
};  
  
typedef TENA::Middleware::Utils::SmartPtr<BSMalertingObserver>  
BSMalertingObserverPtr;
```



Instance(s) of custom observer(s)

- Create an instance of our custom BSMAleringObserver class

```
BSMAleringObserverPtr bsmAlertingObserver(  
    std::make_shared<BSMAleringObserver>()));
```





Attach observer to subscription

- Attach the BSMAlertingObserver to the BSM subscription

```
bsmSubscription->addObserver(bsmAlertingObserver);
```



Self-reflection option

- Controls whether the app receives SDOs & Messages published by the subscribing session
 - (In addition to all other SDOs and messages)
- If true (default, use self reflection): app receives all matching SDOs or Messages, including those published by the subscribing session
- If false (no self-reflection): app receives only matching SDOs & Messages published outside the subscribing session



Subscribe to SDOs



- Declare interest in (subscribe to) BSM SDOs
 - with no self-reflection

**VUG::Track::BSM::subscribe(
session, *bsmSubscription*, false);**



Give time to the Middleware to process callbacks

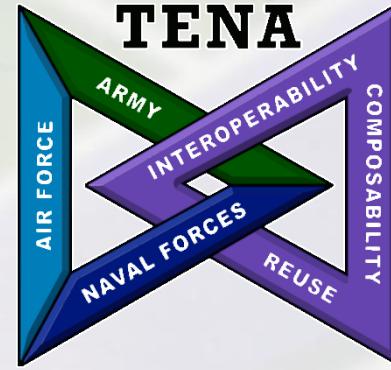
```
unsigned int waitTimeInMicroseconds { 1000000 };  
  
while (!Exiting)  
{  
    session->evokeMultipleCallbacks(waitTimeInMicroseconds);  
}
```



Iterate over the SDO list

- Some apps are less event-driven
 - e.g., apps that may want to periodically poll the state of SDOs to update a display
- Apps can retrieve current SDO list(s) from the corresponding subscription object(s) and iterate over the list(s) if needed

```
VUG::Track::BSM::SDOlist bsmProxies(  
    bsmSubscription->getDiscoveredSDOlist());  
  
size_t count{0};  
  
for(VUG::Track::BSM::ProxyPtr bsmProxy: bsmProxies){  
    VUG::Track::BSM::PublicationStatePtr  
        bsmPubstate = bsmProxy->getPublicationState();  
    std::cout << "BSM " << ++count << ":"  
        << bsmPubstate->get_identifier() << std::endl;  
}
```



Exercise 5

Subscribing to SDOs



Exercise 5

- Start from the exercise-05\start folder
- Review:
 - *Makefile: OM (and OM implementation) include directories and libraries*
 - *localClassImpls.h: LC implementation include statements*
- app2.cpp:
 - Add include statements for SDO that will be subscribed to
 - Create a subscription object for the SDO type
 - Create SDO observer class
 - Create instances of observer
 - Attach observer to subscription
 - Subscribe to SDO type
 - Provide time to the Middleware to evoke subscription callback events
 - Access and iterate over the SDO proxy list (for non-event-driven behavior)
 - Join the class execution and verify you are receiving BSM SDO discoveries and state changes
 - (SDOs will be published from the instructor app)



Exercise 5 app2.cpp, add header files

```
#include <VUG/Track/BSM.h>
#include <TENA/Hardware/System.h>
```





Exercise 5 app2.cpp, add custom observer class

```
class BSMalertingObserver : public VUG::Track::BSM::AbstractObserver {
public:
    BSMalertingObserver()
    : m_gdSRF(TENA::GeodeticSRF::create()), m_gcSRF(TENA::GeocentricSRF::create()) {}

    virtual void discoveryEvent(VUG::Track::BSM::ProxyPtr const & proxy,
        VUG::Track::BSM::PublicationStatePtr const & pubState) {
        cout << "\nVUG::Track::BSM discovered: " << endl;
        printPubState(pubState);
    }

    virtual void stateChangeEvent(VUG::Track::BSM::ProxyPtr const & proxy,
        VUG::Track::BSM::PublicationStatePtr const & pubState) {
        cout << "\nVUG::Track::BSM state changed: " << endl;
        printPubState(pubState);
    }

    virtual void destructionEvent(VUG::Track::BSM::ProxyPtr const & proxy,
        VUG::Track::BSM::PublicationStatePtr const & pubState) {
        cout << "\nVUG::Track::BSM destroyed: " << endl;
        printPubState(pubState);
    }

private:
    TENA::ImmutableGeodeticSRFPtr m_gdSRF;
    TENA::ImmutableGeocentricSRFPtr m_gcSRF;

    void printPubState(VUG::Track::BSM::PublicationStatePtr const & pubState) {
        cout << fixed << " identifier: " << pubState->get_identifier() << endl;
        TENA::GeodeticPositionPtr gdPosition(
            TENA::GeodeticPosition::create(m_gdSRF, pubState->get_tspi()->get_position()));
        cout << "Position [lat, lon, HAET: " << gdPosition->get_latitudeInDegrees() << ", "
            << gdPosition->get_longitudeInDegrees() << ", "
            << gdPosition->get_heightAboveEllipsoidInMeters() << endl;
    }

    BSMalertingObserver & operator=(BSMalertingObserver const &);

};

typedef TENA::Middleware::Utils::SmartPtr<BSMalertingObserver> BSMalertingObserverPtr;
```



Exercise 5 app2.cpp, subscription

```
# Create subscription
VUG::Track::BSM::SubscriptionPtr bsmSubscription(
    make_shared<VUG::Track::BSM::Subscription>(
        false));

# Create instance of observer
BSMAlertingObserverPtr bsmAlertingObserver(
    make_shared<BSMAlertingObserver>());

# Attach observer(s) to subscription
bsmSubscription->addObserver(bsmAlertingObserver);

# Subscribe to BSM SDOs
VUG::Track::BSM::subscribe(
    session, bsmSubscription, false);
```



Exercise 5 app2.cpp, iterate through SDO list

```
while(!exitDetector.exitDetected())
{
    sleep_for(seconds(1));

    # Get SDO list
    VUG::Track::BSM::SDOlist bsmProxies(
        bsmSubscription->getDiscoveredSDOlist());

    cout << "Current BSM identifiers:\n";
    int count{0};
    for(VUG::Track::BSM::ProxyPtr bsmProxy: bsmProxies)
    {
        VUG::Track::BSM::PublicationStatePtr
            bsmPubstate = bsmProxy->getPublicationState();
        cout << "BSM " << ++count << ":" 
            << bsmPubstate->get_identifier() << endl;
    }
}
```



Exercise 5 app2.cpp, unsubscribing from BSM SD0s



```
VUG::Track::BSM::unsubscribe(session);
```



Exercise 5 app2.cpp, giving time to Middleware to evoke callbacks

```
session->evokeMultipleCallbacks(1000000);
```



Exercise 5 Linux, build and run

make

```
./app2 -configFile app2.config
```

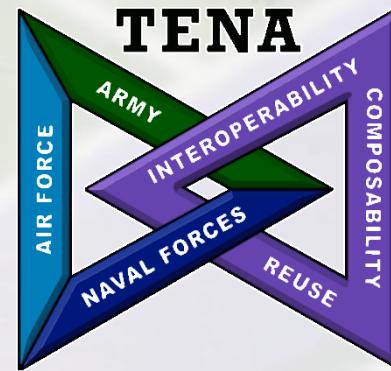
or

```
./app2 \
    -listenEndpoints 127.0.0.1:55100/portspan=10 \
    -emEndpoints 127.0.0.1:55100
```

verify app joined execution

verify SDOs can be received by app2,

by running app1 (SDO publisher)



Lesson 6

Implementing SDO remote methods (from a publisher app)



Lesson 6

- SDO pointers review
- SDO remote methods
 - Review
 - Dissecting a remote method that we will use in exercises
 - Some related topics
 - Oneway remote methods
 - Exceptions
 - Concurrency
- Creating a remote method implementations
 - Accessing/changing SDO state within remote method implementations
- Associating remote method implementations with an SDO servant at the time of its creation



SDO pointers review

- Defined in TDL using an asterisk symbol, like C++ pointers
- Allows apps to refer to SDO instances locally or remotely
- Can be used as
 - Attributes in a class, local class, or message
 - Local or remote method arguments
 - Local or remote method return values
- Can be "dereferenced" in two different ways
 - Creating an instance-based subscription (which provides the client with an SDO Proxy)
 - Apps may obtain SDO pointers as an attribute of another SDO or message, or method return values
 - "Pulling" the associated SDO's publication state.
- Value can be null (i.e., an SDO pointer that does not point to an SDO)
- When attempting to obtain the proxy from an SDO Pointer, there can be several exceptions that may be thrown by the middleware
 - TENA::Middleware::NoSuchSDO: SDO has been destroyed.
 - TENA::Middleware::NetworkError: SDO might have been destroyed, but it could be a network or firewall problem, or the publisher might be hung or stopped in a debugger, etc.
 - std::invalid_argument: SDO is from a different Execution than the SessionPtr that was passed in to subscribe()
 - std::exception: other random failures (e.g. the local Execution has been destroyed and some ManagedPtr threw an InvalidAccess)
- More info: <https://www.trmc.osd.mil/wiki/display/MW/SDO+Pointer>



SDO remote methods review



- Methods declared in a TDL “class” (not local class) specify remote method names, parameters, return values, and special exceptions
 - Methods declared in TDL “local class” are local methods
- Remote method implementations
 - Executed by the publisher host
 - Associated with SDO servants
 - Can be invoked “remotely” from SDO proxies (i.e., subscribers)
 - *Can be invoked “locally” from the SDO servant (i.e., the publisher)*
- Example: VUG-Vehicle-v0.6.0, class *ControllableVehicle*, *setBSMsdoPointer()* (line 282)
- More info:
<https://www.trmc.osd.mil/wiki/display/MW/SDO+Remote+Method>



VUG-Vehicle-v0.6.0, class ControllableVehicle, setBSMsdoPointer()

```
class ControllableVehicle : extends VUG::Entities::Vehicle
{
    TENA::Hardware::ControllerTokenMode tokenMode;
    TENA::Hardware::ControllerInterface * approvedController;

    void setBSMsdoPointer(
        in VUG::Track::BSM * bsmTrackSDOpointer,
        in TENA::Hardware::ControllerToken token);

    void requestControl(
        in TENA::Hardware::ControllerToken token)
        raises(
            TENA::Hardware::InvalidControllerToken,
            TENA::Hardware::RequestRefused);

    void relinquishControl(
        in TENA::Hardware::ControllerToken token)
        raises(
            TENA::Hardware::InvalidControllerToken,
            TENA::Hardware::TokenNotInControl);
};
```



Dissecting a remote method

- Remote method
 - Name: setBSMsdoPointer
 - Return type: void
 - Input parameters
 - VUG::Track::BSM **SDO pointer**
 - TENA::Hardware::ControllerToken
 - (*we'll ignore this in our exercises*)
 - Not a “oneway” remote method
 - Does not declare use of any custom exceptions
- We'll review a few related topics...
 - Oneway remote methods
 - SDO pointers
 - Remote method exceptions



Oneway remote methods

- Indicates that remote method calls do not “block”
 - i.e., they do not wait for the method to return a value
- The invoker will not be informed if the remote method fails due to a communication fault or other errors
 - Unreliable since the invoking application is unable to determine if the invocation was successful
- OM designers should only use oneway methods when blocking behavior is intolerable and apps can tolerate the unreliable nature of these invocations
- This is not the same as a remote method that returns “void” (without oneway specified) – void without oneway is still a blocking call that may return with or without an exception
- More info:
<https://www.trmc.osd.mil/wiki/display/MW/Oneway+Operation+Qualifier>



Remote method exception meta-model construct

- Allows SDO remote method implementor to throw an exception that can be caught and handled by the invoker
- Allows SDO remote method invoker to be informed of issue and take appropriate action
- Can have any fundamental types, such as strings and enums, as attributes
 - Other meta-model constructs such as classes, class pointers, local classes, messages, and vectors cannot be attributes
- Remote method exceptions do not support inheritance
- More info:
<https://www.trmc.osd.mil/wiki/display/MW/Remote+Method+Exception+Meta-Model+Construct>
- Example:

```
exception MyException
{
    string explanation;
};

...
void myMethod() raises (MyException, MyOtherException);
```



Remote method concurrency



- Remote methods
 - Are executed in a middleware thread
 - Can be entered concurrently by multiple middleware threads
- Remote method implementation code must be written in a thread-safe manner

Creating a remote method implementations

- When an app creates an SDO servant with remote methods, the app must implement the remote methods
- A remote methods implementation class needs to inherit from the SDO remote method interface and implement the remote methods

Remote method implementation example

```
using namespace VUG::Entities;
using namespace VUG::Track;

class ControlableVehicleRemoteMethods
    : public virtual ControlableVehicle::RemoteMethodsInterface,
      TENA::Middleware::Utils::noncopyable
{
public:
    ControlableVehicleRemoteMethods(){}
    virtual ~ControlableVehicleRemoteMethods(){}

    virtual void setBSMsdoPointer(
        BSM::SDOPointerPtr const & bsmTrackSDOPointer,
        TENA::Hardware::ImmutableControllerTokenPtr const & token)
    {
        ControllableVehicle::ServantPublicationStateViewPtr cvStateView(
            this->getServantPublicationStateView());
        std::unique_ptr<ControllableVehicle::PublicationStateUpdater>
            cvUpdater(cvStateView->createUpdater());
        cvUpdater->set_bsmTrackSDOPointer(bsmTrackSDOPointer);
        cvStateView->commitUpdater(std::move(cvUpdater));
        return;
    }
    // ... (implement other remote methods)
};
```



Create instance of ControllableVehicleRemoteMethods



```
using namespace std;
using namespace VUG::Entities;

unique_ptr<ControllableVehicle::RemoteMethodsInterface>
controllableVehicleRemoteMethods(
    move(make_unique<ControllableVehicleRemoteMethods>()));
```

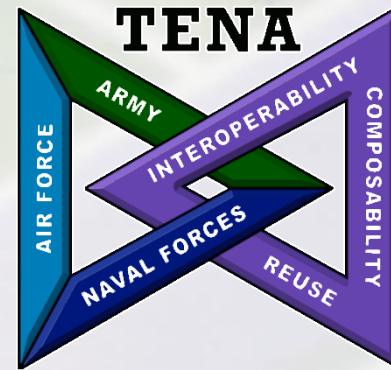


Create a servant with the remote methods implementation



```
using namespace std;
```

```
VUG::Entities::ControllableVehicle::ServantPtr  
cvServant = cvServantFactory->createServant(  
    move(controlLableVehicleRemoteMethods),  
    *initializer, TENA::Middleware::Reliable);
```



Exercise 6

Implementing SDO remote methods (from a publisher app)



Exercise 6

- Start from the exercise-06\start folder
- Create a remote method implementation class and instance for the ControllableVehicle SDO
- Associate the remote method implementation with an SDO at the time of its creation
- Join the class execution and verify the SDOs remote method is invoked from the instructor app



Exercise 6, custom remote methods implementation

```
class ControllableVehicleRemoteMethods
: public virtual VUG::Entities::ControllableVehicle::RemoteMethodsInterface,
  TENA::Middleware::Utils::noncopyable {
public:
    ControllableVehicleRemoteMethods(){}
    virtual ~ControllableVehicleRemoteMethods(){}
    virtual void setBSMsdoPointer(
        VUG::Track::BSM::SDOPointerPtr const & bsmTrackSDOPointer,
        TENA::Hardware::ControllerToken::ImmutableLocalClassPtr const & token)
    {
        VUG::Entities::ControllableVehicle::ServantPublicationStateViewPtr
            stateView(this->getServantPublicationStateView());
        unique_ptr<VUG::Entities::ControllableVehicle::PublicationStateUpdater>
            updater(stateView->createUpdater());
        cvUpdater->set_bsmTrackSDOPointer(bsmTrackSDOPointer);
        stateView->commitUpdater(move(updater));
        cout << "setBSMsdoPointer(): bsmTrackSDOPointer set via RMI" << endl;
    }
    virtual void requestControl(
        TENA::Hardware::ControllerToken::ImmutableLocalClassPtr const & token) {}
    virtual void relinquishControl(
        TENA::Hardware::ControllerToken::ImmutableLocalClassPtr const & token) {}
};
```



Exercise 6, create servant with remote methods

```
# return servant pointer from custom function createControllableVehicleservant()

unique_ptr<VUG::Entities::ControllableVehicle::RemoteMethodsInterface>
controllableVehicleRemoteMethods(
    move(make_unique<ControllableVehicleRemoteMethods>()));

return cvServantFactory->createServant(
    move(controllableVehicleRemoteMethods), *initializer, comProps);

// Set identifier to your Lastname for testing exercise 7 later
initializer->set_identifier("<Lastname>");
```



Exercise 6 Linux, build and run

make

```
./app3 -configFile app3.config
```

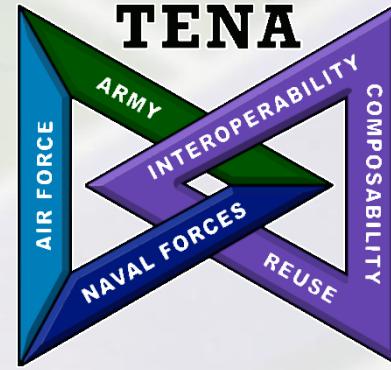
or

```
./app3 \
    -listenEndpoints 127.0.0.1:55100/portspan=10 \
    -emEndpoints 127.0.0.1:55100
```

verify app joined execution

verify SDOs are updating

verify remote methods can be called



Lesson 7

Invoking (calling) SDO remote methods (from a subscribing app)



Lesson 7

- From an SDO proxy, call one of its remote methods
- Calling a remote method from a proxy is straightforward
 - Obtain an SDO proxy
 - Prepare any parameters that need to be passed into the remote method
 - Call the remote method with appropriate input parameters
 - If the remote method is not oneway
 - Wait for the remote method to complete its operation (remote method is a blocking call)
 - If the return type is not void, get the return value
 - It is generally a good idea to wrap remote method calls in try/catch blocks to handle potential exceptions appropriately
 - Review TDL for custom remote method exceptions that may be thrown
 - Review TENA Middleware exceptions that might be thrown (more info: <https://www.trmc.osd.mil/wiki/display/MW/Runtime+Exception+Handling>)



Recall the ControllableVehicle SDO remote method

```
void setBSMsdoPointer(  
    in VUG::Track::BSM * bsmTrackSDOpointer,  
    in TENA::Hardware::ControllerToken token);
```

- In our exercise we will focus on the VUG-Track-BSM SDO pointer and ignore the TENA-Hardware-ControllerToken
 - We can discuss the purpose of TENA-Hardware-ControllerToken if there is interest (and/or refer to the TDL documentation)
- As a subscriber with a ControllableVehicle SDO proxy, we can invoke its setBSMsdoPointer() remote method – but first we will need a BSM SDO pointer to pass to it
- In our exercise, we will subscribe to both ControllableVehicle and BSM, iterate over their SDO proxy lists and look for where their identifier attributes match
 - If we find a match, we will call setBSMsdoPointer() on the ControllableVehicle proxy, passing in an SDO pointer obtained from the BSM proxy



Getting the SDO lists



```
// Get the BSM SDO List
VUG::Track::BSM::SDOlist
    bsmProxies(  

        bsmSubscription->getDiscoveredSDOlist());
  

// Get the ControllableVehicle SDO List
VUG::Entities::ControllableVehicle::SDOlist
    cvProxies(  

        cvSubscription->getDiscoveredSDOlist());
```

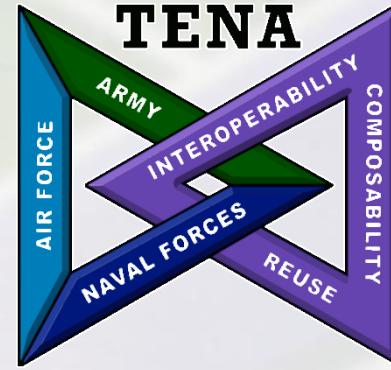
Iterating over the two lists – looking for an identifier match

```
using namespace VUG::Entities;
using namespace VUG::Track;

for(ControllableVehicle::ProxyPtr cvProxy: cvProxies)
{
    ControllableVehicle::PublicationStatePtr
        cvPS = cvProxy->getPublicationState();

    for(BSM::ProxyPtr bsmProxy: bsmProxies)
    {
        BSM::PublicationStatePtr
            bsmPS = bsmProxy->getPublicationState();

        if(bsmPS->get_identifier() == cvPS->get_identifier())
        {
            cvProxy->setBSMsdoPointer(
                bsmProxy->getSDOpointer(),
                TENA::Hardware::ControllerToken::create());
        }
    }
}
```



Exercise 7

Invoking (calling) SDO remote methods (from a subscribing app)

Exercise 7

- Start from the exercise-07\start folder
- From the ControllableVehicle SDO proxy with a matched BSM SDO proxy (same identifier attribute value), call the ControllableVehicle SDO setBSMsdoPointer() remote method
 - Get BSM SDO pointer from a BSM SDO proxy
 - Call the setBSMsdoPointer() remote method passing in parameters for BSM SDO proxy and TENA-Hardware-ControllerToken (using default constructor)
- Join the class execution and verify the instructor app SDO's remote method is called



Exercise 7 app2.cpp call a remote method

```
cvProxy->setBSMsdoPointer(  
    bsmProxy->getSDOpointer(),  
    TENA::Hardware::ControllerToken::create());
```



Exercise 7 Linux, build and run

make

```
./app2 -configFile app2.config
```

or

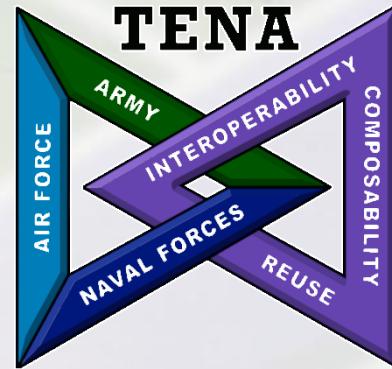
```
./app2 \
    -listenEndpoints 127.0.0.1:55100/portspan=10 \
    -emEndpoints 127.0.0.1:55100
```

verify app joined execution

verify SDOs are being received

verify remote methods are being called

on matching BSM and CV identifiers



Lesson 8

Sending TENA messages



Lesson 8

- Review
 - Adding OM (and OM implementation) include directories and libraries to Makefile
- Add include statements for message type that will be published, and for OM implementations
- Create a message sender (reliable or best effort)
- Create message(s)
- Use message sender to send message(s)



Review – sending messages

- Create a message sender
 - Specifying communication properties and optional filter tags
 - Only one needed per message type
- Create a message
 - Setting attributes as needed
- Use the message sender to send the message
 - Subscribers receive notifications for messages through type subscription



Example - adding OM include directories to Makefile

- `-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/VUG-TrafficControl-v0.4.0`
- `-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-TSPI-v5`
- `-I$(TENA_HOME)/$(TENA_VERSION)/include/OMs/TENA-Time-v2`





Example – adding OM implementation include directories to Makefile



- `-I$(TENA_HOME)/$(TENA_VERSION)/include/impls/TENA-TSPI-v5-StdImpl-v1.0.4`
- `-I$(TENA_HOME)/$(TENA_VERSION)/include/impls/TENA-Time-v2-StdImpl-v1.1.4`



Example - adding OM libraries to Makefile

- `-lVUG-TrafficControl-v0.4.0-$(TENA_PLATFORM)-v$(TENA_VERSION)`
- `-lTENA-TSPI-v5-$(TENA_PLATFORM)-v$(TENA_VERSION)`
- `-lTENA-Time-v2-$(TENA_PLATFORM)-v$(TENA_VERSION)`



Example – adding OM implementation libraries to Makefile

- `-LTENA-TSPI-v5-StdImpl-v1.0.4-$(TENA_PLATFORM)-v$(TENA_VERSION)`
- `-LTENA-Time-v2-StdImpl-v1.1.4-$(TENA_PLATFORM)-v$(TENA_VERSION)`





Example - add include statements for message type that will be published

- `#include <VUG/TrafficControl/TrafficControlRequest.h>`

Example – add include statements for OM implementations

// Provided by TSPI-v5-StdImpl

```
#include <TENA/TSPI/Impl.h>
#include <TENA/Position/Impl.h>
#include <TENA/Velocity/Impl.h>
#include <TENA/Acceleration/Impl.h>
#include <TENA/AngularVelocity/Impl.h>
#include <TENA/AngularAcceleration/Impl.h>
#include <TENA/Orientation/Impl.h>
#include <TENA/GeodeticSRF/Impl.h>
#include <TENA/GeodeticPosition/Impl.h>
#include <TENA/GeocentricSRF/Impl.h>
#include <TENA/GeocentricPosition/Impl.h>
#include <TENA/GeocentricVelocity/Impl.h>
#include <TENA/GeocentricAcceleration/Impl.h>
#include <TENA/GeocentricAngularVelocity/Impl.h>
#include <TENA/GeocentricAngularAcceleration/Impl.h>
#include <TENA/LTPenuSRF/Impl.h>
#include <TENA/LTPenuPosition/Impl.h>
#include <TENA/LTPenuVelocity/Impl.h>
#include <TENA/LTPenuAcceleration/Impl.h>
#include <TENA/LTPenuAngularVelocity/Impl.h>
#include <TENA/LTPenuAngularAcceleration/Impl.h>
#include <TENA/LSTPsrf/Impl.h>
#include <TENA/LSTPposition/Impl.h>
#include <TENA/LSTPvelocity/Impl.h>
#include <TENA/LSTPacceleration/Impl.h>
#include <TENA/DISorientation/Impl.h>
#include <TENA/FRDwrtGeocentricBodyFixedZYXorientation/Impl.h>
#include <TENA/FRDwrtLTPenuBodyFixedZYXorientation/Impl.h>
#include <TENA/FRDwrtGeocentricQuaternionOrientation/Impl.h>
#include <TENA/FRDwrtLTPenuQuaternionOrientation/Impl.h>
```

// Provided by Time-v2-StdImpl

```
#include <TENA/Time/Impl.h>
#include <TENA/Date/Impl.h>
#include <TENA/YTDtime/Impl.h>
#include <TENA/UTCtime/Impl.h>
#include <TENA/GPStime/Impl.h>
#include <TENA/UnixTime/Impl.h>
```



Example – create a message sender

```
using namespace VUG::TrafficControl;  
TrafficControlRequest::MessageSenderPtr tcrMessageSender =  
    TrafficControlRequest::MessageSender::create(  
        session, TENA::Middleware::Reliable);
```





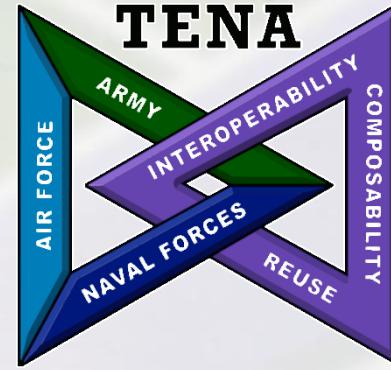
Example – create a message

```
using namespace VUG::TrafficControl;  
  
const TENA::ImmutableGeodeticSRFPtr gdSRF(TENA::GeodeticSRF::create());  
  
// Create a vector of offsets  
vector<ImmutableOffsetPointPtr> offsets;  
  
// Add some offsets  
offsets.push_back(OffsetPoint::create(0, 0));  
offsets.push_back(OffsetPoint::create(0, 100.0));  
offsets.push_back(OffsetPoint::create(100.0, 100.0));  
offsets.push_back(OffsetPoint::create(100.0, 0.0));  
offsets.push_back(OffsetPoint::create(0, 0));  
  
// Create a bounds local class  
BoundsPtr bounds(  
    Bounds::create(TENA::Time::create(),  
    TENA::Position::create(TENA::GeodeticPosition::create(gdSRF, 40.58, -105.08, 5000.8)),  
    offsets));  
  
// Create a vector of bounds  
std::vector<ImmutableBoundsPtr> boundsVector;  
  
// Add 1 bounds to the bounds vector  
boundsVector.push_back(bounds);  
  
// Create a message  
TrafficControlRequest::MessagePtr tcrMessage(  
    TrafficControlRequest::create(  
        /*requestID */"Solution8-Request",  
        /*requestSequence */1,  
        /*vertexOffsetScale */1.0,  
        boundsVector));
```



Example - send a message

```
tcrMessageSender->send(tcrMessage);
```



Exercise 8

Sending TENA messages



Exercise 8

- Download and install the *TrafficControl OM*
- Start from the exercise-08\start folder
- Add OM (and OM implementation) include directories and libraries to Makefile
- Add include statements for message type that will be published, and for OM implementations
- Create a message sender (reliable or best effort)
- Send message(s)
- Join the class execution and verify the message is displayed on the instructor app



Exercise 8 Ubuntu 20.04, installing VUG-TrafficControl OM SDK



- Download the VUG TrafficControl OM SDK from the repository and install it

```
sudo dpkg -i --force-overwrite \
VUG-TrafficControl-Distribution-v0.4.0@Product@u2004-gcc9-64-va498b5f9.deb
```



Exercise 8 app4.cpp, include OM/message header

```
#include <VUG/TrafficControl/TrafficControlRequest.h>
```



Exercise 8 app4.cpp, create a message sender

```
using namespace VUG::TrafficControl;
```

```
TrafficControlRequest::MessageSenderPtr  
tcrMessageSender =  
TrafficControlRequest::MessageSender::create(  
session, comProps);
```



Exercise 8 app4.cpp, create a message

```
using namespace VUG::TrafficControl;
const TENA::ImmutableGeodeticSRFPtr
gdSRF(TENA::GeodeticSRF::create());

vector<ImmutableOffsetPointPtr> offsets;
offsets.push_back(OffsetPoint::create(0, 0));
offsets.push_back(OffsetPoint::create(0, 100.0));
offsets.push_back(OffsetPoint::create(100.0, 100.0));
offsets.push_back(OffsetPoint::create(100.0, 0.0));
offsets.push_back(OffsetPoint::create(0, 0));

BoundsPtr bounds(Bounds::create(TENA::Time::create(),
    TENA::Position::create(TENA::GeodeticPosition::create(
        gdSRF, 40.58, -105.08, 5000.8)), offsets));
vector<ImmutableBoundsPtr> boundsVector;
boundsVector.push_back(bounds);

TrafficControlRequest::MessagePtr tcrMessage(
    TrafficControlRequest::create(
        "exercise8", 1, 1.0, boundsVector));
```



Exercise 8 app4.cpp, send the message

tcrMessageSender->send(tcrMessage);



Exercise 8 Linux, build and run

make

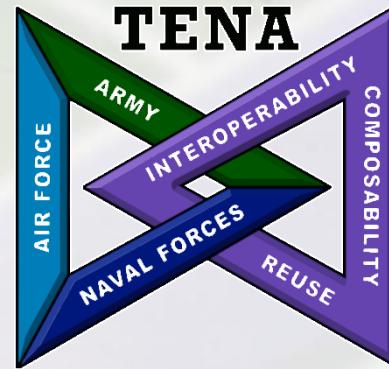
```
./app4 -configFile app4.config
```

or

```
./app4 \
    -listenEndpoints 127.0.0.1:55100/portspan=10 \
    -emEndpoints 127.0.0.1:55100
```

verify app joined execution

verify messages are being sent



Lesson 9

Subscribing to TENA messages



Lesson 9

- Create a message observer class and observer instance
- Create a subscription object
- Attach observer(s) to subscription
- Subscribe to messages
- Provide time in a thread to the Middleware to evoke subscription callback events
- Join the class execution and verify the messages are displayed from the instructor app



Review - subscribing to SDOs & Messages – review

- Create SDO and/or Message subscription objects
 - For messages and SDO event-driven behavior:
 - Implement one or more “observers” for the SDO or message type
 - Attach observer(s) to the subscription
- Subscribe to SDO or Message type using session & subscription object
 - SDOs can be also subscribed to by instance (requires an SDO pointer as an attribute in another SDO or message)
- Provide time to the Middleware to run observer methods as “events” occur
 - Message events: message received
 - SDO events: discovery, state change, entered scope, left scope, destruction
- For SDO non-event-driven behavior:
 - Get the SDO proxy list from the subscription
 - Process SDO proxy state as needed (e.g. iterate through list, print state, etc.)



Review – observers

- Inherit from an abstract observer to create custom observer
- Implement observer methods for events (if needed)
 - SDOs events: discovery, state change, destruction, entered scope, & left scope
 - Message event: message received
- Create instance of observer and attach to subscription
- Observer methods are called by the Middleware as events occur
 - App must provide time to the Middleware using a callback function
- Multiple observers (with same or different behavior) can be attached to the same subscription



Review – evoking callbacks

- Inform Middleware that pending callbacks can be executed
 - Resulting in operations defined by any observers
- Variations of callback evocation
 - `evokeCallback()`
 - `evokeCallback(unsigned int waitTimeInMicroseconds)`
 - `evokeMultipleCallbacks(unsigned int waitTimeInMicroseconds)`
- The return value (a `size_t`) indicates the number of callbacks remaining on the queue

Review – variations of callback evocation

- `evokeMultipleCallbacks(`
`unsigned int waitTimeInMicroseconds)`
 - Execute as many callbacks as possible in the given duration
 - Most used variation
- `evokeCallback()`
 - Execute a single callback (if one exists), then return immediately
- `evokeCallback(`
`unsigned int waitTimeInMicroseconds)`
 - Wait for a single callback (if one exists), then return after timeout or
callback completion



Example – create a message observer class

```
using namespace VUG::TrafficControl;

class TCRAlertingObserver
: public virtual TrafficControlRequest::AbstractObserver
{
public:
    TCRAlertingObserver(){}
    void messageEvent(
        TrafficControlRequest::ReceivedMessagePtr const & receivedMessage)
    {
        std::cout << "TrafficControlRequest requestID: "
            << receivedMessage->get_requestID() << std::endl;
    }
    ...
};

typedef TENA::Middleware::Utils::SmartPtr<TCRAlertingObserver>
    TCRAlertingObserverPtr;
```



Example – create message observer instance

```
// Create an instance of our custom TCRAlertingObserver class  
  
TCRAlertingObserverPtr tcrAlertingObserver(  
    std::make_shared<TCRAlertingObserver>());
```



Example – create a subscription object

```
using namespace VUG::TrafficControl;  
  
TrafficControlRequest::SubscriptionPtr tcrSubscription(  
    std::make_shared<TrafficControlRequest::Subscription>());
```





Example - add observer to subscription

```
// Attach the TCRAlertingObserver to the TrafficControlRequest  
  
subscriptiontcrSubscription->addObserver(  
    tcrAlertingObserver);
```





Example – subscribe to message type

// Subscribe to *TrafficControlRequest* messages, with no self-reflection

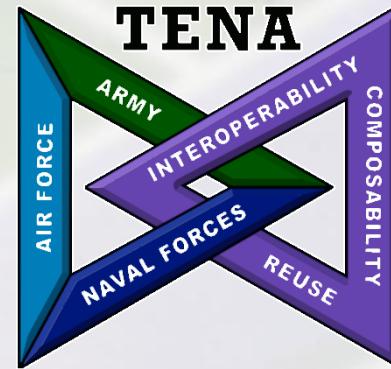
```
VUG::TrafficControl::TrafficControlRequest::subscribe(  
    session, tcrSubscription, false);
```





Example – give time to the Middleware to process callbacks

```
while (!exiting)
{
    session->evokeMultipleCallbacks(1000000);
}
```



Exercise 9

Subscribing to TENA messages



Exercise 9

- Create a message observer class
- Create an instance of the observer class
- Create a subscription object
- Attach observer to subscription
- Subscribe to messages
- Provide time in a thread to the Middleware to evoke subscription callback events
- Join the class execution and verify the messages are displayed from the instructor app



Exercise 9 app5.cpp, OM/message header file

```
#include <VUG/TrafficControl/TrafficControlRequest.h>
```



Exercise 9 app5.cpp, custom observer

```
using namespace VUG::TrafficControl;

class TCRalertingObserver
    : public virtual TrafficControlRequest::AbstractObserver
{
public:
    TCRalertingObserver(){}
    void messageEvent(
        TrafficControlRequest::ReceivedMessagePtr const receivedMessage)
    {
        cout << "TrafficControlRequest received, requestID: "
            << receivedMessage->get_requestID() << endl;
    }
private:
    TCRalertingObserver & operator=(TCRalertingObserver const &);
};

typedef TENA::Middleware::Utils::SmartPtr<TCRalertingObserver>
TCRalertingObserverPtr;
```



Exercise 9 app5.cpp, subscribing to messages

```
using namespace VUG::TrafficControl;

# Create a subscription pointer
TrafficControlRequest::SubscriptionPtr tcrSubscription(
    make_shared<VUG::TrafficControl::TrafficControlRequest::Subscription>());

# Create an instance of our custom observer
TCRalertingObserverPtr tcrAlertingObserver(make_shared<TCRalertingObserver>());

# Add observer to subscription
tcrSubscription->addObserver(tcrAlertingObserver);

# Subscribe to messages
VUG::TrafficControl::TrafficControlRequest::subscribe(
    session, tcrSubscription, false);
```



Exercise 9 app5.cpp, unsubscribing to messages

```
VUG::TrafficControl::TrafficControlRequest::unsubscribe(  
    session);
```



Exercise 9 app4.cpp, giving time to Middleware to evoke callbacks

```
session->evokeMultipleCallbacks(1000000);
```



Exercise 9 Linux, build and run

make

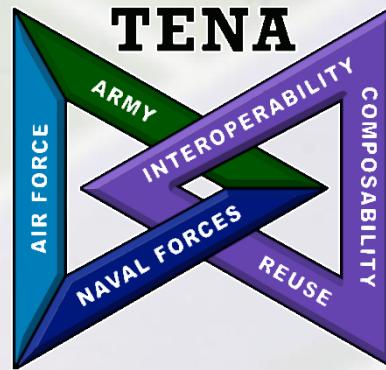
```
./app5 -configFile app5.config
```

or

```
./app5 \
    -listenEndpoints 127.0.0.1:55100/portspan=10 \
    -emEndpoints 127.0.0.1:55100
```

verify app joined execution

verify messages are being received



Additional topics

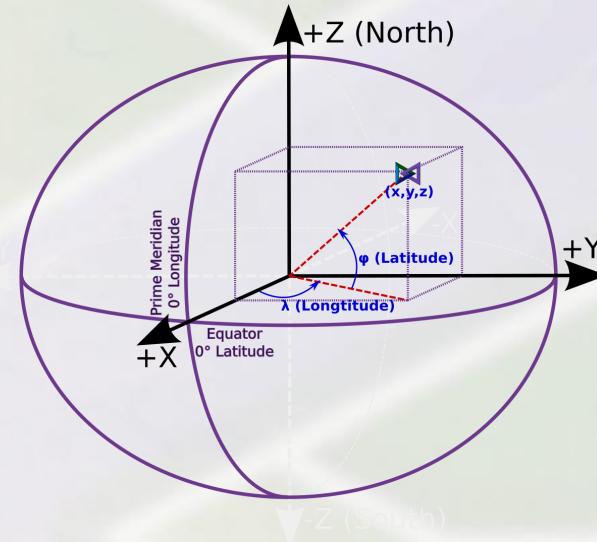
A closer look at the TSPI OM

Simulation time with TENA

OM design guidance

Using cmake with TENA

VS Code for TENA software development



A closer look at the TENA TSPI OM



Acronyms

- TENA: Test & Training Enabling Architecture
- TDL: TENA Definition Language
- OM: Object Model
- LC: Local Class
- TSPI: Time-Space-Position Information
- SDK: Software Development Kit
- API: Application Programming Interface
- GPS: Global Positioning System
- DIS: Distributed Interactive Simulation
- HAE: Height Above Ellipsoid
- ECEF: Earth-Centered, Earth-Fixed
- RAE: Range, Azimuth, and Elevation
- UTC: Coordinated Universal Time
- YTD: Year-To-Date
- SRF: Spatial Reference Frame
- ORM: Object Reference Model
- RT: Reference Transformation
- WGS: World Geodetic System
- FRD: Front-Right-Down
- WRT: With Respect To
- LTP: Local Tangent Plane
- ENU: East-North-Up
- LSTP: Local Spherical Tangent Plane
- POI: Point of Interest



Agenda

- TENA-TSPI-v5 overview
 - Reviewing these slides and some related web pages (e.g., the TSPI OM in the repository)
- TENA-TSPI-v5 demonstrations
 - Running apps that illustrate use of the TSPI OM



TSPI LC design

- The TSPI LC is defined in the TENA-TSPI-v5 OM
 - Describes the structure and interface for geospatial data and related dynamics data (in a variety of coordinate systems) to be exchanged between systems
- Requires a time and position attribute
- Optionally can have orientation, velocity, acceleration, angular velocity, and angular acceleration attributes
- Uses “holder” attributes, which contain a “more specific type”
 - e.g., a position might hold a geocentric position or a geodetic position
- An OM implementation is provided (included with the TENA SDK)
 - Helps enforce how the TSPI OM is used
 - Provides capabilities to convert between specific data types (e.g., to convert between geocentric and geodetic positions)



What is a “TENA Local Class” (LC)?

- A TENA Definition Language (TDL) form of data encapsulation
 - “Class”: Model of an object, including state and behavior
 - “Local”: Methods are performed “locally” (by the app calling the methods)
 - In contrast to remote methods, which may be performed by a remote application
- Data types that can not be directly published, but can be
 - Used as attributes in Classes (SDOs), Messages & other LCs
 - Used as method parameters and return values
- If a TENA OM defines LCs with custom constructors or methods, then an OM “implementation” must be provided
 - For the TENA Standard OMs (e.g., TENA-TSPI-v5), OM implementations (software libraries) are provided in the OM distribution



TSPI local class in the TRMC repository

[Repository Home](#) \ [Browse Components](#) \ [TENA](#) \ [TSPI \(ObjectModel\)](#)



TENA-TSPI-v5

[Download](#)

[State:](#) Released

[Restrictions:](#) Public

[Imports:](#) [TENA-Time-v2](#)

[Create Helpdesk Case](#)

[Middleware:](#) 6.0.7

6.0.6

6.0.5.1

6.0.5

Updated: 2/14/18 11:07 AM Superseded By: None

Point of Contact: [TENA](#) Supersedes: None

[TDL](#) | [Documentation](#) | [History](#)

```
1 // This software was written under US Government contracts 1435-04-01-CT-31085,
2 // DASG60-02-D-0006-040, DASG60-02-D-0006-122, and W900KK-09-C-0013; and the US
3 // Government retains unlimited rights to the software.
4
5 import <TENA-Time-v2.tdl>
6 /*
7 * This is version 5 of the TSPI (Time, Space Position Information) object model
8 * developed by the TENA SDA project. This object model is intended to support
9 * TENA::TSPI local class objects that holds information about time, position,
10 * velocity, acceleration, and orientation of an object. The time information
11 * held in TSPI can be converted between UNIX, GPS, and UTC formats. The
12 * position information held in TSPI can be converted between Geocentric,
13 * Geodetic, Local Spherical Tangent Plane, and Local Tangent Plane
14 * East-North-Up coordinate systems.
15 */
```



TSPI local class in the TRMC repository

```
1216  
1217     enum AngularAccelerationKind  
1218     {  
1219         AngularAccelerationKind_Geocentric,  
1220         AngularAccelerationKind_LT_Penu  
1221     };  
1222  
1223     local class AngularAcceleration  
1224     {  
1225         // Each of the below constructors sets a single attribute  
1226         AngularAcceleration( GeocentricAngularAcceleration geocentricAngularAcceleration );  
1227         AngularAcceleration( LT_PenuAngularAcceleration ltpENUacceleration );  
1228  
1229         // Quick way to see what kind of angular acceleration is held here  
1230         AngularAccelerationKind get_kind_asTransmitted() const;  
1231  
1232         readonly optional GeocentricAngularAcceleration geocentric_asTransmitted;  
1233         readonly optional LT_PenuAngularAcceleration ltpENU_asTransmitted;  
1234     };  
1235  
1236     /*  
1237      * Local class that can represent TSPI information in a variety of SRFs.  
1238      */  
1239     local class TSPI  
1240     {  
1241         TSPI( Time theTime, Position position );  
1242  
1243         Time time;  
1244         Position position;  
1245         optional Velocity velocity;  
1246         optional Acceleration acceleration;  
1247         optional Orientation orientation;  
1248         optional AngularVelocity angularVelocity;  
1249         optional AngularAcceleration angularAcceleration;  
1250     };  
1251 } // package TENA
```



TSPI local class TDL

local class TSPI

{

 TSPI(Time theTime, Position position);

 Time time;

 Position position;

optional Velocity velocity;

optional Acceleration acceleration;

optional Orientation orientation;

optional AngularVelocity angularVelocity;

optional AngularAcceleration angularAcceleration;

};



TSPI “holder” attributes

- You might here us refer to the “generic” TSPI attributes as “holder” attributes
 - Position
 - Velocity
 - Acceleration
 - Orientation
 - Angular velocity
 - Angular acceleration
- Each of these attributes hold a more specific kind of data



TSPI holder attributes - minimum example

TENA-TSPI Local Class - "Holder" Attributes

Required Attributes

Position ("hold" one kind)

Geodetic Geocentric LTP-ENU LSTP Time
nanoseconds since 1970

Optional Attributes

Orientation (if set, "hold" one kind)

FRDwrtGeocentricBodyFixedZYX FRDwrtLTPenuBodyFixedZYX
DIS
FRDwrtGeocentricQuaternion FRDwrtLTPenuQuaternion

Velocity (if set, "hold" one kind)

Geocentric LTP-ENU LSTP

Acceleration (if set, "hold" one kind)

Geocentric LTP-ENU LSTP

Angular Velocity (if set, "hold" one kind)

Geocentric LTP-ENU

Angular Acceleration (if set, "hold" one kind)

Geocentric LTP-ENU

COLOR KEY

Specific kind set
Holder set
Not set

Here, we're setting a time and a geodetic position, but we could have set other position kinds instead



TSPI holder attributes – all attributes example

TENA-TSPI Local Class - "Holder" Attributes

Required Attributes

Position ("hold" one kind)

Geodetic Geocentric LTP-ENU LSTP Time
nanoseconds since 1970

Optional Attributes

Orientation (if set, "hold" one kind)

FRDwrtGeocentricBodyFixedZYX FRDwrtLTPenuBodyFixedZYX
DIS
FRDwrtGeocentricQuaternion FRDwrtLTPenuQuaternion

We're now setting a specific kind for every TSPI attribute

Velocity (if set, "hold" one kind)

Geocentric LTP-ENU LSTP

Acceleration (if set, "hold" one kind)

Geocentric LTP-ENU LSTP

Angular Velocity (if set, "hold" one kind)

Geocentric LTP-ENU

Angular Acceleration (if set, "hold" one kind)

Geocentric LTP-ENU

COLOR KEY

Specific kind set
Holder set
Not set

TSPI holder attributes – a more typical example

TENA-TSPI Local Class - "Holder" Attributes

Required Attributes

Position ("hold" one kind)

- Geodetic
- Geocentric
- LTP-ENU
- LSTP**
- Time
- nanoseconds since 1970

Optional Attributes

Orientation (if set, "hold" one kind)

- FRDwrtGeocentricBodyFixedZYX
- FRDwrtLTPenuBodyFixedZYX**
- DIS
- FRDwrtGeocentricQuaternion
- FRDwrtLTPenuQuaternion

Velocity (if set, "hold" one kind)

- Geocentric**
- LTP-ENU
- LSTP

Acceleration (if set, "hold" one kind)

- Geocentric
- LTP-ENU
- LSTP**

Angular Velocity (if set, "hold" one kind)

- Geocentric
- LTP-ENU**

Angular Acceleration (if set, "hold" one kind)

- Geocentric
- LTP-ENU**

Here's a more typical example

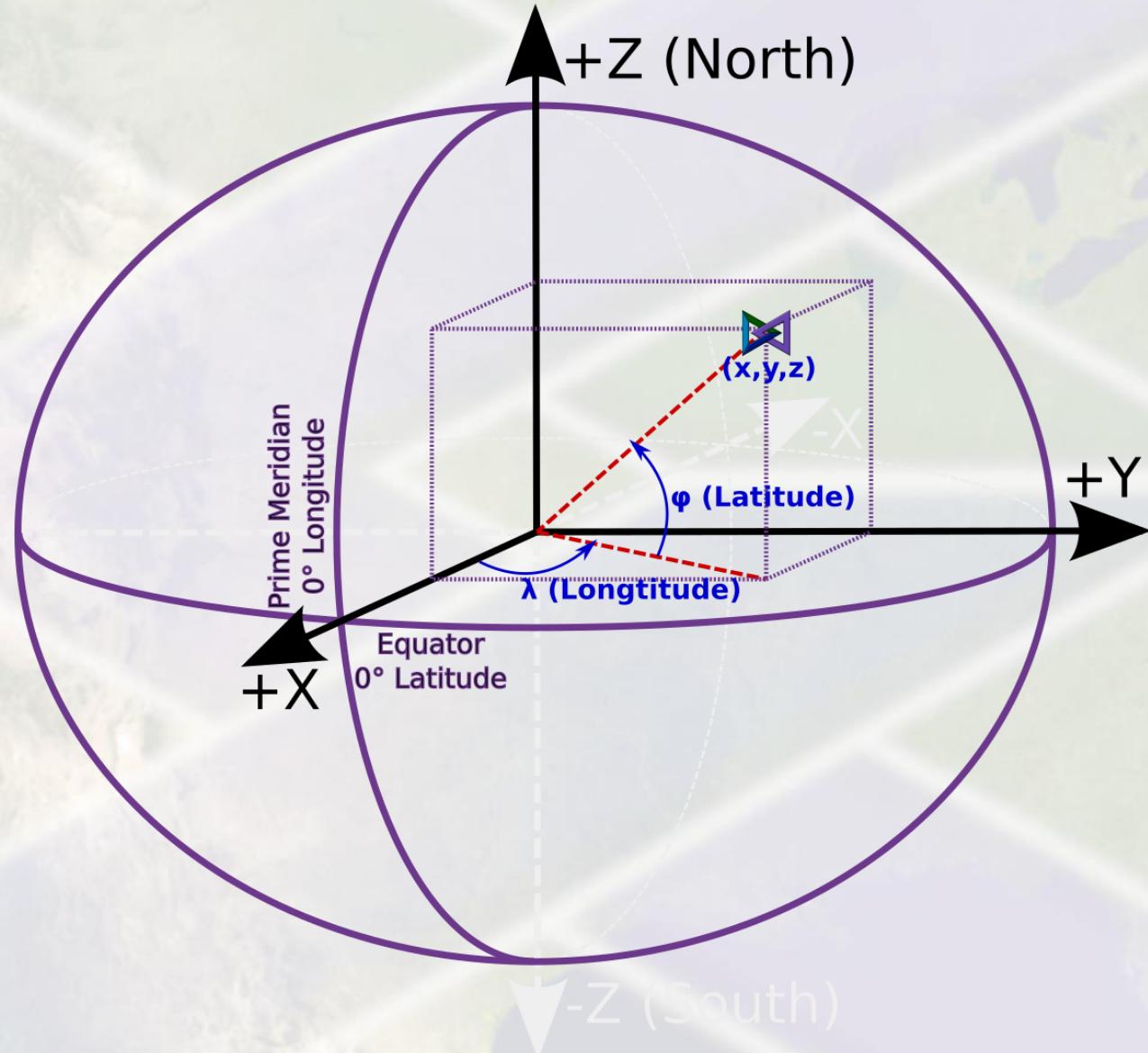
COLOR KEY

- Specific kind set**
- Holder set
- Not set

TSPI standard implementation

Provides conversions between specific kinds of attributes

- e.g., conversion between geodetic and geocentric positions



Automatic TSPI conversion example

TSPI attribute conversions are automatic

Publisher and Subscriber use
same coordinate system

TENA Publisher
(sets **geographic** positions in TSPI)

Publisher sends SDO with
TSPI position in **geographic**

TENA Subscriber
(uses **geographic** positions from TSPI)

*Subscriber gets a **geographic** position
No conversion needed or performed*

Publisher and Subscriber use
different coordinate system

TENA Publisher
(sets **geographic** positions in TSPI)

Publisher sends SDO with
TSPI position in **geographic**

TENA Subscriber
(uses **geocentric** positions from TSPI)

*Subscriber gets a **geocentric** position
Conversion is automatically performed*

This process works the same for any of the specific kinds within a TSPI holder attribute. If conversion is not needed, it will not be performed. If conversion is needed it will be automatically performed. If the subscriber needs to know the specific kind, there are methods available that indicate this.



Attributes of the TSPI local class

- We will look at some of the details of
 - Time
 - Position
 - Orientation
 - Velocity
 - Acceleration
 - Angular velocity
 - Angular acceleration



TENA-Time-v2 OM

- The TENA-TSPI-v5 OM imports the TENA-Time-v2 OM allowing the SPI local class to have a TENA-Time LC attribute
- A Time object can be constructed 7 different ways (different constructor inputs)
 - Default construct to “now”
 - Nanoseconds since 1970
 - Seconds since 1970 plus nanoseconds into current second
 - Unix time
 - GPS time
 - UTC
 - YTD time



TENA Time Local Class

local class Time

{

 Time(); // Default construct to "now"

 Time(int64 nanosecondsSince1970);

 Time(int32 secondsSince1970, uint32 nanosecondsIntoSecond);

 Time(UnixTime theUnixTime); // Converts a UnixTime to a Time

 Time(GPSTime theGPSTime); // Converts a GPSTime to a Time

 Time(UTCtime theUTCtime); // Converts a UTCtime to a Time

 Time(YTDtime theYTDtime); // Converts a YTDtime to a Time

readonly int64 nanosecondsSince1970; // Can cover +/-292.27 years from 1970

/*

* Method to set the Time using nanosecondsSince1970.

*/

void set(**in** int64 nanosecondsSince1970);

};

Quick review of terms

- Ellipsoid
- SRF: Spatial reference frame
- Geocentric
- Geodetic
- LTP: Local tangent plane
- LSTP: Local spherical tangent plane



Ellipsoid

- “A surface that may be obtained from a sphere by deforming it by means of directional scalings”
 - <https://en.wikipedia.org/wiki/Ellipsoid>
- “An Earth ellipsoid... is a mathematical figure approximating the Earth's form... Various different ellipsoids have been used as approximations”
 - https://en.wikipedia.org/wiki/Earth_ellipsoid



Spatial reference frame (SRF)

- Used as the basis for describing the position of an object in space
- The combination of:
 - An ORM (e.g., a model of the Earth), and
 - A compatible coordinate system (e.g., a Cartesian coordinate system for a three-dimensional space)
- Coordinates specify positions with respect to the ORM
- For the TSPI OM, we are focused on Earth ORMs and a handful of common coordinate system types
 - TSPI OM SRFs have an RT code, e.g., WGS84

Kinds of TENA Position

- World scope
 - Geodetic
 - Latitude, longitude, and HAE
 - Geocentric
 - Also commonly referred to as ECEF
 - Cartesian X, Y, Z coordinates
- Local scope
 - LTP-ENU
 - Cartesian X, Y, Z coordinates
 - LSTP
 - RAE coordinates



Why not “just one” common coordinate system?

- In simple exercises we may have the luxury to specify and use a single well-defined coordinate system
- In distributed LVC operations we usually need to address exchange of information between systems that have different TSPI representations
- Even within single system, there is often need for multiple coordinate systems
 - e.g., geodetic (latitude and longitude) is often used to display position to people on screens, while Cartesian (XYZ) coordinates are more convenient for mathematical functions under the hood



TENA Position Local Class

local class Position

```
{
```

```
// Each of the constructors sets a single attribute
```

```
Position( GeocentricPosition position );
```

```
Position( GeodeticPosition position );
```

```
Position( LTPenuPosition position );
```

```
Position( LSTPposition position );
```

Only one of these will be set

```
// Quick way to see what kind of position is held here
```

```
PositionKind get_kind_asTransmitted() const;
```

```
readonly optional GeocentricPosition geocentric_asTransmitted;
```

```
readonly optional GeodeticPosition geodetic_asTransmitted;
```

```
readonly optional LTPenuPosition ltpENU_asTransmitted;
```

```
readonly optional LSTPposition lstp_asTransmitted;
```

```
};
```

Geodetic

- Locations are described in terms of latitude, longitude, and HAE
 - The TSPI OM represents latitude and longitude in decimal degrees (rather than degrees, minutes, and seconds) and HAE in meters
- Latitude: the angle between the equatorial plane and a point on the ellipsoid surface
 - Equator: 0° latitude
 - North pole: 90° north latitude
 - South pole: is at 90° south latitude
 - Decimal degrees: positive indicates north, and negative indicates south
- Longitude: lines that run from north to south along surface of ellipsoid
 - Prime meridian: 0° longitude (divides ellipsoid into the eastern and the western Hemispheres)
 - 180th meridian (or antimeridian): 180° longitude
 - Decimal degrees: positive indicates east, and negative indicates west



TENA GeodeticPosition LC



local class GeodeticPosition

{

GeodeticPosition(GeodeticSRF srf, **float64** latitudeInDegrees,
float64 longitudeInDegrees, **float64** heightAboveEllipsoidInMeters);
GeodeticPosition(GeodeticSRF srf, Position position);

void set_latitudeInDegrees(**in** **float64** latitudeInDegrees);
void set_longitudeInDegrees(**in** **float64** longitudeInDegrees);
void set_heightAboveEllipsoidInMeters(
in **float64** heightAboveEllipsoidInMeters);

readonly GeodeticSRF srf;
readonly **float64** latitudeInDegrees;
readonly **float64** longitudeInDegrees;
readonly **float64** heightAboveEllipsoidInMeters;

};



Geocentric

- “Relating to, measured from, or as if observed from the earth's center”
 - <https://www.merriam-webster.com/dictionary/geocentric>
- TSPI OM: ECEF 3-dimensional Cartesian coordinate system
 - The X-axis points toward the prime meridian
 - The Y-axis points 90 degrees away from the X-axis along the equatorial plane
 - The Z-axis points north



TENA GeocentricPosition LC



local class GeocentricPosition

{

GeocentricPosition(GeocentricSRF srf, **float64** xInMeters,
float64 yInMeters, **float64** zInMeters);
GeocentricPosition(GeocentricSRF srf, Position position);

readonly GeocentricSRF srf;

float64 xInMeters;

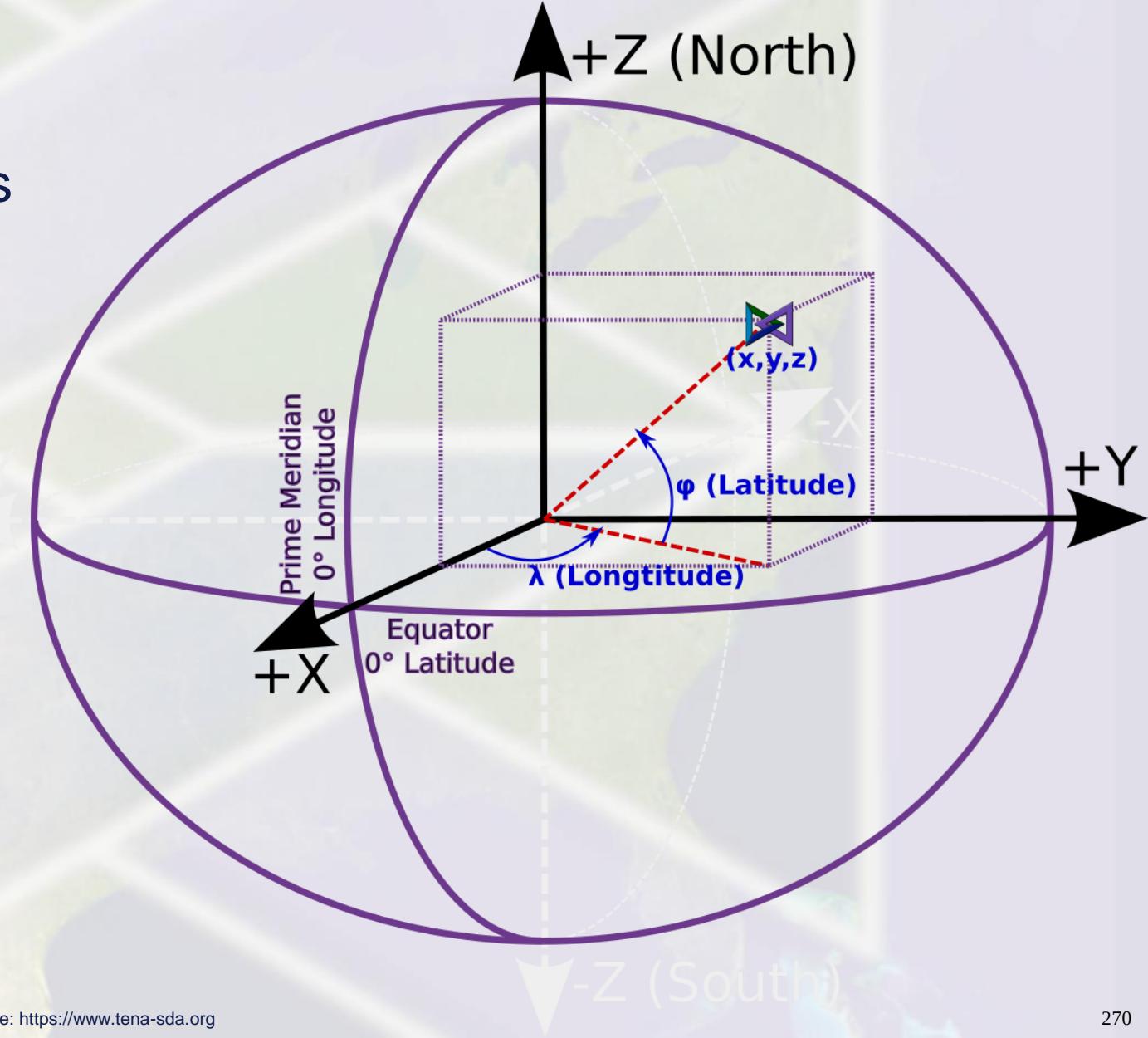
float64 yInMeters;

float64 zInMeters;

};

TENA TSPI: Geodetic versus Geocentric

- Geodetic
 - Latitude in decimal degrees
 - Longitude in decimal degrees
 - HAE in meters
- Geocentric
 - X in meters
 - Y in meters
 - Z in meters



“Local” coordinate systems

- LTP-ENU: Cartesian “XYZ” coordinate system centered at a specified position
 - ENU orientation that can be rotated
 - Commonly used by test ranges
- LSTP: Spherical “RAE” coordinate system centered at a specified position
 - North orientation that can be rotated
 - Used by tracking systems that follow POIs
 - “Center” of LSTP is the tracking system location



local class LTPenuPosition

{

 LTPenuPosition(LTPenuSRF srf, float64 xInMeters,
 float64 yInMeters, float64 zInMeters);

 LTPenuPosition(LTPenuSRF srf, Position position);

 readonly LTPenuSRF srf;

 float64 xInMeters; // East position in meters

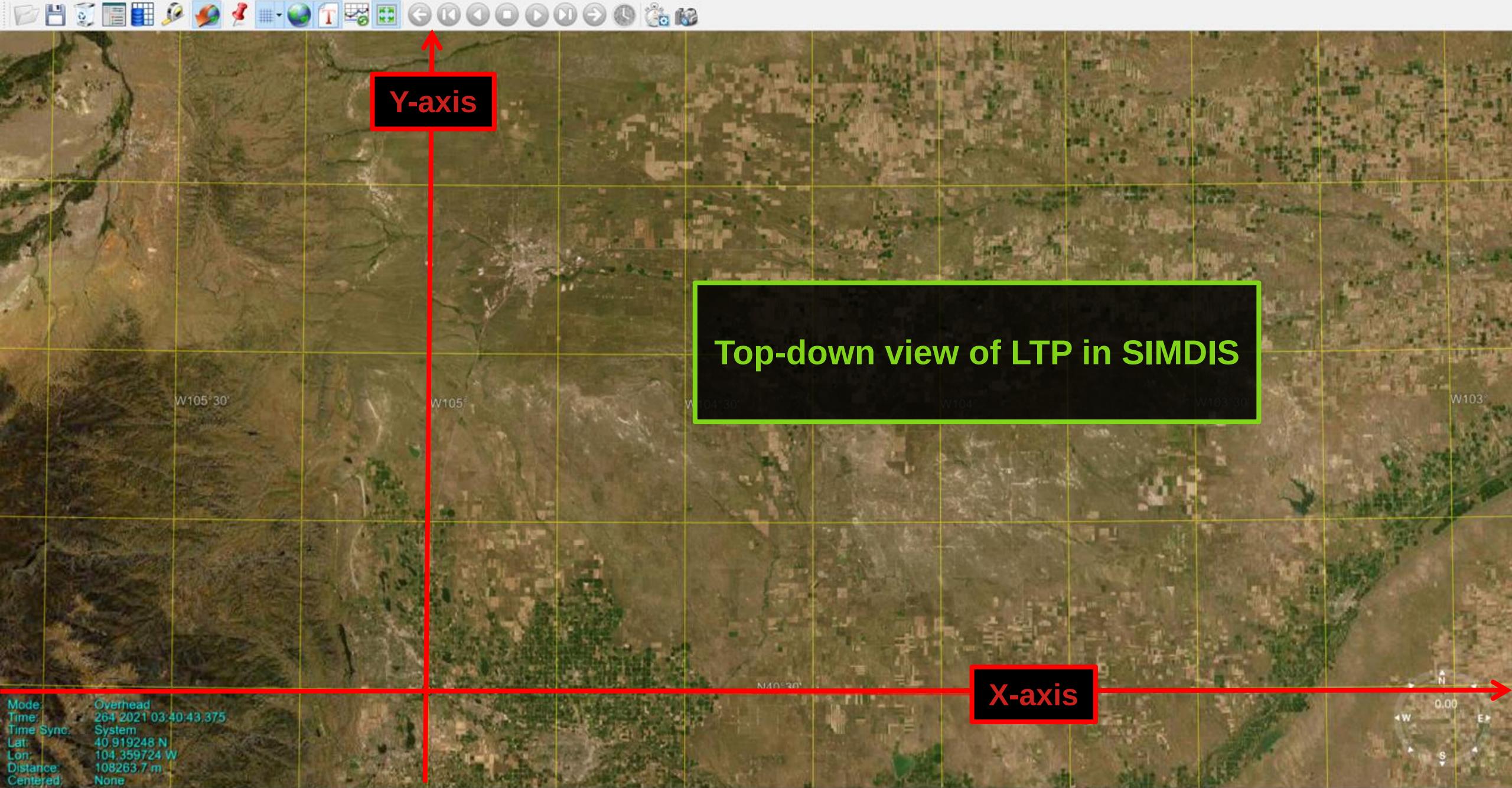
 float64 yInMeters; // North position in meters

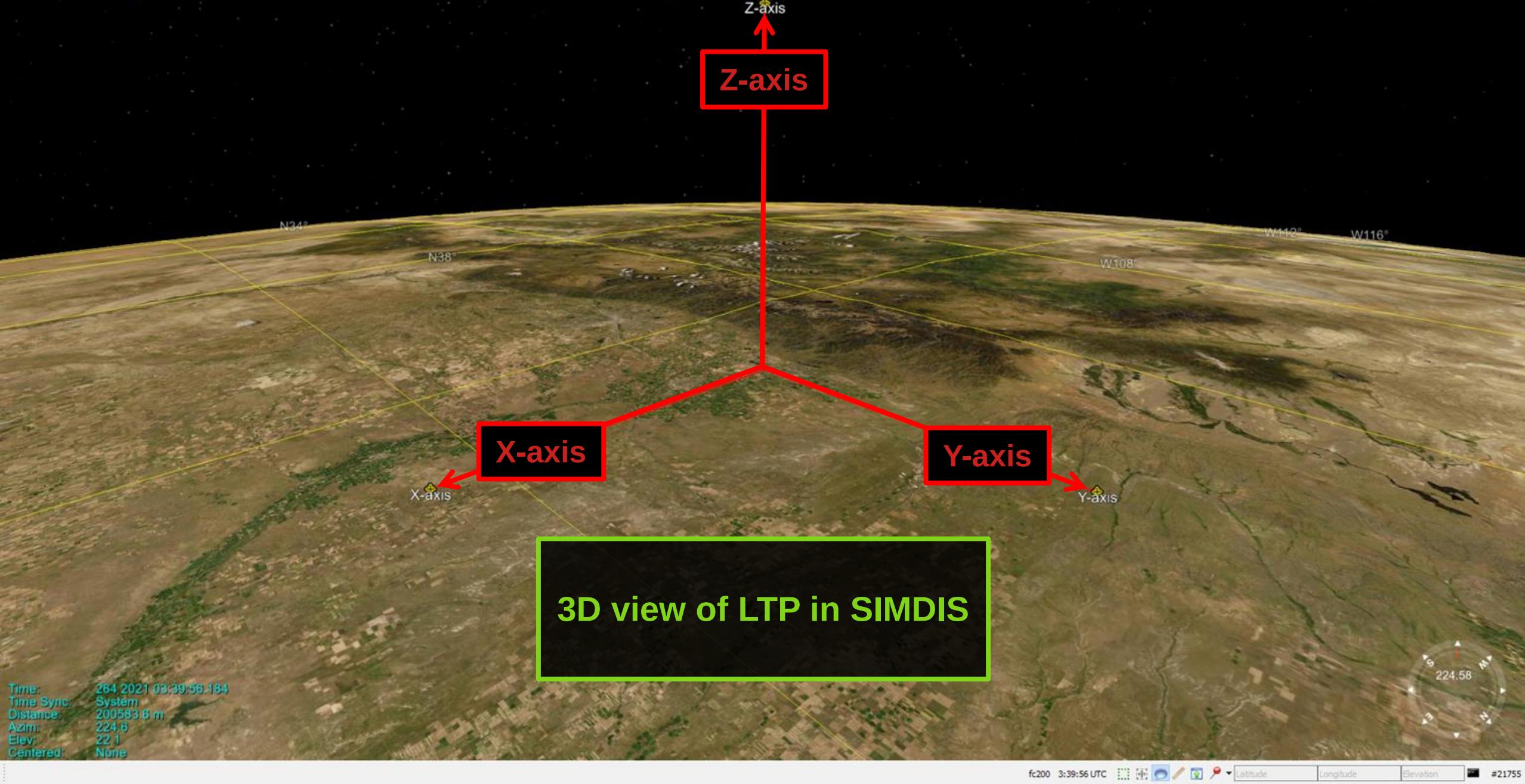
 float64 zInMeters; // Up position in meters

};

LTP-ENU example

- LTP-ENU coordinate system centered somewhere near Fort Collins, Colorado
- Geodetic SRF
 - RT code: WGS84
 - 40.5 degrees north latitude
 - 105 degrees west longitude (-105)
 - 2000 meters HAE
- X-axis points east
 - Negative X points west
- Y-axis points north
 - Negative Y points south
- Z-axis points “up” away from Earth
 - Negative Z points “down” toward and into the Earth
- We will look at two-views
 - Top-down (2D view of XY plane, tangent to Earth surface)
 - 3D view to give additional perspective





- LSTP and LTP-ENU coordinate systems both have a user-specified local origin, but for LSTP:
 - Positions are given in terms of
 - Azimuth clockwise around origin
 - Measured in radians
 - Zero radians is North
 - Elevation
 - Measured in radians
 - Positive elevation angles up, negative angles down
 - Range
 - Measured in meters
 - Distance from origin



TENA LSTPposition LC

```
local class LSTPposition
```

```
{
```

```
    LSTPposition(LSTPsrf srf, float64 azimuthInRadians,  
                 float64 elevationInRadians, float64 rangeInMeters );
```

```
    LSTPposition(LSTPsrf srf, Position position );
```

```
    void set_azimuthInRadians(in float64 azimuthInRadians);
```

```
    void set_elevationInRadians(in float64 elevationInRadians);
```

```
    void set_rangeInMeters(in float64 rangeInMeters);
```

```
    readonly LSTPsrf srf;
```

```
    readonly float64 azimuthInRadians;
```

```
    readonly float64 elevationInRadians;
```

```
    readonly float64 rangeInMeters;
```

```
};
```

Kinds of TENA orientation

- Orientation defined with Euler angles
 - FRDwrtGeocentricBodyFixedZYX
 - ZYX coordinates in radians
 - DIS
 - FRDwrtGeocentricBodyFixedZYX where RT code is always WGS84
 - FRDwrtLTPenuBodyFixedZYX
 - Easiest kind to translate between heading, pitch, and roll angles
- Orientation defined with quaternions
 - FRDwrtGeocentricQuaternion
 - FRDwrtLTPenuQuaternion



Euler angles for object orientation

- A set of three angles used to describe orientation of an object with respect to a coordinate system
 - DIS uses mathematically convenient geocentric Euler angles rotated by ZYX
 - The intuitive concept of heading, pitch, and roll can be represented as Euler angles based on an LTP origin at the objects geocentric position
- Susceptible to gimbal lock
 - *"The loss of one degree of freedom in a three-dimensional, three-gimbal mechanism that occurs when the axes of two of the three gimbals are driven into a parallel configuration, "locking" the system into rotation in a degenerate two-dimensional space"*
https://en.wikipedia.org/wiki/Gimbal_lock



Quaternions for object orientation

- The quaternion number system extends the complex numbers and is applied to mechanics in 3D space
- Quaternions can be used for calculations involving 3D rotation
- Unlike Euler angles, quaternions are not susceptible to gimbal lock



Kinds of TENA Velocity

- Geocentric
 - XYZ in meters per second
- LTP-ENU
 - XYZ in meters per second
- LSTP
 - Azimuth and elevation in radians per second
 - Range in meters per second



TENA Velocity Local Class



local class Velocity

```
{
```

```
// Each of the below constructors sets a single attribute
```

```
Velocity( GeocentricVelocity geocentricVelocity );
```

```
Velocity( LTPenuVelocity ltpENUvelocity );
```

```
Velocity( LSTPvelocity lstpVelocity );
```

```
// Quick way to see what kind of velocity is held here
```

```
VelocityKind get_kind_asTransmitted() const;
```

```
readonly optional GeocentricVelocity geocentric_asTransmitted;
```

```
readonly optional LTPenuVelocity ltpENU_asTransmitted;
```

```
readonly optional LSTPvelocity lstp_asTransmitted;
```

```
};
```

Kinds of TENA Acceleration

- Geocentric
 - XYZ in meters per second squared
- LTP-ENU
 - XYZ in meters per second squared
- LSTP
 - Azimuth and elevation in radians per second squared
 - Range in meters per second squared



TENA Acceleration Local Class



local class Acceleration

```
{
```

```
// Each of the below constructors sets a single attribute  
Acceleration( GeocentricAcceleration geocentricAcceleration );  
Acceleration( LTPenuAcceleration ltpENUacceleration );  
Acceleration( LSTPacceleration lstpAcceleration );
```

```
// Quick way to see what kind of acceleration is held here  
AccelerationKind get_kind_asTransmitted() const;
```

```
readonly optional GeocentricAcceleration geocentric_asTransmitted;  
readonly optional LTPenuAcceleration ltpENU_asTransmitted;  
readonly optional LSTPacceleration lstp_asTransmitted;
```

```
};
```



Kinds of angular velocity

- Geocentric
 - XYZ in radians per second
- LTP-ENU
 - XYZ in radians per second





Kinds of angular acceleration

- Geocentric
 - XYZ in radians per second squared
- LTP-ENU
 - XYZ in radians per second squared





TSPI demo

- Publishers
 - SittingDuck
 - EntityStepper
- Subscribers
 - SIMDIS with TENA plugin
 - EntityAngler
 - TDCS data collector, using DataView



TSPI demo

EntityStepper - Connected

File About

TENA connection Entity publication Diagnostics

Constant attributes

Identifier: PleasantWatson-53263
Source identifier: EntityStepper-21774-29001

Entity ID

Site ID: 21774
App ID: 29001
Object ID: 53263

LVC indicator: Constructive

DIS entity type

Kind: 1
Domain: 2
Country: 225

Variable attributes

Y meters: -4696144.03
Z meters: 4131832.19

HPR orientation

Heading degrees: 35.00
Pitch degrees: 10.00
Roll degrees: 5.00

DIS/Euler orientation

Psi degrees about Z: 19.88
Theta degrees about Y: -46.53
Phi degrees about X: -135.69

Quaternion orientation: w:0.08, x:0.88, y:0.46, z:0.06

Reset constant defaults Reset variable defaults

Only send updates when "Update SDO" button is pressed Create SDO
Send updates as attribute values are changed Update SDO
Destroy SDO

Time of creation: 2022-11-16 09:47:50 -0800
Time of update: 2022-11-16 09:51:10 -0800
State version: 11

Connected to TENA execution

EntityStepper - Connected

File About

TENA connection Entity publication Diagnostics

Constant attributes

Identifier: OklahomaArchbanger-43343
Source identifier: EntityStepper-40550-51886

Entity ID

Site ID: 40550
App ID: 51886
Object ID: 43343

LVC indicator: Constructive

DIS entity type

Kind: 1
Domain: 2
Country: 225

Variable attributes

Y meters: -4698232.57
Z meters: 4127332.52

HPR orientation

Heading degrees: -20.00
Pitch degrees: -10.00
Roll degrees: 10.00

DIS/Euler orientation

Psi degrees about Z: 99.70
Theta degrees about Y: -36.21
Phi degrees about X: 171.20

Quaternion orientation: w:0.02, x:-0.58, y:-0.81, z:0.12

Reset constant defaults Reset variable defaults

Only send updates when "Update SDO" button is pressed Create SDO
Send updates as attribute values are changed Update SDO
Destroy SDO

Time of creation: 2022-11-16 09:47:43 -0800
Time of update: 2022-11-16 09:51:18 -0800
State version: 9



EntityAngler - Connected

File About

TENA connection Point of Interest (POI) subscription Diagnostics

POI A: OklahomaArchbanger-43343

Latitude degrees: 40.5100
Longitude degrees: -104.9700
HAE meters: 9434

Geocentric position:

X meters: -1256251
Y meters: -4698233
Z meters: 4127333

Geodetic position:

Latitude degrees: 40.5600
Longitude degrees: -104.9200
HAE meters: 9855

POI B: PleasantWatson-53263

Latitude degrees: 40.5600
Longitude degrees: -104.9200
HAE meters: 9855

Geocentric position:

X meters: -1251303
Y meters: -4696144
Z meters: 4131832

Geodetic position:

Latitude degrees: 40.5600
Longitude degrees: -104.9200
HAE meters: 9855

TSPI time: 2022-11-16 09:51:18 Pacific Standard Time

Heading degrees: 35.00
Pitch degrees: 10.00
Roll degrees: 5.00

DIS/Euler orientation:

Psi degrees about Z: 19.88
Theta degrees about Y: -46.53
Phi degrees about X: -135.69

Latitude degrees to POI B: 57
Longitude degrees to POI B: 17
Elevation degrees to POI A: 6
Zenith degrees to POI B: 84

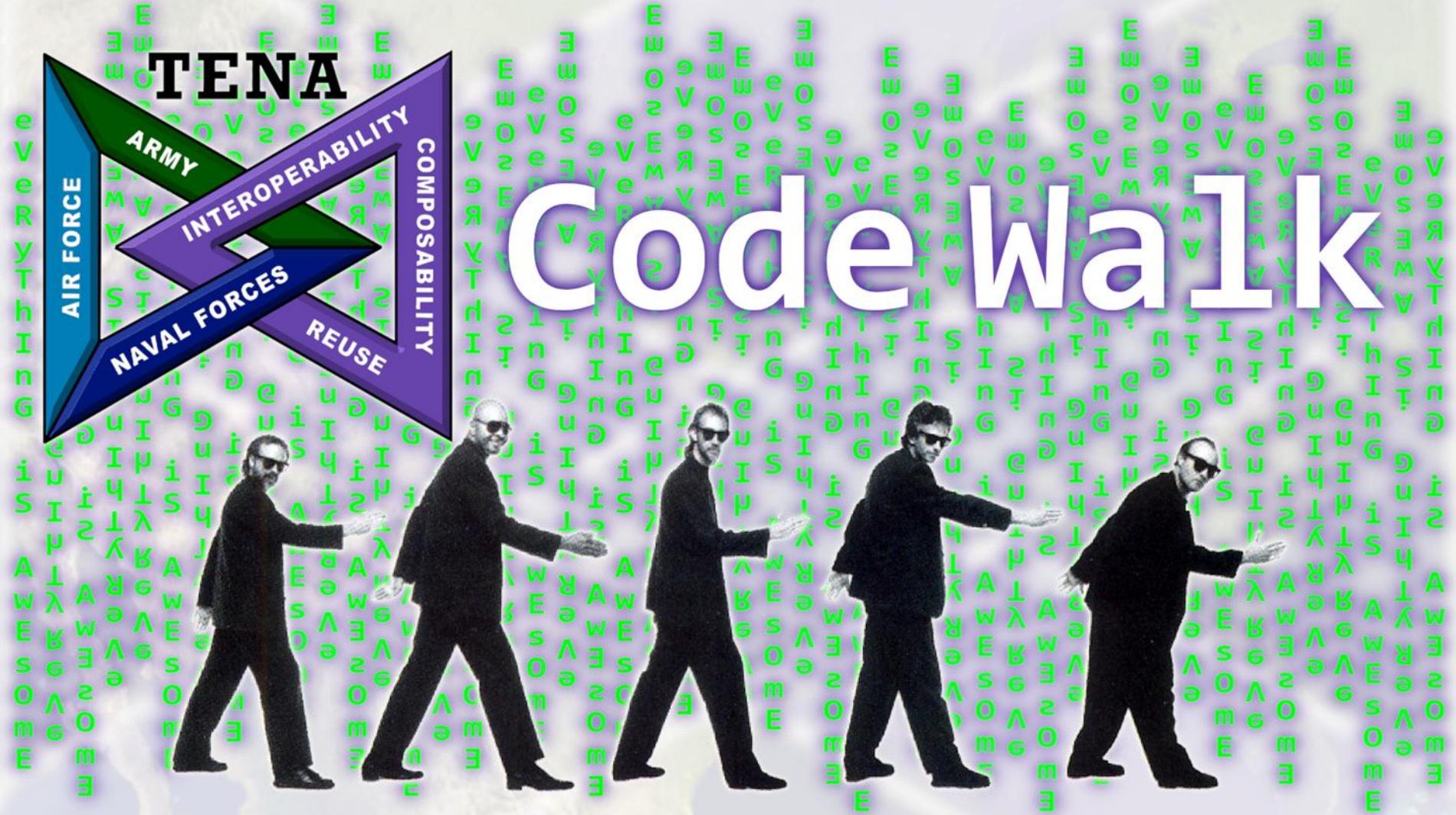
Azimuth degrees to POI B: 57
Elevation degrees to POI A: 6
Zenith degrees to POI A: 84

TSPI time: 2022-11-16 09:51:10 Pacific Standard Time

Distance between POIs in meters: 7007

TSPI code walk?

Depending on time, we can always bring up an IDE and do a code walk



Example: Geodetic to Geocentric in C++

```
using namespace TENA;

// Create a GeocentricSRF that we'll use for GeocentricPositions
GeocentricSRFPtr gCSR( GeocentricSRF::create());

// Create a GeodeticSRF that we'll use for GeodeticPositions
GeodeticSRFPtr gdSR( GeodeticSRF::create());

// Create a GeodeticPosition
GeodeticPositionPtr gdPosition(
    GeodeticPosition::create(gdSR, 40.0, -105.0, 1500.0));

// Hold GeodeticPosition in Position
PositionPtr position( Position::create(gdPosition));

// Create a GeocentricPosition from Position
GeocentricPositionPtr gcPosition(
    GeocentricPosition::create(gCSR, position));
```

Example: Geocentric to Geodetic in C++

```
using namespace TENA;

// Create a GeocentricSRF that we'll use for GeocentricPositions
GeocentricSRFPtr gcSRF(GeocentricSRF::create());

// Create a GeodeticSRF that we'll use for GeodeticPositions
GeodeticSRFPtr gdSRF(GeodeticSRF::create());

// Create a GeocentricPosition
GeocentricPositionPtr gcPosition(
    GeocentricPosition::create(gcSRF, -1257337.0, -4692444.0, 4121334.0));

// Hold GeocentricPosition in Position
PositionPtr position(Position::create(gcPosition));

// Create a GeodeticPosition from Position
GeodeticPositionPtr gdPosition(
    GeodeticPosition::create(gdSRF, position));
```



Example: Geodetic to Geocentric in Java

```
// Create a GeocentricSRF that we'll use for GeocentricPositions
TENA.GeocentricSRF.LocalClass gcSRF =
    TENA.GeocentricSRF.LocalClass.create();

// Create a GeodeticSRF that we'll use for GeodeticPositions
TENA.GeodeticSRF.LocalClass gdSRF =
    TENA.GeodeticSRF.LocalClass.create();

// Create a GeodeticPosition
TENA.GeodeticPosition.LocalClass gdPosition =
    TENA.GeodeticPosition.LocalClass.create(gdSRF, 40.0, -105.0, 1500.0);

// Hold GeodeticPosition in Position
TENA.Position.LocalClass position =
    TENA.Position.LocalClass.create(gdPosition);

// Create a GeocentricPosition from Position
TENA.GeocentricPosition.LocalClass gcPosition =
    TENA.GeocentricPosition.LocalClass.create(gcSRF, position);
```



Example: Geocentric to Geodetic in Java

```
// Create a GeocentricSRF that we'll use for GeocentricPositions
TENA.GeocentricSRF.LocalClass gcSRF =
    TENA.GeocentricSRF.LocalClass.create();

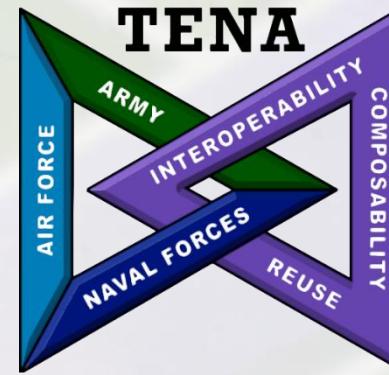
// Create a GeodeticSRF that we'll use for GeodeticPositions
TENA.GeodeticSRF.LocalClass gdSRF =
    TENA.GeodeticSRF.LocalClass.create();

// Create a GeocentricPosition
TENA.GeocentricPosition.LocalClass gcPosition =
    TENA.GeocentricPosition.LocalClass.create(
        gcSRF, -1257337.0, -4692444.0, 4121334.0);

// Hold GeocentricPosition in Position
TENA.Position.LocalClass position =
    TENA.Position.LocalClass.create(gcPosition);

// Create a GeodeticPosition from Position
TENA.GeodeticPosition.LocalClass gdPosition =
    TENA.GeodeticPosition.LocalClass.create(gdSRF, position);
```

source: <https://www.tena-sda.org>



Simulation time with TENA

Modeling and managing time in non-realtime simulations

Time in distributed simulations

- **Realtime**
- Managed simulation time - may be faster or slower than realtime
 - Scaled time
 - **Centralized simulation clock** with time sent to participants
 - Easy to implement and understand
 - Not usually sufficient for simulation requirements
 - We will provide a use case for a simple implementation
 - **Conservative time management** - time advance requests are aggregated, and smallest request is granted to all
 - There have been custom TENA OMs using this concept
 - **Optimistic time management** - participants operate at independent rates, but “roll back” when issues are detected
- A simple use case: TENA apps being developed for Link Margin Analysis (LMA)
 - OM overview: extending Standard OMs with mission planning timelines
 - SimEntityPub: Entity publisher for telemetry transmitters and receivers with mission path
 - SimControl: Controls instances of SimEntityPub entities along the mission timeline



Realtime

- How humans perceive time in the real world
- Sometimes referred to as wall clock time
- All real-world systems operate in real time
- There is no pause, fast-forward, rewind, etc. for “live” systems
- Faster or slower than “real time” in executions do not involve live/real time participants
 - This presents challenges in integrating simulation and real systems
- May still require synchronized high-resolution time sources at each participant
- TENA middleware is mostly used in realtime systems
 - But it is also used in many non-realtime simulations



Non-realtime / Simulation time

- We are concerned with time-based distributed simulations that have a concept of time
 - Not all simulations are distributed
 - Not all simulations have a concept of time (e.g., purely event-based)
- Time-based simulations may have an internal representation of time that may not be related to “real time”
 - Sometimes “simulation time” is “real time”
 - A training simulator for live participants is an example of a real time simulation
- A group of time-based simulations cooperating in a distributed system, need to have a shared understanding of “simulation time”



Event lists, faster or slower than realtime

- Some simulation run “as fast as possible”
 - Usually based on a list of events
 - This might mean slower or faster than realtime
 - Faster calculations (e.g., position updates) are fast
 - Easier to adjust for interaction with realtime systems
 - Slower calculations (e.g., nuclear effects on large areas) are slow
 - Harder (or impossible) for interaction with realtime systems
- Probably requires synchronized high-resolution time sources at each participant

Scaled time

- “Simply” manage a shared simulation clock and scale time faster or slower (compared to real time) depending on system capabilities and needs
 - A straightforward concept
 - May be sufficient for some use cases
 - Not sufficient for all use case



Discrete event simulations

- Simulation may or may not need the concept of a shared clock
- Just mentioned here for completeness
 - Simulation students were likely first introduced to the concept of a discrete event simulation before thinking about distributed simulations with shared/coordinated clocks

Issues in distributed simulation time

- Managed clock
 - In TENA, we can accomplish this several ways – e.g.,
 - Simple messages to convey time
 - A central time management application that calls RMIs to advance time
 - An SDO with time attributes that are updated
 - Published/updated by a time management application
 - Subscribed to by other distributed simulation participants
- Synchronized time
 - Each participant may need a high-resolution time source (TENA has the capability to integrate applications with high-resolution time sources)
- Communication latency issues (and packet ordering)
 - Communication between processes on the same computer are typically measured in nanoseconds (billions of a second)
 - Communication between processes on different computers are typically measured in milliseconds (thousandths of a second)
 - “Significant” latencies across networks (and packet ordering) should be considered in distributed simulation design

Some goals of simulation time management

- These are “some” common themes of simulation time management objectives, but requirements vary
 - Ability to run faster or slower than realtime rate
 - Run at realtime to interact with systems that can not support simulation time (e.g., live systems)
 - Detect and/or prevent causality violations (e.g., a bridge blows up and then a truck drives over where it used to be)
 - Ensure repeatable results
 - May require an agreement on how to order events with identical timestamps

Simulation time in TENA

- Needs will vary, but “typically”...
 - A central clock will need to be modeled in a custom OM
 - e.g., via a messaging system, SDO attributes, and/or RMI calls
 - A central time management application will need to be built
 - All applications in the TENA application should have the same concept of time as defined in the custom OM
- As simple use case: TENA apps being developed for Link Margin Analysis (LMA)
 - OM overview: extending Standard OMs with mission planning timelines
 - SimEntityPub: Entity publisher for telemetry transmitters and receivers with mission path
 - SimControl: Controls instances of SimEntityPub entities along the mission timeline
 - Our use case is “simple” because we are primarily interested in relative body angles, and we are not interested in ordering of events such as weapon fire messages



LMA OM use case and app demo

- Review of OM
- Demonstration with
 - SIMDIS for visualization of
 - Entity time, position and orientation
 - Simulation path for each entity
 - Line-of-bearing between entities
 - SimControl for control of the simulation timeline
 - SimEntityPub – four instances controlled by SimControl
 - 1 Telemetry transmitter
 - 3 Telemetry receivers
 - EntityAngler to show
 - Simulation time of each entity
 - Relative body angles between entities (pertinent to LMA)

Simple time points on a path

```
// Point local class
// Local Class used to identify a particular point that will be used as part
// of a planned or recorded path (see class Path).
// The Point local class is defined by an optional identifier and required
// timeOffsetInNanoseconds, position, and orientation. This construct is
// used as part of a sequence of planned or recorded points (e.g. a flight
// path).
local class Point
{
    uint64 timeOffsetInNanoseconds; // Time *offset* in nanoseconds
    TENA::Position position; // Position
    TENA::Orientation orientation; // Orientation
    optional string identifier; // Optional identifier
};
```

```
// Path local class
// Local class that provides the ability to link together a series of Point
// local classes in a sequence.
// Applications should check that time offsets are ascending in value. When
// offset values are not ascending in value, the user should be warned. It
// is then up to the application developer to determine if it is appropriate
// to re-order Points, which will be used for interpolation.
local class Path
{
    string identifier; // Identifier for Path
    vector<Point> points; // Sequence of Point objects
};

// EntityStatus enumeration
// Used as an Entity SDO class attribute to convey current status
enum EntityStatus
{
    EntityStatus_Uninitialized, // Waiting for run properties
    EntityStatus_Initializing, // Initializing
    EntityStatus_Initialized, // Initialized and ready
    EntityStatus_MovingToCommandedTime, // Moving to commanded time
    EntityStatus_ReachedCommandedTime, // Reached commanded time
    EntityStatus_CannotReachCommandedTime, // Time outside path timeline
    EntityStatus_Exiting, // Application is exiting
    EntityStatus_ErrorState // Application in error state
};
```

Custom entity with simple “advanceTo()” RMI

```
// LVC-MissionPlan-Entity SDO class
// Base class for SDOs under simulation control
// When the advanceTo() remote method is called, this method shall set
// EntityStatus to MovingToCommandedTime and set the new TSPI value.
class Entity : extends TENA::LVC::Entity
{
    // Path that this entity is associated with
    const Path path;

    // entityStatus enumeration attribute
    // Used to convey status to simulation controller. Set value to
    // "EntityStatus_Uninitialized" when SDO is first created (or reset).
    EntityStatus entityStatus;

    // timeZero optional TENA::Time attribute
    // This attribute is to be set when a value is passed from the
    // initialize() RMI and unset when the reset() RMI is called.
    optional TENA::Time timeZero;

    // initialize() remote method
    // Accepts TENA::Time timeZero from calling application
    // Set entity.timeZero to the value of timeZero passed via RMI
    // Set radioEnabled to false
    // Set tspli according to first Point in Path.
    // Set EntityStatus to "Initialized"
    void initialize(in TENA::Time timeZero);

    // advanceTo() remote method
    // Set TSPI based on interpolated values from Path
    oneway void advanceTo(in TENA::Time commandedTime);

    // uninitialized() remote method
    // Unset timeZero
    // Set tspli according to first Point in Path, with wallclock time.
    // Set EntityStatus to "Uninitialized"
    void uninitialized();
};
```

An overly simple example?

LMA mission planning is concerned with relative body angles along a timeline

More complex simulations need to account for event ordering



Example entities and paths – SimEntityPub instances

SimEntityPub - Publishing TXentity SDO - TX

File About

TENA connection Entity Configuration Path Configuration Antenna Configuration Status

Simulation Path

Load points from path file
Save points to path file

	Time offset	Latitude	Longitude	HAE	Heading	Pitch	Roll	Identifier
1	-20.000	40.5395590	-105.1967560	2213.000	90.000	0.000	0.000	Stowed
2	-10.000	40.5395590	-105.1967560	2213.000	90.000	45.000	90.000	Movingforlaunch
3	0.000	40.5395590	-105.1967560	2213.000	90.000	90.000	0.000	Launch
4	10.000	40.5395590	-105.1961841	3000.000	90.000	85.000	0.000	T: 10
5	20.000	40.5395590	-105.1944467	5300.000	90.000	80.000	0.000	T: 20
6	30.000	40.5387506	-105.1887174	8700.000	90.000	75.000	0.000	T: 30
7	40.000	40.5364000	-105.1783694	13200.000	101.000	70.000	0.000	T: 40
8	50.000	40.5317122	-105.1605650	18600.000	107.000	65.000	0.000	T: 50

Remove point Add point

Publish Entity SDO Destroy Entity SDO

Publishing TXentity SDO - TX

SimEntityPub - Publishing RXentity SDO - RX1

File About

TENA connection Entity Configuration Path Configuration Antenna Configuration Status

Simulation Path

Load points from path file
Save points to path file

	Time offset	Latitude	Longitude	HAE	Heading	Pitch	Roll	Identifier
1	-20.000	37.2051000	-101.6922000	20000.000	34.000	0.000	0.000	T: -20
2	1120.000	38.5707000	-100.5406000	20000.000	34.000	0.000	0.000	T: 1120

Remove point Add point

Publish Entity SDO Destroy Entity SDO

SimEntityPub - Publishing RXentity SDO - RX3

File About

TENA connection Entity Configuration Path Configuration Antenna Configuration Status

Simulation Path

Load points from path file
Save points to path file

	Time offset	Latitude	Longitude	HAE	Heading	Pitch	Roll	Identifier
1	-20.000	35.9191690	-95.1399670	20000.000	176.447	0.000	5.251	T: -20
2	0.000	35.8898090	-95.1398380	20000.000	-177.238	0.000	5.251	T: 0
3	20.000	35.8606150	-95.1436960	20000.000	-170.922	0.000	5.251	T: 20
4	40.000	35.8319420	-95.1514890	20000.000	-164.606	0.000	5.251	T: 40
5	60.000	35.8041360	-95.1631190	20000.000	-158.290	0.000	5.251	T: 60
6	80.000	35.7775350	-95.1784400	20000.000	-151.974	0.000	5.251	T: 80
7	100.000	35.7524590	-95.1972650	20000.000	-145.658	0.000	5.251	T: 100
8	120.000	35.7292140	-95.2193610	20000.000	-139.343	0.000	5.251	T: 120

Remove point Add point

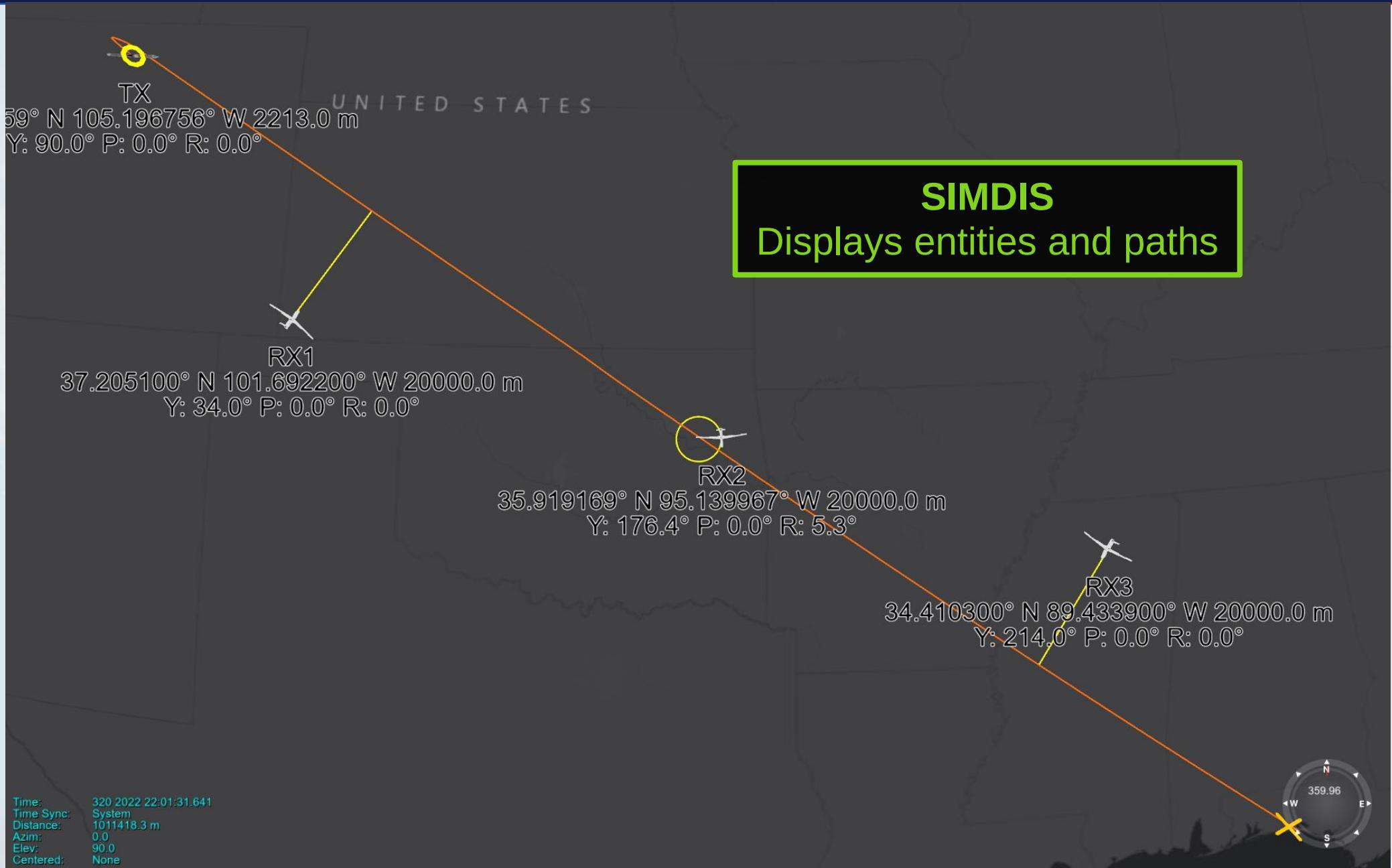
Publish Entity SDO Destroy Entity SDO

SimEntityPub

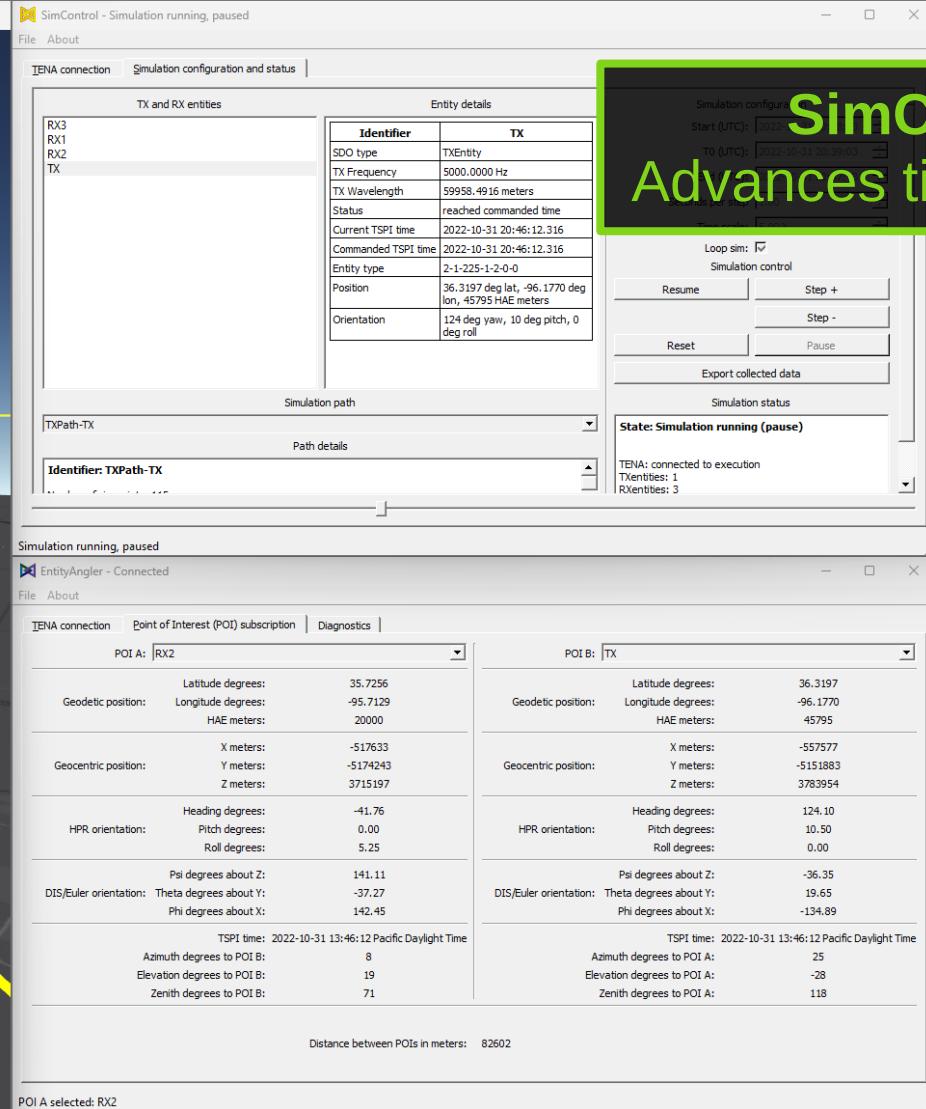
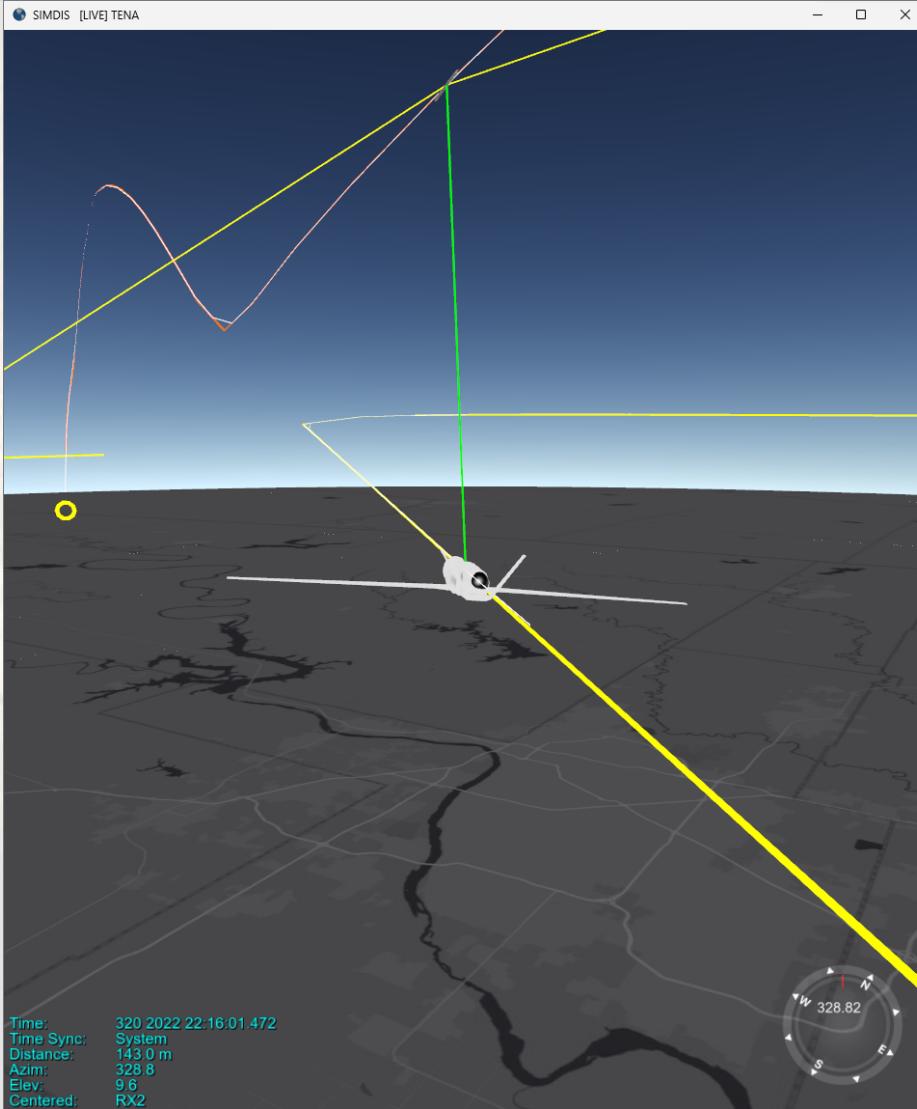
Publishes entities and paths

Waits for simulation time advancement via RMI

Example entities and paths – 2D view



All entities paused by SimControl – EntityAngler displaying relative body angles



SimControl
 Advances time via RMLs



Use case - Acme-TimeManagement OM

- *(Name changed from real use case)*
- Time Management Object (TMO)
- Defines communication mechanism for a time management service
- Used by apps under control of a time manager for advancing logical time
- Conveys time advance requests & grants



Acme-TimeManagement OM

```
import <TENA-Time-v2.tdl>

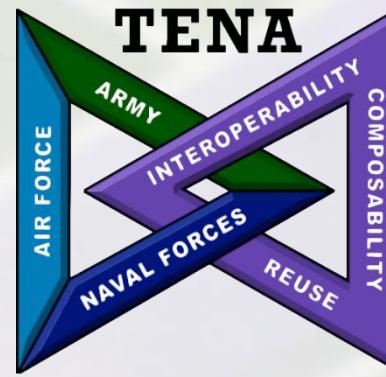
package Acme {

    enum TimePoint
    {
        TimePoint_Read, TimePoint_MixedReadWrite, TimePoint_Write
    };

    class TimeManagement
    {
        string identifier;
        TENA::Time requestSimTime;
        TimePoint requestTimePoint;
        boolean timeAdvanceGrant(
            in TENA::Time simTime,
            in TimePoint timePoint);
    };
}
```

Time advance usage

- Apps convey time advance requests to a time manager
 - Via requestSimTime and requestTimePoint fields
 - The identifier field identifies app to the time manager
- Time manager conveys time advance grants to the apps
 - Using the remote method timeAdvanceGrant()
- *Requests for a given time are answered by grants for a sim time advance*



OM design guidance



OM design guidance

- Intended audience: Systems and software engineers
- Pilfered/modified from the TENA Hands-On-Training slides
 - *(which contain more information than is summarized here)*



Good OM_s are the key to interoperability

- OM_s are “information exchange” agreements
- OM_s should be as clear and unambiguous as possible
- OM_s are like legal contracts
 - Care must be taken to minimize loopholes
 - The TENA Middleware will enforce these OM “contracts”



Flexibility

- TENA can be as flexible as you want, for example...

```
message FlexibleMessage
{
    vector<octet> data;
};
```

```
class FlexibleSDO {
    void doit(inout vector<octet> arg);
    vector<octet> data;
};
```

- ... which shows that there's such as thing as “too much flexibility”
- Sometimes “Flexibility” leads to “Chaos”
- With data interfaces, it’s best to be as specific as practical



OM flexibility use case

- Data producers and consumers need to agree on the data syntax and semantics for a meaningful exchange
 - Data semantics may not be needed if the consumer is purely generic (e.g., data logger, relay system), but most consumers need to understand the data
 - Too often, users want flexibility because they are unfamiliar with a system that can enforce consistent formal definitions in an efficient manner

Model the problem, not one solution to the problem

- If the problem involves determining the location of an object from a remote sensor, it may be prudent to avoid needlessly “littering” the OM with references to radar
- While radar may be used to solve the problem today, it may be more forward-thinking to use more general concepts and terms, such as “location sensor”
 - e.g., sonar can be used as a location sensor under water

Premature optimizations

- Avoid making premature (and potentially unnecessary) optimizations at the expense of clarity
- Avoid “contorting” OM design for speculative “optimizations”
 - Developers are notorious for “optimizing” software without performing the experimentation necessary to determine where significant resource bottlenecks of the system are
- “Contorted” designs are difficult to understand
 - Some optimizations can be made without “contorting” the OM design
 - Not all optimizations are speculative
- Optimizations are typically implementation-dependent
 - Sometimes implementation improvements can be deployed before an anticipated (speculative?) optimization is ever actually needed



Names matter



- Use precise, descriptive names
 - For an attribute of a sensor, the name `distanceToTarget` is better than just `distance` which is better than just `d`
- Use units in names whenever appropriate
 - Consider the sensor attribute `distanceToTarget` above
 - What is the unit of measure here: feet, meters, miles, furlongs?
 - `distanceToTargetInMeters` is better than `distanceToTarget`



Names matter (continued)

- Units should be appended to the end of name
 - Name and units should be separated by the word 'In'
 - Provides a clear demarcation between name and units
 - Improves readability e.g., distanceInMeters is clearer than distanceMeters
- Units should also be used in enumerations, SDOs, messages, and method names.
 - `get_angleInRadians()` or `get_angleInDegrees()` is preferred over `get_angle()`





Use an SDO or a Message?

- SDOs should be used to model things or concepts that have more than a momentary existence
 - e.g., a tank, a missile, a command post, or the current threat level
- Message should be used to model things or concepts that are ephemeral in nature
 - e.g., an explosion, or announcements such as “Lunchtime!”



Use a Message or an SDO remote method?

- Messages model one-to-many communication
- SDO remote methods model one-to-one communication
 - Since SDO remote methods can return values, they can model a pair of one-to-one transmissions, e.g., command and acknowledgement



Use an SDO or a local class?

- Both SDOs and local classes can have state and/or operations
- SDOs are shared among applications
 - Only the publisher can modify its state
 - Subscribers can read the state and get change notifications
 - SDO methods performed by publisher
 - Regardless of where they were invoked
 - SDO instances may or may not have same method implementation
- Local classes are not directly shared between applications
 - Local classes can only be shared as part of an SDO or Message
 - Local class methods performed locally in the app that invoked them
 - Local class methods should have the same method implementation



Use an SDO or a local class? (continued)

- Avoid using a remote method for something that can be done locally
 - Such as converting feet to meters
- A balance can be struck by using a local class as an attribute of an SDO
 - e.g., TENA-Geospatial-PointOfInterest is an SDO that has a TENA-TSPI local class as an attribute



const or variable SDO attributes?

- SDO attributes can be marked const
 - Indicates attribute's value at time SDO was created will not change
- SDO attributes should be marked const whenever feasible and appropriate



Optional or required attributes?

- Attributes can be marked “optional” to indicate that its value may or may not be set
 - TENA-TSPI objects are required to have Time and Position attributes set, but Velocity, Acceleration, etc. are optional
- Use of “magic” values to indicate that a required attribute’s value is not available indicates it has been modeled incorrectly
 - e.g., data producers should not set the velocityInMetersPerSecond attribute to 0, -1, ∞ , etc. if the velocity is not available to them
 - Instead, the velocityInMetersPerSecond attribute should be optional
- Consider using multiple classes related by inheritance instead of optional attributes

Optional attributes vs. inheritance

Modeling Using Optional Attributes:

```
class Shape {  
    float64 area;  
    optional float64 radius;  
    optional uint32 numVertices;  
};
```

- Pattern using optional attributes
 - Can be used wrong
 - Both radius and numVertices can be set simultaneously
 - Subscriber must take care to handle cases where optional attributes not set
 - Requires new OM version to add new shape
 - This likely requires re-deploying existing applications

Modeling Using Inheritance:

```
class Shape { float64 area; };  
class Circle : extends Shape {  
    float64 radius;  
};  
class Polygon : extends Shape {  
    uint32 numVertices;  
};
```

- Pattern using inheritance
 - Can use OM subsetting to add new shape
 - No re-deployment necessary
 - Allows subscribers to indicate “I require radius”



Enforce attribute values be valid whenever possible

- Write local classes and messages to allow implementation to perform run-time checks to ensure values are valid
- Compare these two versions of the same local class
 - First: allows damageInPercent to be set to any number,
 - Such as negative numbers and numbers greater than 100
 - Second: enables an implementation to ensure that the damageInPercent is always between 0 and 100

```
local class FragileThing
{
    float64 damageInPercent;
};
```

```
local class FragileThing
{
    FragileThing(float64 damageInPercent );
    set_damageInPercent(in float64 damageInPercent);
    readonly float64 damageInPercent;
};
```



Summary – think about the nature of things being modeled

- Think about the nature of what is being modeled
 - Persistent or ephemeral?
- Think about the nature of the communication patterns
 - One-to-many or one-to-one?
 - A pair of one-to-ones (e.g., command & acknowledgement)?
- Think about the nature of each attribute of each class
 - Constant or variable?
 - Optional or required?
 - What is the sensible range of valid values?



Associating SDOs

- Use SDO pointer for associations between SDO pairs
 - e.g., a tank having a driver, or an airplane having a missile
- An alternative is to correlate attributes in the SDOs
 - e.g., the tank SDO may have a “nameOfDriver” attribute that is correlated with the driver SDO’s “name” attribute
 - This alternative is generally inferior to an SDO pointer because it is error prone (was the name spelled correctly?) and tends to increase unnecessary network traffic
 - With a tank SDO, you sort through every driver to find just one



How do I get started?

- Think about what information is needed by different publishers and subscribers
- Ask yourself questions from both points of view
 - Publisher Questions
 - What is the key information I need to share with subscribers?
 - Is the data persistent or ephemeral?
 - How often does the data need to be disseminated?
 - Is there information I need from subscribers in order to meet their needs?
 - How many different subscribers care about my information (1, few, many)?
 - Subscriber Questions
 - What data (that I don't have) do I need to perform my job?
 - How often do I need the data updated?
 - Will every attribute value be valid on every update?

How do I get started?

- Write a high-level OM Description
 - Should be a concise description of the OM
 - Indicate what the OM is attempting to address
 - Describe the features the OM will provide
 - Provide additional information that may benefit users
- Example: *TENA-Time OM Description*
 - *The TENA-Time object model will enable the exchange of time data between systems that are not using identical time coordinate systems. A time object will allow its data to be accessed and modified using any of several supported time coordinate systems: Unix, GPS, UTC, etc. Additionally, the TENA Time OM will allow users to use time in their preferred coordinate system without needing knowledge of any other user's time coordinate system choice. Finally, the TENA-Time OM will have an associated implementation to provide for conversion to and from all supported time formats.*



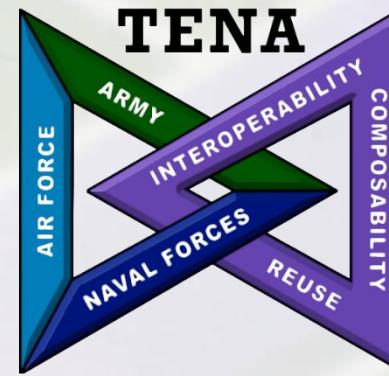
Maximize use of Standard OMs

- Review the Standard OMs, looking for reuse opportunities
 - If an object contains all the data and methods you need, then use it
 - If an object contains most of the data and methods, then consider
 - Extending the TENA object (inheritance)
 - Including the TENA object into another one (containment)
- Designing a new OM from scratch without using any TENA Standard OMs is uncommon
 - Generally, applications need at least one of the following
 - TENA-Time – various time representations
 - TENA-TSPI – time-space-position representations
- Also look at OMs submitted by other user groups for possible reuse opportunities



Summary of OM design guidance

- Model the problem, not one solution to the problem
- Think about the nature of
 - What is being modeled, persistent or ephemeral?
 - The communication patterns, one-to-many or one-to-one?
 - A pair of one-to-ones, e.g., command & acknowledgement?
 - Each attribute of each class
 - Constant or non-constant?
 - Optional or required?
 - What is the sensible range of valid values?
- Maximize the use of Standard OMs



Using cmake with TENA



Using cmake

- Integrating TENA into cmake is similar to integrating other libraries into a cmake-based project
- We'll go over an example that uses the 5 apps developed in the previous exercise
 - Including wix-based packaging for Windows and deb packaging for Ubuntu
- The example provided is “one” way to integrate TENA into a cmake based project – although many other ways exist and may be more appropriate for your projects
- We will focus on TENA specific integration – not everything in the CMakeLists.txt file – which is mostly boiler-plate



Example CMakeLists.txt

- We will go through the example at
`cmake-example/vugapps/CMakeLists.txt`





Cmake version

- We recommend using at least version 3.17

```
cmake_minimum_required(VERSION 3.17)
```



Tell cmake to use C++ 17 flags for compilers

```
set(CMAKE_CXX_STANDARD 17)
```

Environment variables to cmake variables

- You might find it convenient to parse TENA environment variables into cmake variables

```
file(TO_CMAKE_PATH
    "$ENV{TENA_HOME}"
    TENA_HOME)
```

```
file(TO_CMAKE_PATH
    "${TENA_HOME}/$ENV{TENA_VERSION}/include"
    TENA_INCLUDE)
```

```
file(TO_CMAKE_PATH
    "${TENA_HOME}/lib"
    TENA_LIBDIR)
```

```
set(TENA_BUILD
    "$ENV{TENA_PLATFORM}-v$ENV{TENA_VERSION}")
```



TENA requires TENA_PLATFORM defined for build

```
add_compile_definitions(  
    "TENA_PLATFORM=$ENV{TENA_PLATFORM}")
```





Example – specifying include directories (for MW, OMs, OM/impls)

```
set(INCLUDE_DIRS
"${TENA_INCLUDE}"
"${TENA_INCLUDE}/OMs/TENA-exceptions-v1.0.0"
"${TENA_INCLUDE}/OMs/TENA-Time-v2"
"${TENA_INCLUDE}/OMs/TENA-TSPI-v5"
"${TENA_INCLUDE}/OMs/TENA-Hardware-System-v1.0.0"
"${TENA_INCLUDE}/OMs/TENA-Geospatial-TSPIcovariance-v1.0.0"
"${TENA_INCLUDE}/OMs/TENA-Geospatial-PointOfInterest-v1.0.0"
"${TENA_INCLUDE}/OMs/TENA-Instrumentation-Track-State3D-v1.0.0"
"${TENA_INCLUDE}/OMs/VUG-Entity-v0.1.0"
"${TENA_INCLUDE}/OMs/VUG-TrafficLight-v0.3.0"
"${TENA_INCLUDE}/OMs/VUG-Vehicle-v0.6.0"
"${TENA_INCLUDE}/OMs/VUG-TrafficControl-v0.4.0"
"${TENA_INCLUDE}/impls/TENA-Time-v2-StdImpl-v1.1.4"
"${TENA_INCLUDE}/impls/TENA-TSPI-v5-StdImpl-v1.0.4"
"${TENA_INCLUDE}/impls/TENA-Geospatial-TSPIcovariance-v1.0.0-StdImpl-v1.0.1"
"${TENA_INCLUDE}/impls/TENA-Hardware-System-v1.0.0-StdImpl-v1.0.1"
)
target_include_directories(myproject PUBLIC ${INCLUDE_DIRS})
```



Example – specifying library path (for MW, OMs, OM/impls)



```
link_directories(myproject "${TENA_LIBDIR}")
```



Example - finding libraries

```
# Library file prefix and extension are different between Windows and Linux

if (MSVC)
    set (FINDLIB_EXTENSION "${TENA_BUILD}")
    set (TENA_LIB_PREFIX "lib")
endif (MSVC)

if (UNIX)
    set (FINDLIB_EXTENSION "${TENA_BUILD}.so")
    set (TENA_LIB_PREFIX "lib")
endif (UNIX)

# Middleware Library example
find library(LIB_TENA_Middleware
    NAMES "${TENA_LIB_PREFIX}TENA_Middleware-${FINDLIB_EXTENSION}"
    PATHS "${TENA_LIBDIR}" NO_DEFAULT_PATH REQUIRED)

# OM Library example
find library(LIB_TENA_TSPI
    "${TENA_LIB_PREFIX}TENA-TSPI-v5-${FINDLIB_EXTENSION}"
    PATHS "${TENA_LIBDIR}" NO_DEFAULT_PATH REQUIRED)

# OM Library Local class implementation example
find library(LIB_TENA_TSPI_IMPL
    "${TENA_LIB_PREFIX}TENA-TSPI-v5-StdImpl-v1.0.4-${FINDLIB_EXTENSION}"
    PATHS "${TENA_LIBDIR}" NO_DEFAULT_PATH REQUIRED)

## (find library for every needed OM and OM Local class implementation)
```



Example – target link libraries

```
set(LIB_THREAD "")  
if (UNIX)  
    set(THREADS_PREFER_PTHREAD_FLAG ON)  
    find_package(Threads REQUIRED)  
    set(LIB_THREAD "Threads::Threads")  
endif (UNIX)  
set(LIBS  
    ${LIB_TENA_Middleware}  
    ${LIB_TENA_TSPI}  
    ${LIB_TENA_TSPI_IMPL}  
    # ... (see example CMakeLists.txt for complete example)  
    ${LIB_THREAD})  
target_link_libraries(myproject ${LIBS})
```



Example - package library dependencies

```
set(LIB_EXTENSION "${FINDLIB_EXTENSION}.dll")
if (UNIX)
    set(LIB_EXTENSION "${FINDLIB_EXTENSION}")
endif (UNIX)

install(FILES
    "${TENA_BIN}/${TENA_LIB_PREFIX}TENA_Middleware-${LIB_EXTENSION}"
    "${TENA_BIN}/${TENA_LIB_PREFIX}TENA-TSPI-v5-${LIB_EXTENSION}"
    "${TENA_BIN}/${TENA_LIB_PREFIX}TENA-TSPI-v5-StdImpl-v1.0.4-${LIB_EXTENSION}"
    # ... (see example CMakeLists.txt for complete example)
    DESTINATION ${LIB_INSTALL_DIR}
)
```



Packaging

- You may wish to include the TENA platform in the package (installer) file name

```
set(CPACK_PACKAGE_VERSION ${PROJECT_VERSION})
```

```
set(CPACK_PACKAGE_FILE_NAME  
    "${CMAKE_PROJECT_NAME}-v${PROJECT_VERSION}-${TENA_BUILD}")
```



Demo

- Quick review of CMakeLists.txt
- Generate project files with cmake
- Build project apps
- Package apps
 - Ubuntu: DEB file
 - Windows: MSI file
 - Red Hat / CentOS: RPM file
- Install package
- Run installed apps



cmake example, Ubuntu 20.04

```
# Install cmake, and dpkg-dev if you want to package deb files
sudo snap install cmake -classic
sudo apt install dpkg-dev

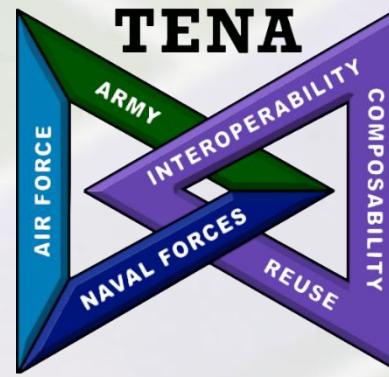
# Navigate to the class pack directory
cd ~/classpack
cd cmake-example/vugapps

# Prepare a directory for building in, and run cmake
mkdir build
cd build
cmake ..

# Build, package, and install the project
make
make package
sudo dpkg -i vugapps-v0.0.1-u2004-gcc9-64-v6.0.9.deb

# Run the installed apps
app1
app2
app3
app4
app5

# Remove the installed package
sudo dpkg -r vugapps
```



Using VS code with TENA



Some VS-Code pointers

- Microsoft Visual Studio (VS) Code is not required to build TENA apps
 - However, we'll provide a few pointers for folks that use VS-Code
- VS-Code with C++ Tools for IntelliSense, etc.
 - Tell VS-Code where to find TENA header files
 - Set TENA_PLATFORM preprocessor directive



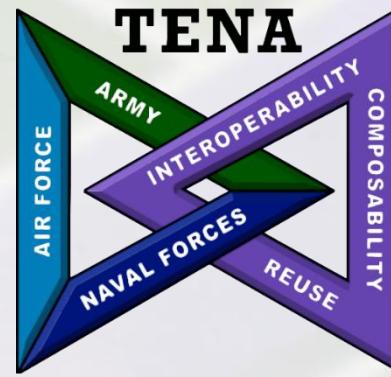
C/C++ for VS Code, IntelliSense

- In settings.json...
 - Or project specific settings file (.vscode/c_cpp_properties.json)
 - Tell IntelliSense where to look for header files

```
"C_Cpp.default.includePath":  
[  
    "${TENA_HOME}/${TENA_VERSION}/include/**",  
    "${workspaceFolder}/**"  
]
```

- Tell IntelliSense what “TENA_PLATFORM” to use

```
"C_Cpp.defaultdefines":  
[  
    "TENA_PLATFORM=${TENA_PLATFORM}"  
]
```

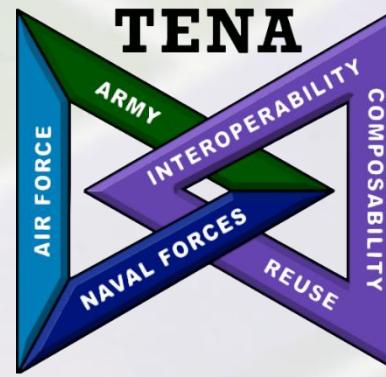


Backup topics



Backup topics

- TDCS & DataView
- Middleware configuration
- Advanced filtering
- Middleware IDs
- Middleware metadata
- Handling exceptions
- Sending alerts
- Log files
- TRMC website services



TDCS & DataView



A note on documentation

- We will just cover some basics
 - Refer to documentation for details
- Documentation:
 - <https://www.trmc.osd.mil/wiki/display/TDCS/Documentation>
- Helpdesk:
 - <https://www.trmc.osd.mil/helpdesk/projects/TDCS>

The screenshot shows a web browser window with the title "Documentation - TENA: Data C X". The URL in the address bar is <https://www.trmc.osd.mil/wiki/display/TDCS/Documentation>. The page content is organized into sections:

- TENA Data Collection System Documentation**
 - **TENA Data Collection System User Guide** – Explains the operation of the TENA Data Collection System product.
 - **Release Notes** – Provides details on the fixes, improvements, and known defects for each TENA Data Collection System release.
 - **TDCS Database Schema for v1.1.3 & v1.1.4 tools (MW 6.0.7)**
 - Details the database schema current version
 - **TDCS Database Schema for v1.1.5+ tools** - Details the database schema for the next version
 - **TDCS Versions and schemas** - Details the release dates and database versions.- Post-Event Metrics Analyzer Documentation**
 - **Post-Event Metrics Analyzer User Guide** – Explains the operation of the Post-Event Metrics Analyzer product
 - **Post-Event Metrics Analyzer Release Notes** – Provides details on the fixes, improvements, and known defects for each Post-Event Metrics Analyzer release.
- TENA DataView Documentation**
 - **TENA DataView User Guide** - Explains the operation of the TENA DataView Tool
 - **TENA DataView Release Notes** – Provides details on the fixes, improvements, and known defects for each TENA DataView release

The browser interface includes a navigation bar with back, forward, and search functions, as well as a sidebar with icons for spaces, create, and search.



Presentation/demo overview

- Part 1: TDCS basics
 - What is TDCS?
 - Acquiring & installing TDCS
 - TDCS demonstration
- Part 2: Using DataView with TDCS
 - What is DataView?
 - Acquiring & installing DataView
 - DataView demonstration



Part 1: TDCS basics

- What is TDCS?
- Acquiring & installing TDCS
- TDCS Demonstration



What is TDCS?

- TDCS is used for collecting & playing back TENA data
 - Data can also be monitored & exported to other formats
 - TDCS is distributed PER OBJECT MODEL (OM)
- What does it include?
 - Data collector: records data for a specific OM
 - Playback tool: plays back data for a specific OM
 - dbExporter: converts DB data to a spreadsheet
 - dbConvert: converts DBs from previous versions to latest version
 - We do not plan to demo dbConvert in this presentation
- Why should I use it?
 - Collecting, replaying, and exporting data



TDCS data collector

- A subscriber that stores SDO events and Messages in a database
- Built for a specific OM
 - e.g., TENA-LVC-Engagement v1.1.0



TDCS playback tool

- A publisher that plays back SDO events and Messages from a database
- Built for a specific OM
 - e.g., TENA-LVC-Engagement v1.1.0



TDCS DB exporter

- Generates a spreadsheet in XML format from a database
 - Filters data based on time or SDO/Message type
 - Data is split between worksheets
 - Separate worksheet for each SDO & Message type
 - Additional worksheets for vector data
 - Spreadsheet can be opened in Microsoft Excel
- Built for any database created by a data collector



Acquiring & installing TDCS

- Requesting/downloading TDCS
 - Based on a particular Object Model (OM)
- Installation



Acquiring TDCS

- Determine which OM you are using
 - What if I have multiple OMs?
 - You will need an OM that “includes” the OMs that you would like to collect data for
 - For our “TDCS basics” example slides and demonstration, we will use the TENA-LVC-Engagement-v1.1.0 OM
- Download the “Data Collection System” from OM repository page
- Let’s get TDCS for the TENA-LVC-Engagement v1.1.0 OM...

Home - TRMC - TRMC Website +

This website is for process, or transmit any CLASSIFIED information.

Spaces Create ... Search ?

Dashboard Edit Save for later Watch ...

Home

Test Resource Management Center

 **JMETC**

Joint Mission Environment Test Capability

The JMETC mission is to provide a persistent capability for linking distributed facilities, enabling DoD customers to develop and test warfighting capabilities in a Joint Context. JMETC provides a test infrastructure consisting of the components necessary to conduct Joint distributed test events by cost-effectively integrating live, virtual, and constructive (LVC) test resources that are configured to support the users' needs. The JMETC program provides its

 **TENA**

Test and Training Enabling Architecture

The United States Office of the Secretary of Defense Test Resource Management Center (TRMC) has developed a common architecture to support effective integration and reuse of testing, training, and simulation capabilities that require real-time collaboration between distributed computer systems operating within diverse testing and training environments. Through the establishment of the Test

 **National Cyber Range Complex**

The National Cyber Range Complex (NCRC) is a holistic, integrated cyber range capability comprising the National Cyber Range (NCR), Regional Service Delivery Points (RSDP), and the Joint Mission Environment Test Capability (JMETC) Multiple Independent Levels of Security (MILS) Network (JMN). The NCRC provides Department of Defense (DoD) and other government and industry users with representative environments, a distributed network infrastructure and other tools and services to support

Click on the TENA link

Home - TENA - TRMC Website +

https://www.trmcosd.mil/wiki/display/TENA/Home

This website is for UNCLASSIFIED USE ONLY. Do not discuss, enter, transfer, process, or transmit any CLASSIFIED information.

Spaces Create ... Search ? 

Dashboard  Edit Save for later Watch ...

Test and Training Enabling Architecture

TENA News & Events

- JMETC Tech Talks will begin in June - visit the JMETC Tech Talks page to see upcoming talks
- Proceedings & helpdesk cases for previous JTEX meetings (JTEX-01 thru -05)
- "Bringing Speed to DoD Cybersecurity: Applying Best Practices for Agile RDT&E Environments" (ITEA Journal, Dec 2020)
- JMETC Brings Readily-Available Persistent Connectivity for Joint Distributed Test Events (2020-09-03)
- JMETC Provides Persistent Connectivity and Support to Enable Air Force System Interoperability Test (AFSIT) (2019-11-14)
- JMETC Support of Multiple Joint Interoperability Tests (JITs) (2019-10-29)
- TENA Overview Briefing (2019-10-15)
- TRMC Big Data Knowledge Management (BDKM) Enterprise Overview Briefing (2019-08-01)
- JMETC Overview Briefing (2019-06-05)
- TENA Middleware 6.0.7 released, added computer platform support and made various improvements (2019-05-13)

TENA-team members can add news items on the [News](#) wiki page.

Favorite User Groups

- TENA: Utilities and Tools   



Click on the TENA Repository link



TENA Repository

Access the TENA repository for browsing and building object models.



TENA Project Information

Access TENA project and project related information.



TENA Middleware

Obtain information concerning the TENA Middleware.



TENA Training

Obtain information concerning TENA training.



Object Models

Obtain information about TENA Object Models.



TENA Helpdesk

Access the TENA Helpdesk to submit issues or search for solutions.



TENA User Profile

Review and update your user profile.

TENA - Test and Training Enabling Architecture

TENA Website Repository Home Components Help

Welcome
The TENA Repository is a web-based storehouse for TENA interoperability solutions.



TENA Repository
Discover, reuse and share
TENA components

Click on the “All products and Object Models” link

Getting Started			
Quick Start Guide Information describing how to use the TENA Repository.	All Products and Object Models Browse products and object models from all groups.	TENA Standard Object Models A Collection of C++ SDKs for the TENA Standard Object Models	TENA Canary The TENA Canary is a simple application that can connect to an execution to help test the network.
Featured Products and Object Models Featured Products and Object Models	Submit Object Model Submit or revise an object model	TENA Middleware The TENA Middleware is a software framework that enables rapid and reliable development of scalable distributed systems.	TENA Console The TENA Console is a tool that assists in the troubleshooting and monitoring of TENA executions.

3.2.5.2trunk

Products and Object Models in PMRF		
REPO Products and Object Models in REPO	RMAST Products and Object Models in RMAST	ProLogic Products and Object Models in ProLogic
RTTC Products and Object Models in RTTC	RelayNode Products and Object Models in RelayNode	RTC Products and Object Models in RTC
SATIN Products and Object Models in SATIN	SIMDIS Products and Object Models in SIMDIS	SAIC Products and Object Models in SAIC
STAT Products and Object Models in STAT	SUMS Products and Object Models in SUMS	SPAWAR Products and Object Models in SPAWAR
TA Products and Object Models in TA	TACE Products and Object Models in TACE	Simmetry Products and Object Models in Simmetry
TDCS Products and Object Models in TDCS	TCS Products and Object Models in TCS	
TRCE Products and Object Models in TRCE	TSPIL Products and Object Models in TSPIL	TENA Products and Object Models in TENA
USNTTR Products and Object Models in USNTTR	VENGEANCE Products and Object Models in VENGEANCE	TVDS Products and Object Models in TVDS
WSMR Products and Object Models in WSMR	WebBinding Products and Object Models in WebBinding	VPEF Products and Object Models in VPEF
YTC Products and Object Models in YTC	tmp Products and Object Models in tmp	Weibel Products and Object Models in Weibel

Navigate to the group that your OM is in

TENA Repository: Search Components			
https://www.trmc.osd.mil/wiki/plugins/repository.action#Home?type=Components&section=Search&page=1/group=TENA&componentType=[Product ObjectModel]&platform=source&language=en&sortOrder=1 190%			
Object Model	Description	Version	Released
TENA-Instrumentation-Track-State3D-v1.0.0	This file contains the definition of and documentation for the TENA::Instrumentation::Track::State3D SDO	ObjectModel	Released Public
TENA-Instrumentation-Weather-Station-v1.0.0	This file contains the definition of and documentation for the TENA::Instrumentation::Weather::Station SDO used to report weather measurements	ObjectModel	Released Public
TENA-LVC-Common-v1.0.0	This file contains the definition and documentation for the TENA::LVC::DeadReckoning local class	ObjectModel	Released Public
TENA-LVC-Engagement-v1.1.0	Navigate to the OM that you are interested in	ObjectModel	Released Public
TENA-LVC-Entity-v1.1.0	This file contains the definition and documentation for the TENA::LVC::Entity classes and related items	ObjectModel	Released Public
TENA-LVC-EntityAppearanceCapabilities-v1.0.0	This file contains the definition of the TENA::LVC::EntityAppearance and TENA::LVC::EntityCapabilities local classes	ObjectModel	Released Public
TENA-LVC-EntityType-v1.0.0	This file contains the definition and documentation for the TENA::LVC::EntityType Local Class	ObjectModel	Released Public
TENA-LVC-IDs-v1.0.0	This file contains the definition and documentation for various local classes used for identification purposes, such as the TENA::LVC::EntityID, TENA::LVC::TrackID, TENA::LVC::Engagement::ResultsEventID, and the generic TENA::LVC::EventID	ObjectModel	Released Public
TENA-LVC-RadarSystem-v1.1.0	This file contains the definition and documentation for the LVC radar system classes and measurements	ObjectModel	Released Public

TENA-LVC-Engagement-v1.1.0 (Obj) +

<https://www.trmc.osd.mil/wiki/plugins/repository.action#Details/TENA,LVC-Engagement,1.1.0,.ObjectModel.source.all>

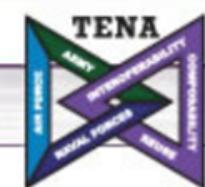
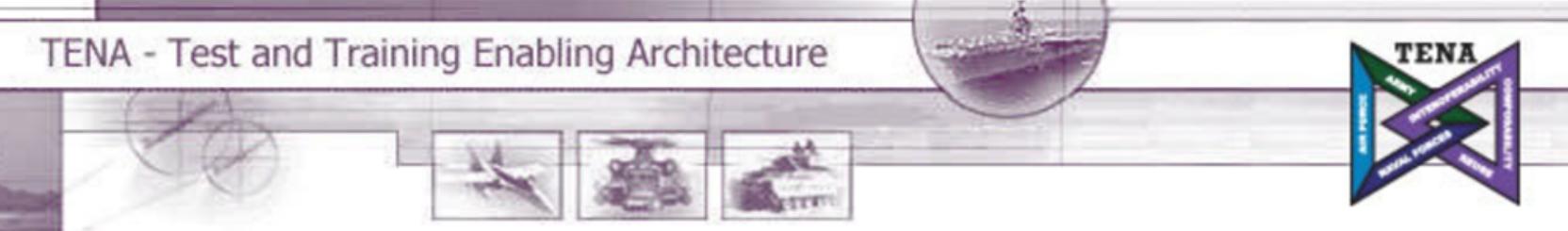
190%

TENA - Test and Training Enabling Architecture

Components

Help

TENA Website Repository Home Components Help



[Repository Home](#) \ [Browse Components](#) \ [TENA](#) \ LVC-Engagement (ObjectModel)



TENA-LVC-Engagement-v1.1.0

Versions

[Download](#)

[Upload New Version](#)

Click the “Download” link

TDL

C++ Binding

[Actions](#): Public

Imports: [TENA-Geospatial-PointOfInterest-v1.0.0](#)

[Create Helpdesk Case](#)

[Middleware](#): 6.0.7

6.0.6

6.0.5.1

(...8 more...)

Java Binding

.NET Binding

[TENAs](#): [Public](#)

Imports: [TENA-Geospatial-TSPLcovariance-v1.0.0](#)

[TENA-Hardware-System-v1.0.0](#)

(10 more...)

Data Collection System

[Supersedes](#)

Then click the “Data Collection System” option

Web Binding

RelayNode

Mapping Spreadsheets

[TDL](#) | [Documentation](#) | [History](#)

/C-Engagement-v1.1.0.tdl

```
2  /**
3   * This file contains the definition of and documentation for the TENA::LVC::Engagement classes and related items.
4   */
5   /**
6    * This object model (OMs) is intended to support the modeling of engagements as a result of the firing of simulated
```

Middleware version

Please choose a middleware version

Component: TENA-LVC-Engagement-v1.1.0

Distribution: Data Collection System

Middleware Version:

- 6.0.7
- 6.0.6
- 6.0.5.1
- [\(...8 more...\)](#)

State: Released Restrictions:

Updated:

Point of Contact: [TENA](#)

```
1  /// \file TENA-  
2  ///  
3  /// This file c  
4  ///  
5  /// This object  
6  /// weapons and  
7  ///  
8  /// \attention This file was developed under various United States Government contracts. The United States Government
```

Click the “Next” button

Download Component: REPO-Dat... +

https://www.trmc.osd.mil/wiki/plugins/repository.action#Process/TENA.Download.1.process.noarch.all/component=TENA.LVC-Engagement.1.1.0.ObjectModel,noarch.all&autoSelect=*,LVC

TENA Website Repository Home \ Brows

Repository Home \ Brows

TENA-L

Download Upload No

State: Released

Updated:

Point of Contact: [TENA](#)

1 //file TENA-
2 // This file c
3 // This object
4 // weapons and
5 // attention This file was developed under various United States Government contracts. The United States Government

Platform Selection

Please choose a platform

Component: TENA-LVC-Engagement-v1.1.0

Distribution: Data Collection System

Middleware Version: 6.0.7

Please choose your operating system:

Windows

Which flavor of the operating system?:

Windows 10

Which compiler?:

Visual Studio 2019

64-bits?

Mode:

Optimized Version

6.0.7
6.0.6
6.0.5.1
[\(...8 more...\)](#)

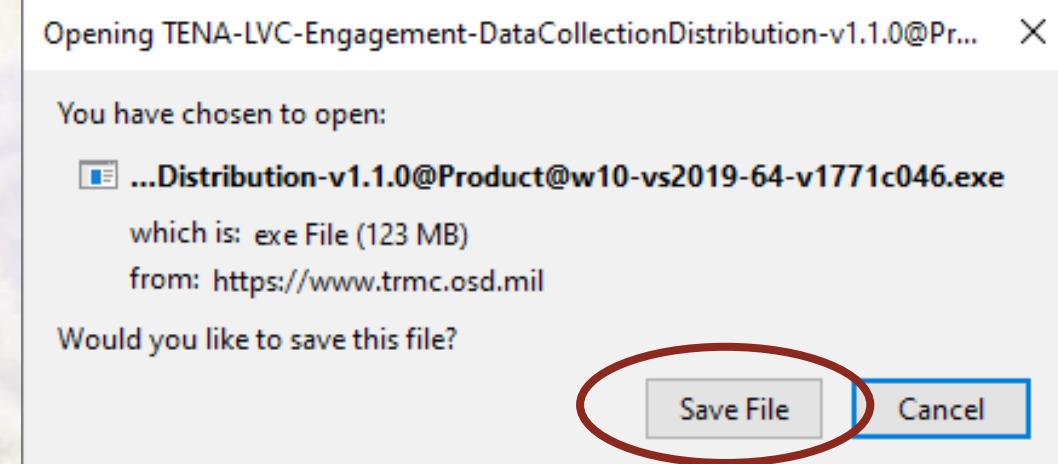
Select operating system & other options

Back Next Cancel Finish

Then click the “Next” button

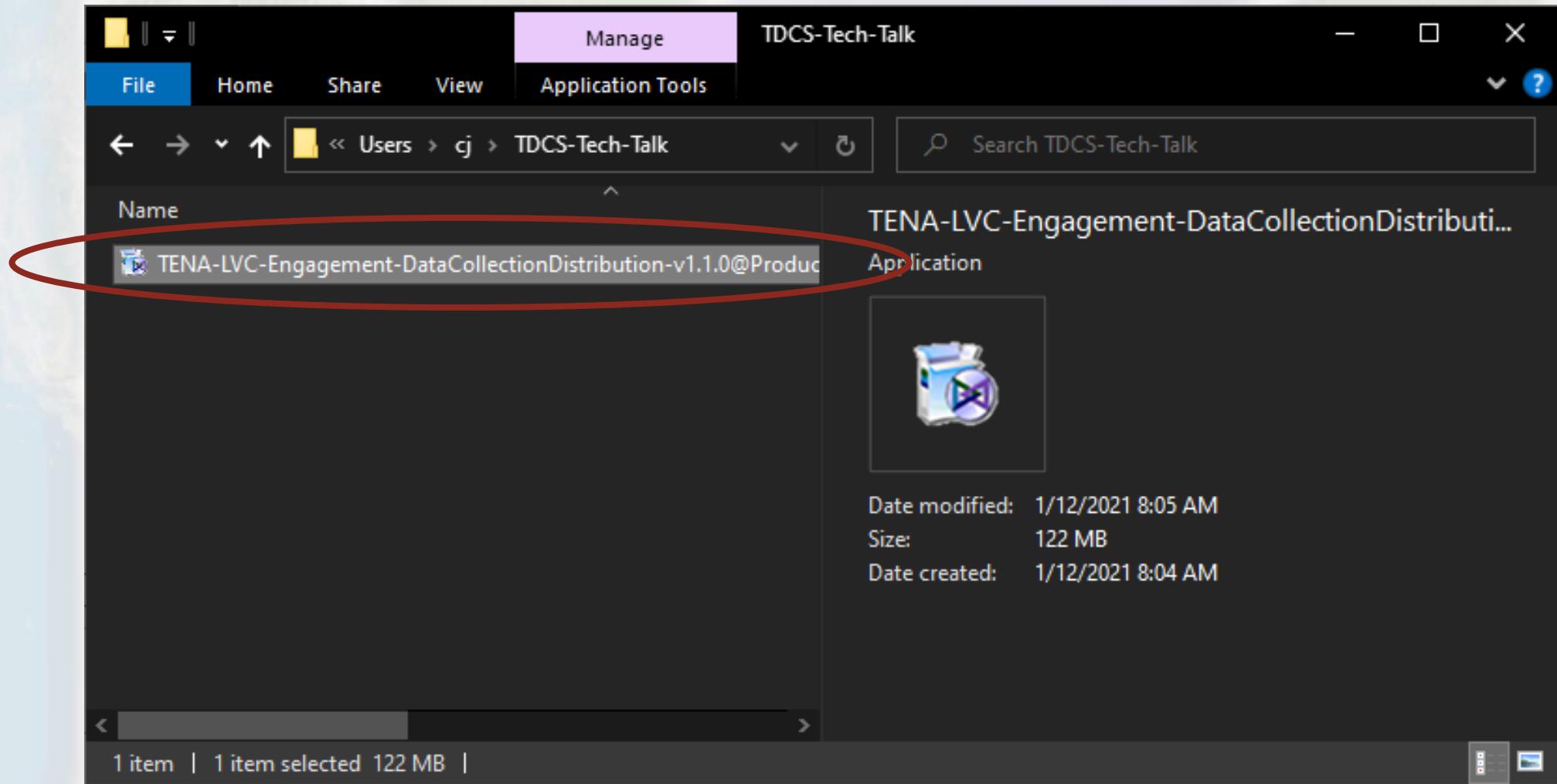


Download the installer

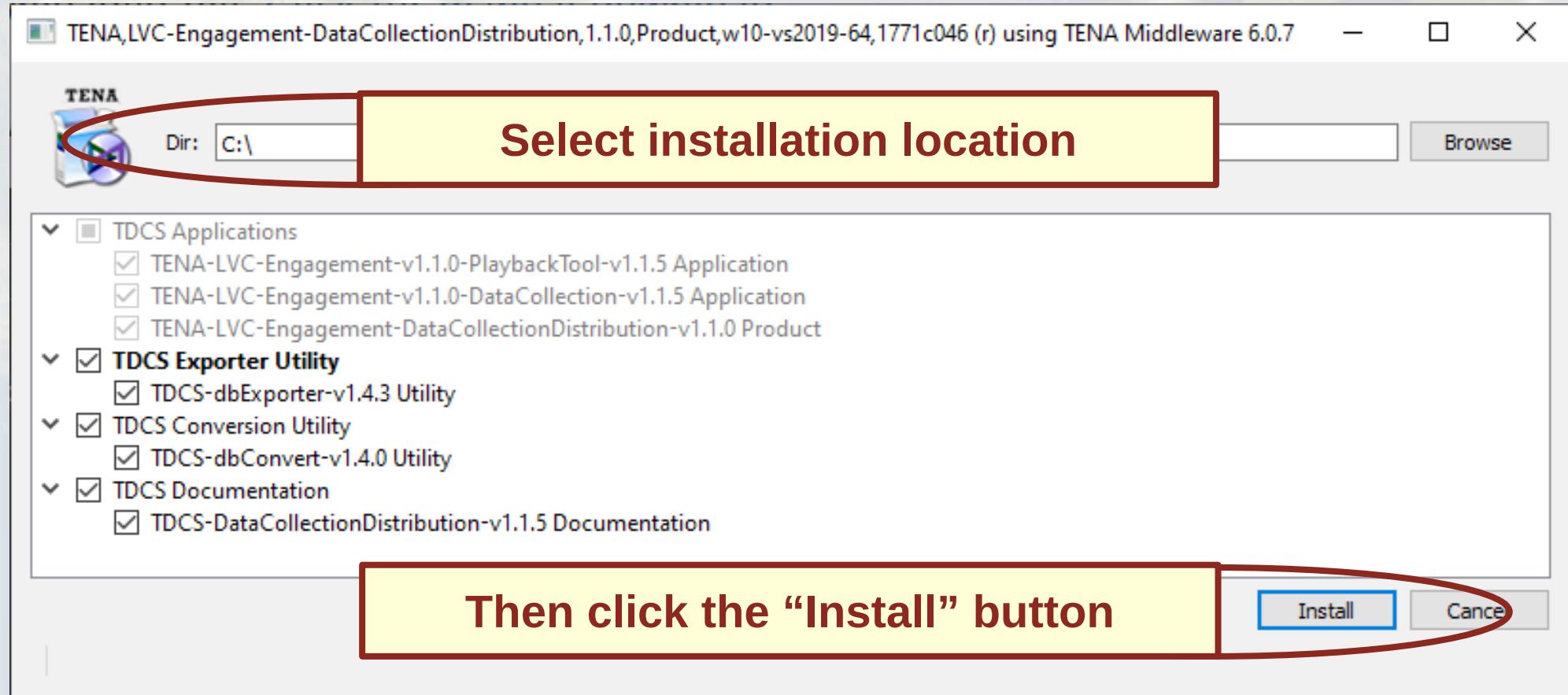




Run the installer

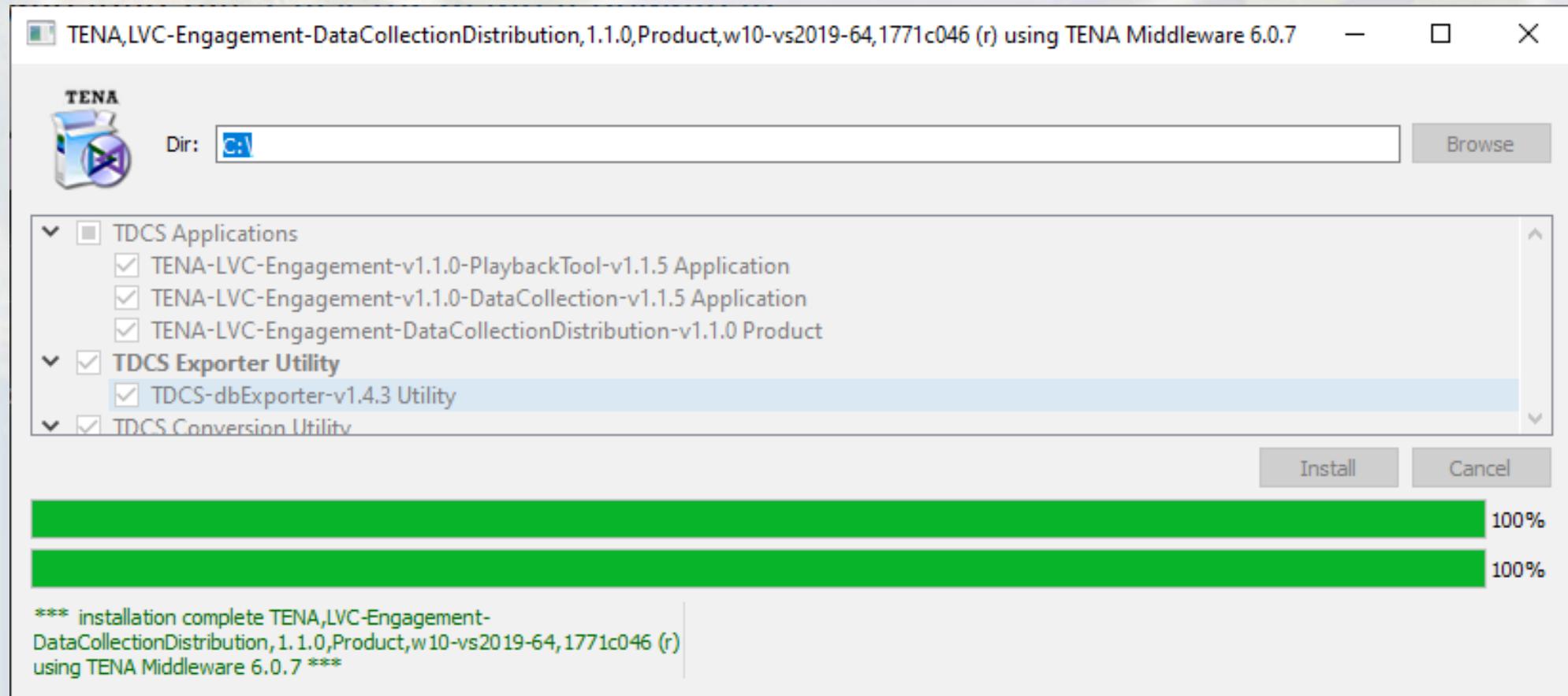


Installer





Installer will close when done





Four apps have been installed

- An OM-specific data collector
 - \$TENA_HOME/
LVC-Engagement-v1.1.0-DataCollection-v1.1.5
- An OM-specific playback tool
 - \$TENA_HOME/
LVC-Engagement-v1.1.0-PlaybackTool-v1.1.5
- An OM-generic dbExporter tool
 - \$TENA_HOME/dbExporter-v1.4.3
- An OM-generic dbConvert tool
 - \$TENA_HOME/dbExporter-v1.4.0



A note on GUI versus CLI

- There are two modes of operation for the data collector and playback tool
 - Graphical User Interface (GUI) is default
 - Command Line Interface (CLI) is optional
- Suppress GUI using the “-nogui” option
 - All configuration performed on command line
- We will use GUI mode for screenshots and demonstration



TDCS demonstration

- Start TENA EM from Console
- Start SIMDIS TENA plugin for data visualization
- Start the TDCS Data Collector
- Publish & collect some TENA data
 - View in SIMDIS (original data)
- Stop the TDCS Data Collector and publishers
- Start the TDCS Playback Tool
 - View in SIMDIS (recorded data)
- Export data to spreadsheet
 - View in Microsoft Excel



TENA Console & EM started

TENA Console v1.0.23, Connected to EM 0 at 10.1.10.200:55100

File View Help

EM Running Execution Execution OM Stats Applications Network Monitoring Execution Managers Consoles Alerts *

Keep Active Apps Selected Hide Resigned/Terminated Apps Force Remove Remove from Execution

ID	App Name	Hostname	IP Address	Port	MW Version	Platform	Joined	Resigned	Status	+ +
No apps have joined yet										

Application: n/a

Configuration Application Details Heartbeat History OM Stats

Hide Default Values Filter: * No Filter * ▾

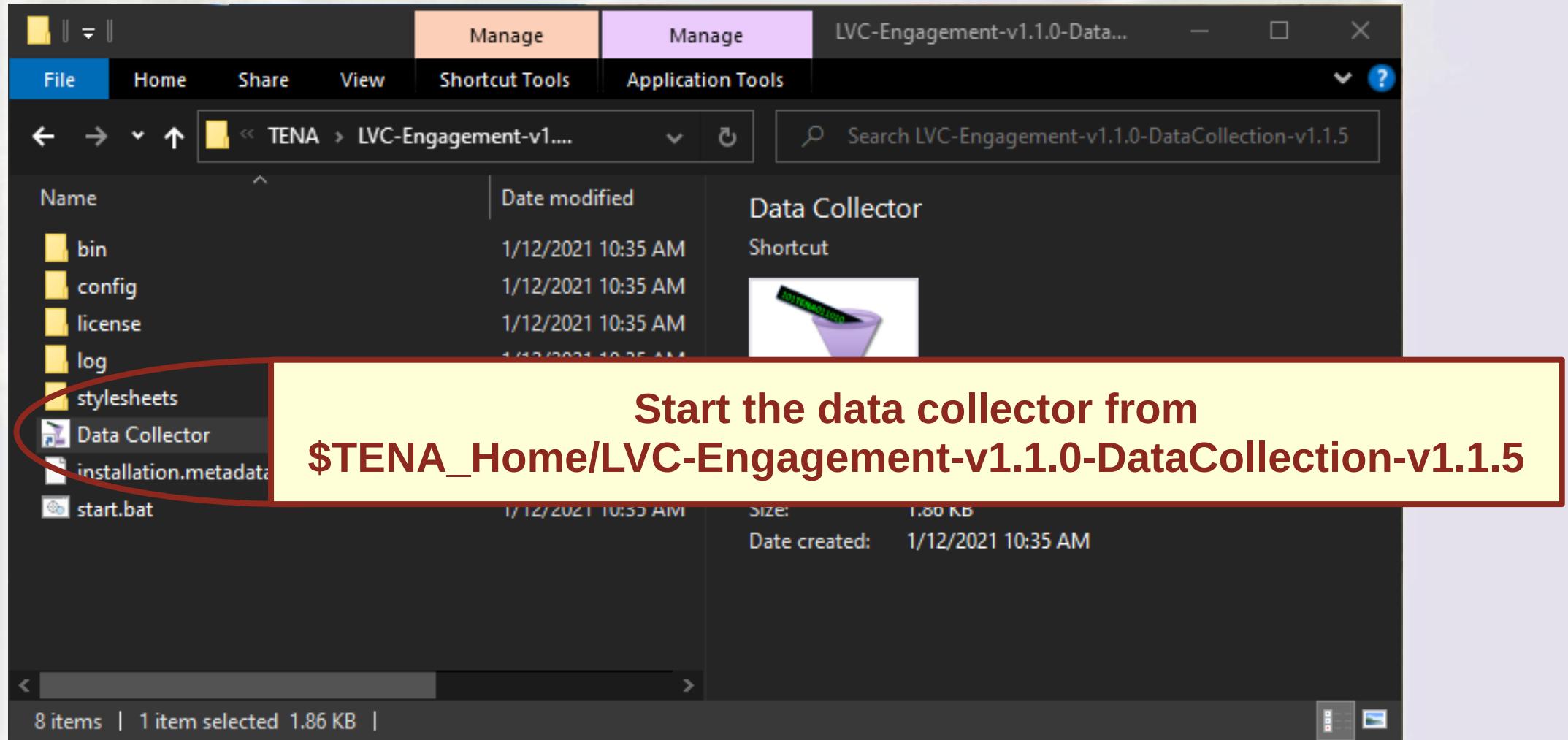
ID	Option	Value	Source	Parameter Type	+ +



We will use SIMDIS for viewing SDOs and Messages

Time: 001 1970 00:00:00.000
Time Step: 1.000 Secs
Distance: 6500000.0 m
Azim: 0.0
Elev: 90.0
Centered: None

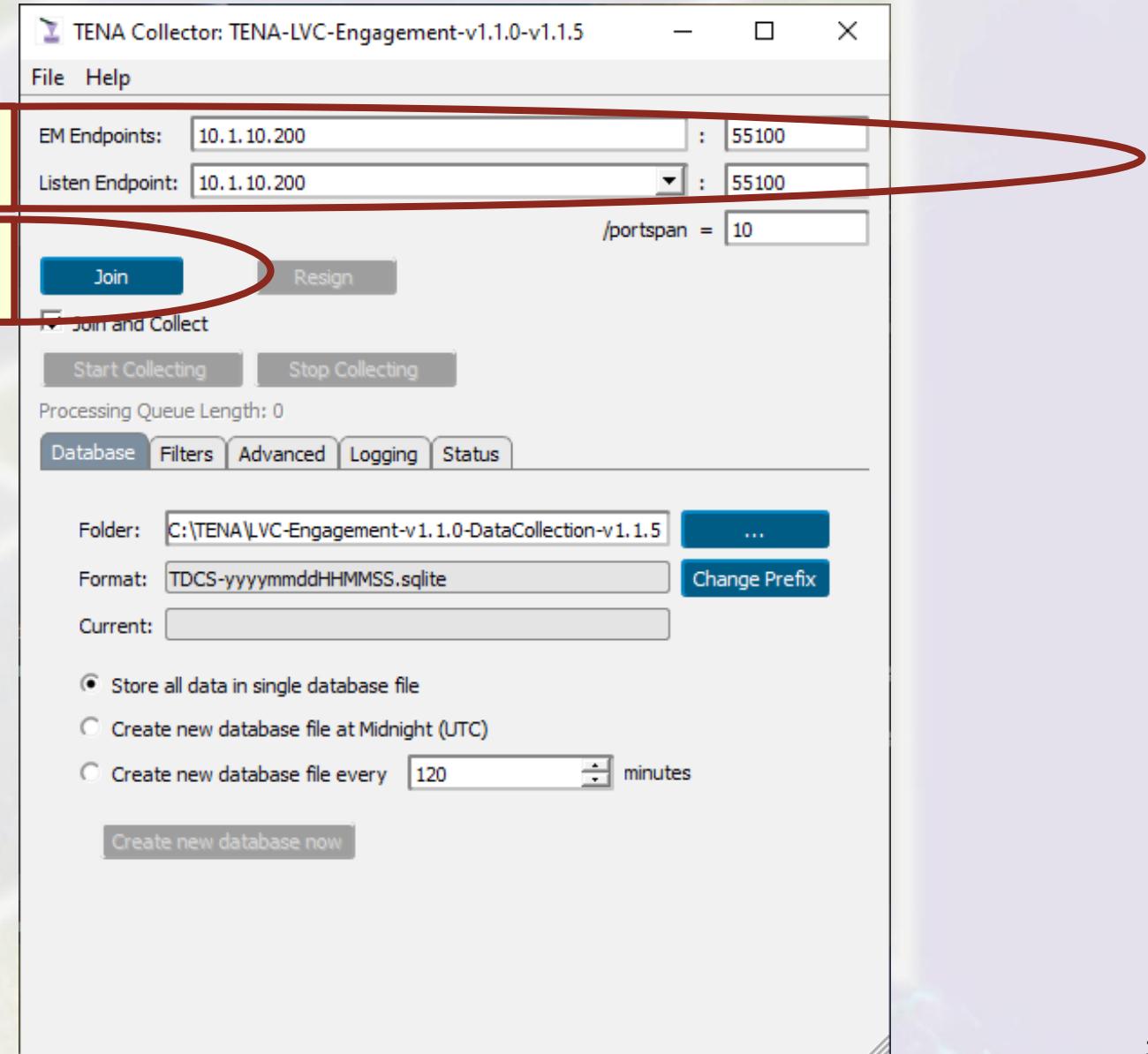
Data collector shortcut



Data collector

Set listen endpoints & EM endpoints

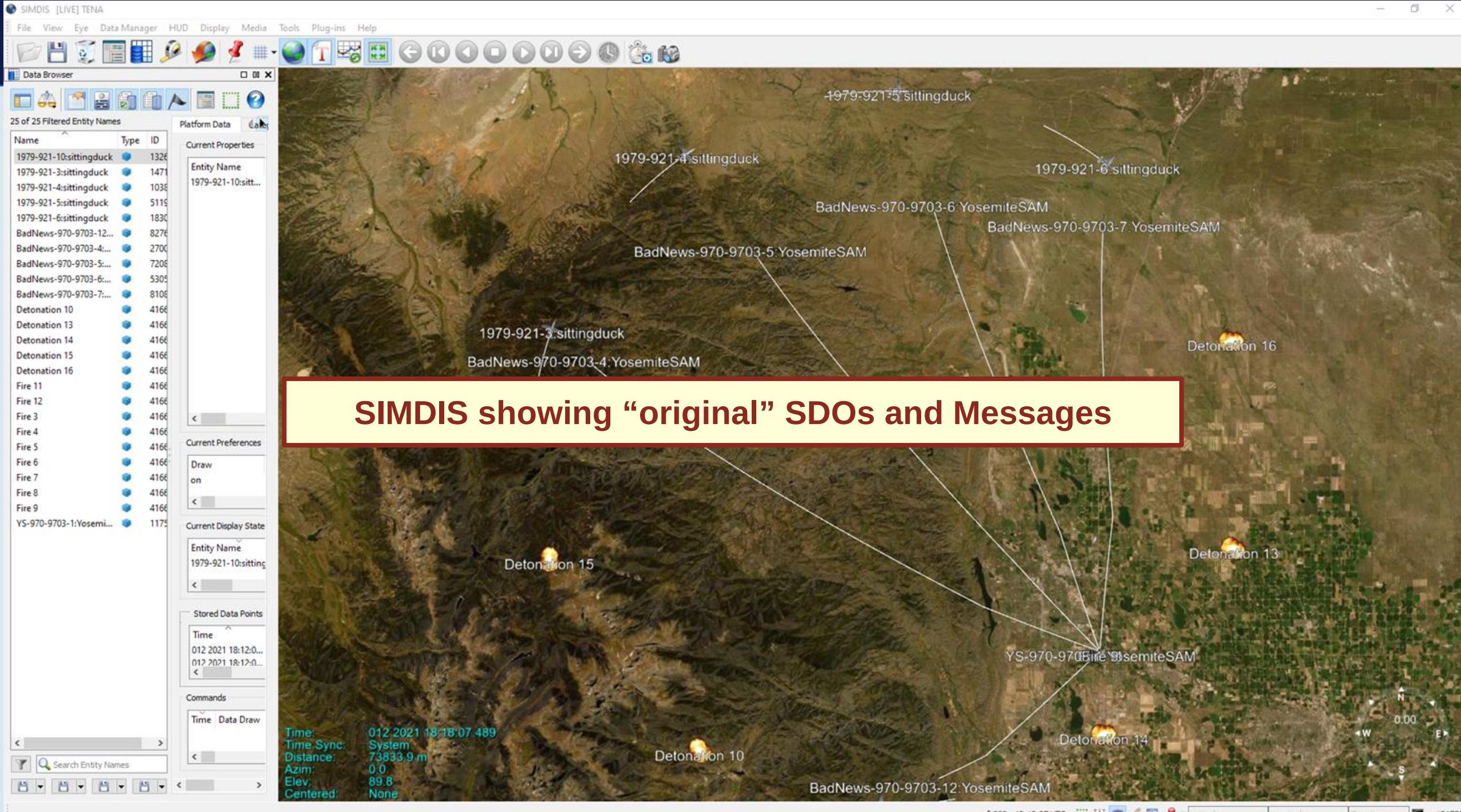
Click the “Join” button





Our test scenario

- A-10 Warthogs fly over northern Colorado
 - Demo app: SittingDuck
 - Publishes
 - TENA-LVC-Entity SDOs for the A-10 Warthogs (Implements setDamageState() RMI)
 - Subscribes to
 - TENA-LVC-Engagement-MunitionDetonation messages
- SAM site fires missiles at the Warthogs
 - Demo app: YosemiteSAM
 - Publishes
 - TENA-LVC-Entity SDOs for the SAM site & missiles
 - TENA-LVC-Engagement-MunitionFire & MunitionDetonation messages
 - Subscribes to
 - TENA-LVC-Engagement-MunitionDetonation messages
- Viewed in SIMDIS & recorded by TDCS





Data collector in Console

TENA Console v1.0.23, Connected to EM 0 at 10.1.10.200:55100

File View Help

EM Running Execution Execution OM Stats Applications Network Monitoring Execution Managers Consoles Alerts *

Keep Active Apps Selected Hide Resigned/Terminated Apps Force Remove Remove from Execution

ID	App Name	Hostname	IP Address	Port	MW Versi...	Platform	Joined	Resign...	Status	+/-
2	SIMDIS10 10.0 1.28.0		10.1.10.2...	55102	6.0.7	w10-vs...	01/12/21 1...		OK	
3	"GittingDuck		10.1.10.2...	55103	6.0.7	w10-vs...	01/12/21 1...		OK	
6	TENA-LVC-Engagement-v1.1.0-DataCollection-v1.1.5		10.1.10.2...	55104	6.0.7	w10-vs...	01/12/21 1...		OK	
7	yosemiteam.exe		10.1.10.2...	55105	6.0.7	w10-vs	01/12/21 1...		OK	

Application: # 6, tenaCollector-TENA-LVC-Engagement-v1.1.0-v1.1.5

Configuration Application Details Heartbeat History OM Stats

Refresh OM Stats Last Refreshed at 01/12/21 11:20:36.348-0700 Totals for All Sessions

ID	Session	Type Name	Type	Serv... Create...
6	TENA-LV...	TENA::Hardware::System	SDO		0	0	0	0	0	0	-	-	-
6	TENA-LV...	TENA::Instrumentation::Track::State3D	SDO		0	0	0	0	0	0	-	-	-
6	TENA-LV...	TENA::LVC::Entity	SDO		0	0	0	17	5361	0	-	-	-
6	TENA-LV...	TENA::LVC::Track	SDO		0	0	0	0	0	0	-	-	-

Using Console to get collector OM stats

Joined & collecting data

TENA Collector: TENA-LVC-Engagement-v1.1.0-v1.1.5

File Help

EM Endpoints: 10.1.10.200 : 55100

Listen Endpoint: 10.1.10.200 : 55100

/portspan = 10

Join and Collect

Processing Queue Length: 0

Joined, collecting data

OM Type	Event Count
TENA::Geospatial::PointOfInterest	0
TENA::Hardware::ControllableSystem	0
TENA::Hardware::ControllerInterface	0
TENA::Hardware::System	0
TENA::Instrumentation::Track::State	0
TENA::LVC::Engagement::Adjudication	0
TENA::LVC::Engagement::MunitionDetonation	10
TENA::LVC::Engagement::MunitionFire	10
TENA::LVC::Engagement::MunitionFireRequest	0
TENA::LVC::Engagement::Results	0
TENA::LVC::Entity	11629
TENA::LVC::Track	0

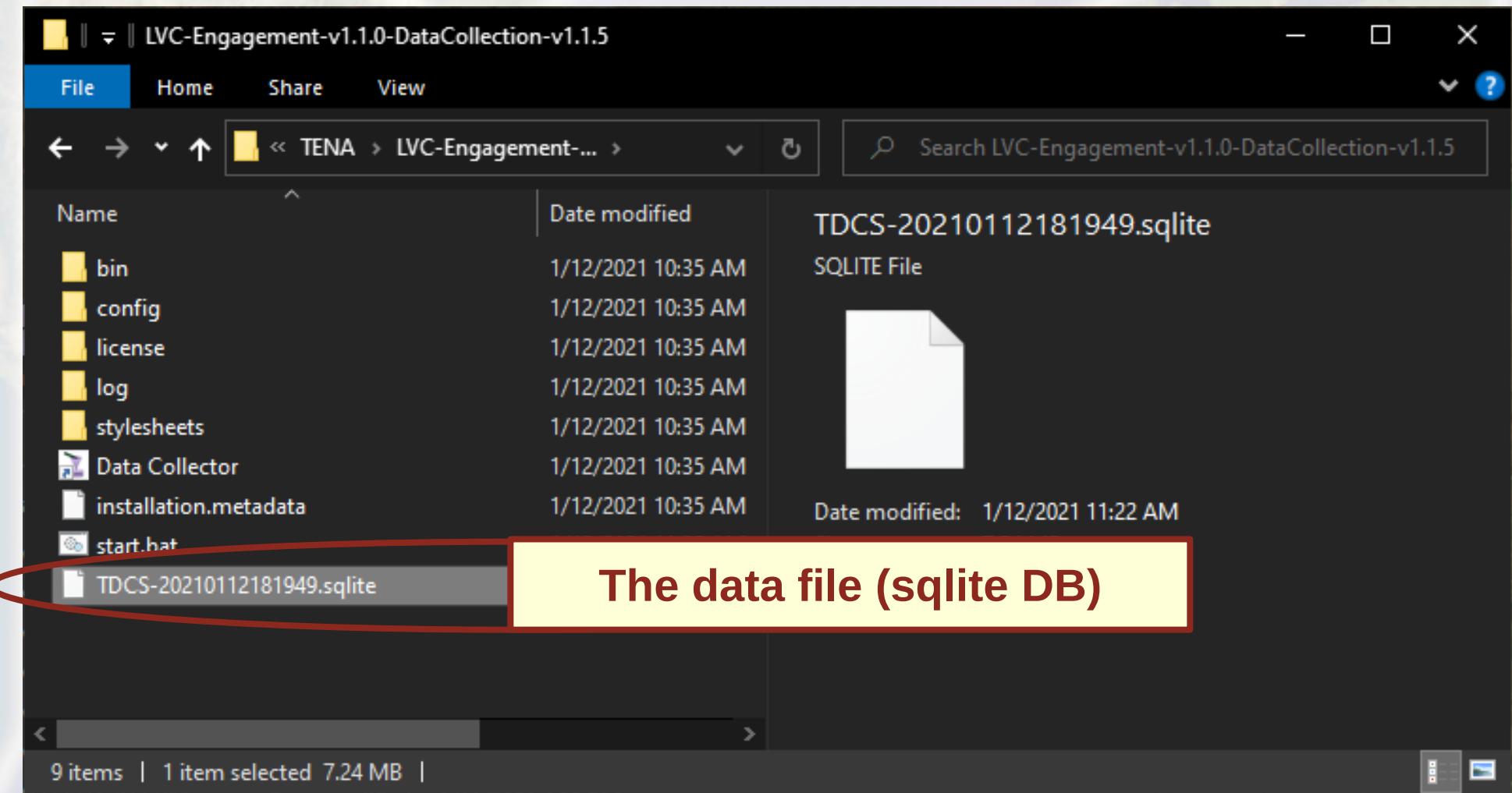
Collecting Data...

(OM specific)

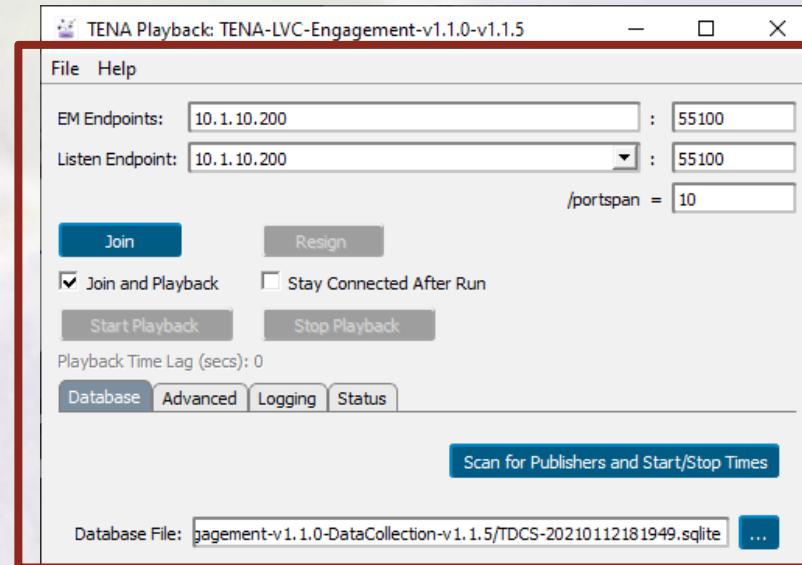
Event counts in collector

TDCS data file

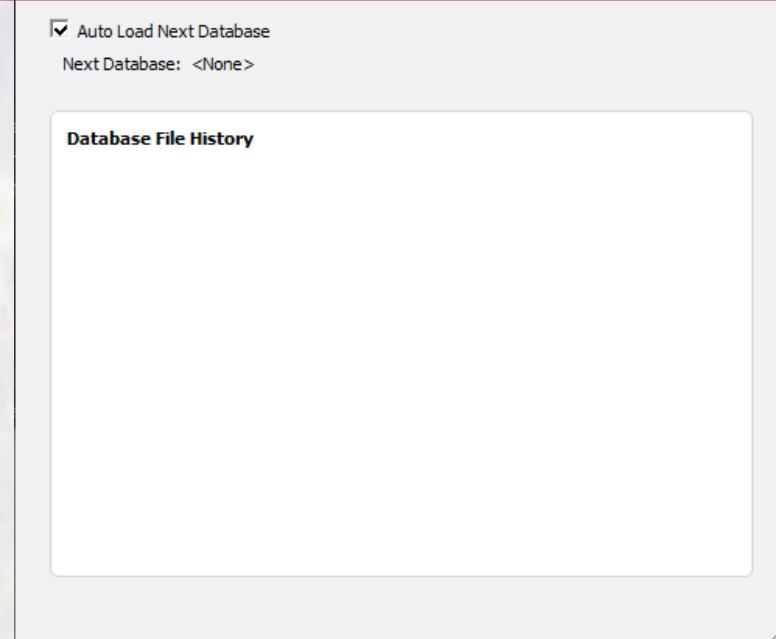
- TDCS-20210112181949.sqlite



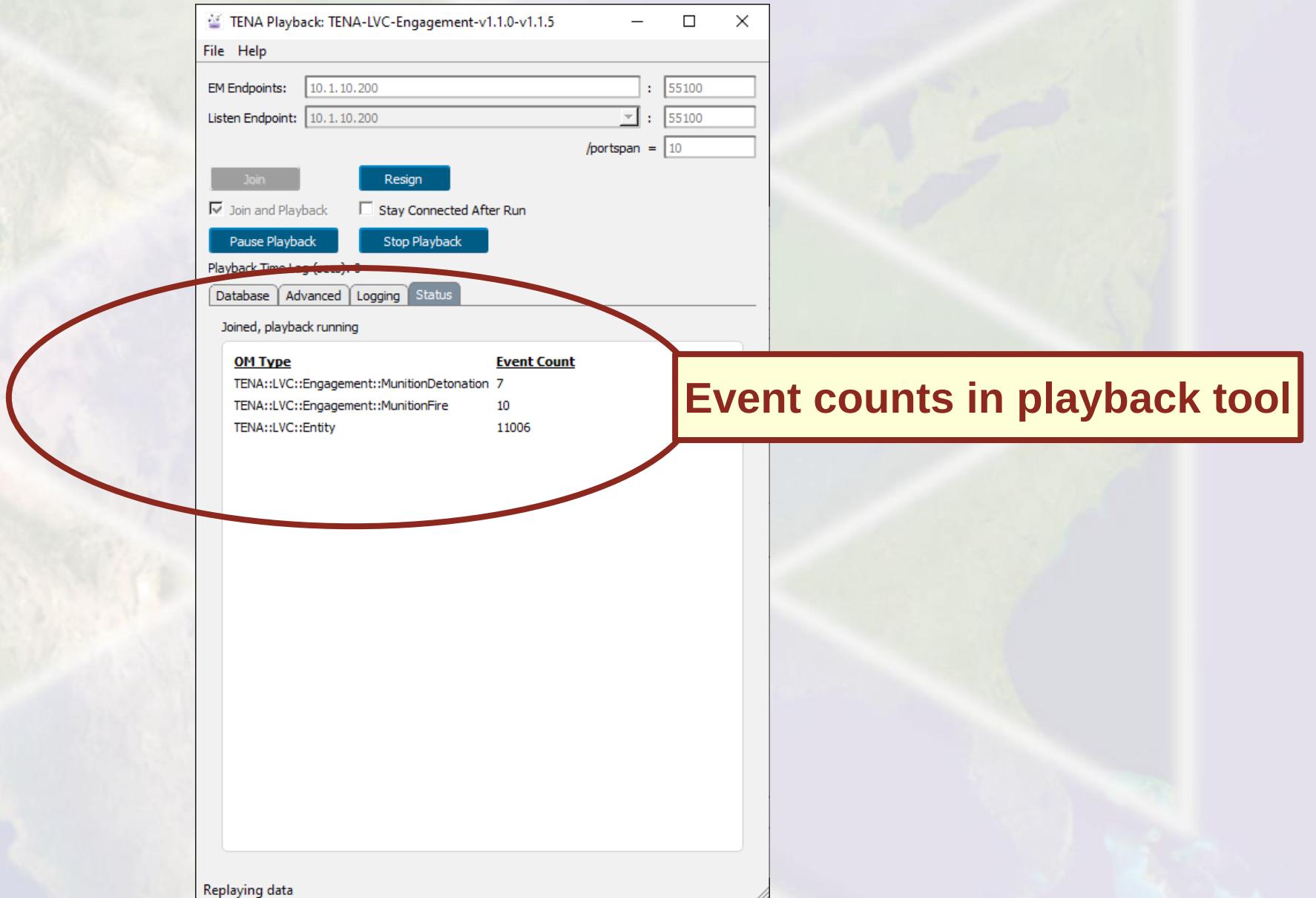
Starting playback tool

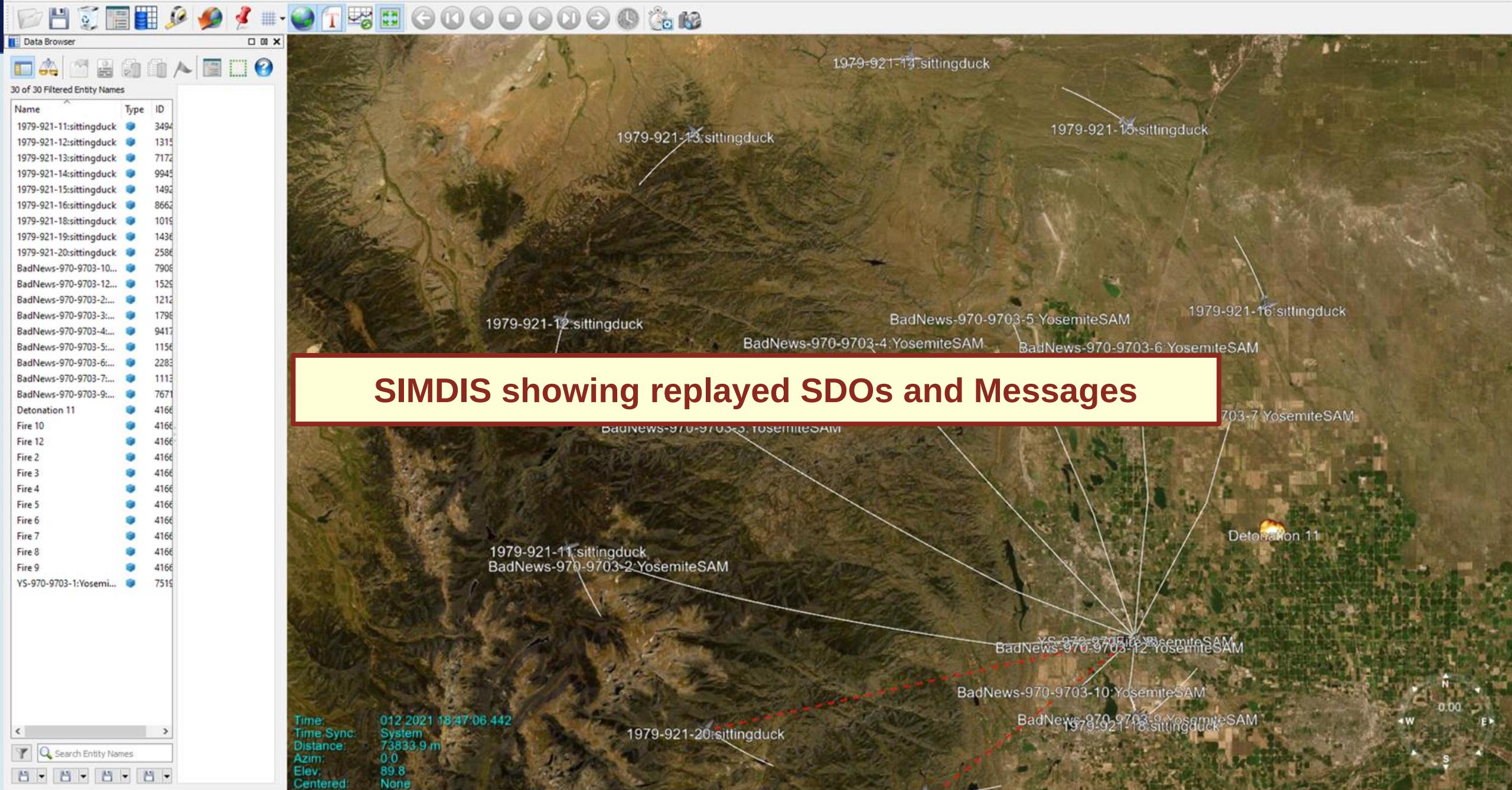


Select DB file,
set endpoints &
click “Join” button



Playback tool running





Playback tool in Console

TENA Console v1.0.23, Connected to EM 0 at 10.1.10.200:55100

File View Help

EM Running Execution Execution OM Stats Applications Network Monitoring Execution Managers Consoles **Alerts ***

Keep Active Apps Selected Hide Resigned/Terminated Apps Force Remove Remove from Execution

ID	App Name	Hostname	IP Address	Port	MW Versi...	Platform	Joined	Resign...	Status	+/-
2	SIMDIS10 10.0 1.28.0		10.1.10.2...	55102	6.0.7	w10-vs...	01/12/21 1...		OK	
9	TENA-LVC-Engagement-v1.1.0-PlaybackTool-v1.1.5		10.1.10.2...	55103	6.0.7	w10-vs...	01/12/21 1...		OK	

Application: # 9, tenaPlayback-TENA-LVC-Engagement-v1.1.0-v1.1.5

Configuration Application Details Heartbeat History **OM Stats**

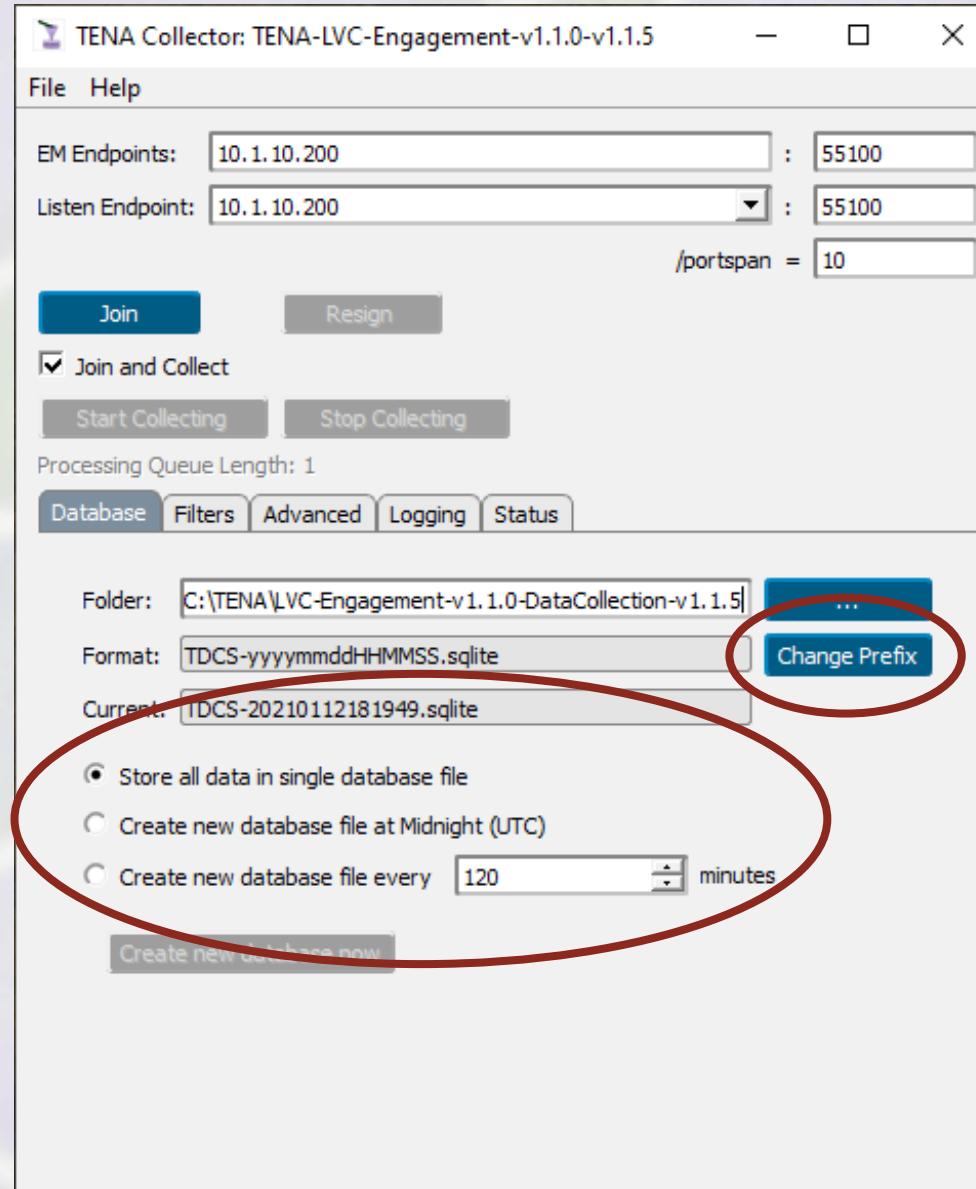
Refresh OM Stats Last Refreshed at 01/12/21 11:52:33.119-0

ID	Session	Type Name	Type	Servants Created	Updates Sent	Servants Destroyed	Discover	Updates Recvd	Destruct	Msgs Sent	Msgs Recvd	+/-
9	TENA-LV...	TENA::LVC::Entity	SDO	21	10600	8	0	0	0	-	-	
9	TENA-LV...	TENA::LVC::Track	SDO	0	0	0	0	0	0	-	-	
9	TENA-LV...	TENA::LVC::Engagement:MunitionDet...	Message	-	-	-	-	-	-	4	0	
9	TENA-LV...	TENA::LVC::Engagement:MunitionFire	Message	-	-	-	-	-	-	10	0	

Using Console to get playback tool OM stats

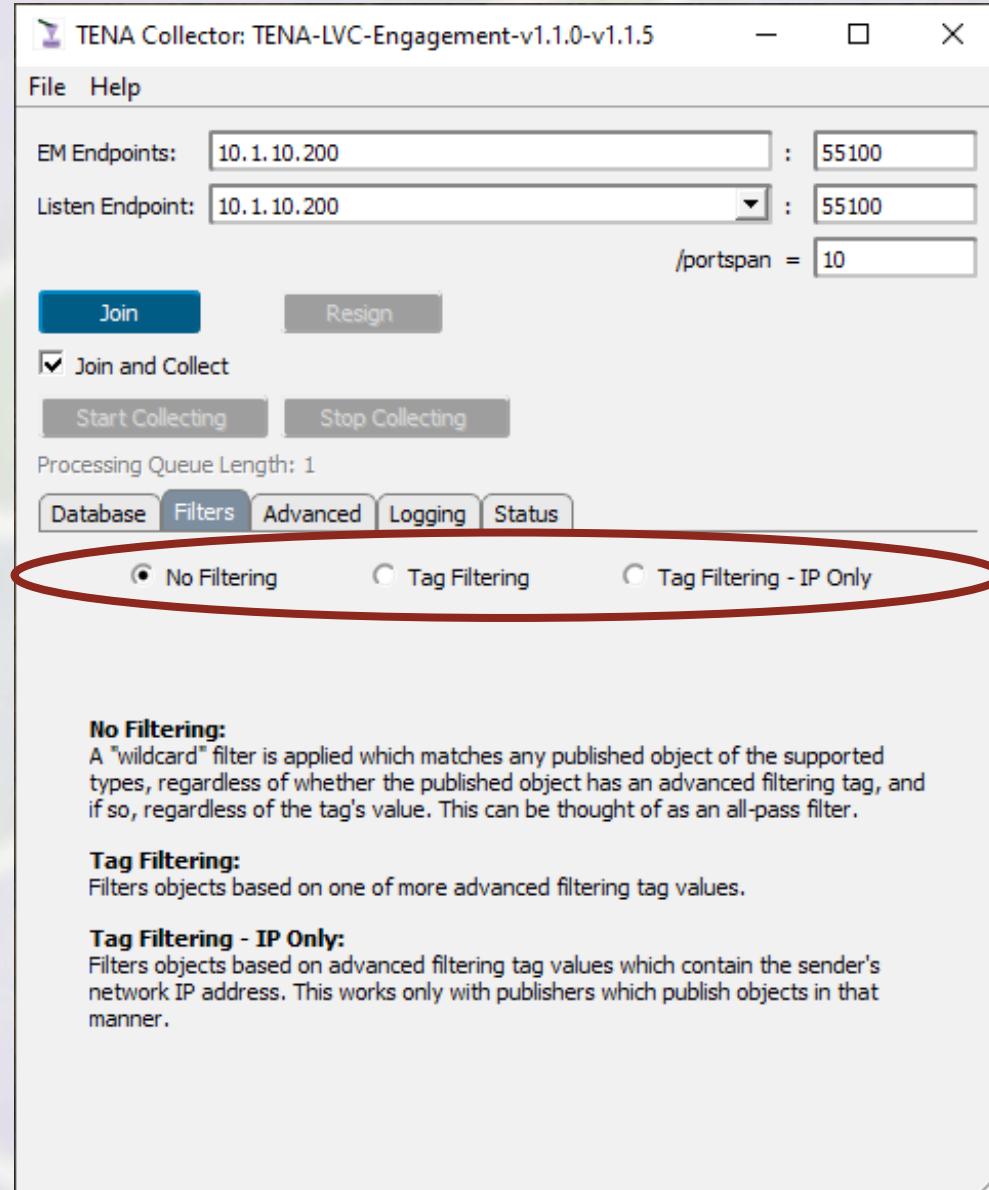
Other collector options

- Database options
 - File name prefix can be changed
 - Create new DB at midnight UTC or specify number of minutes



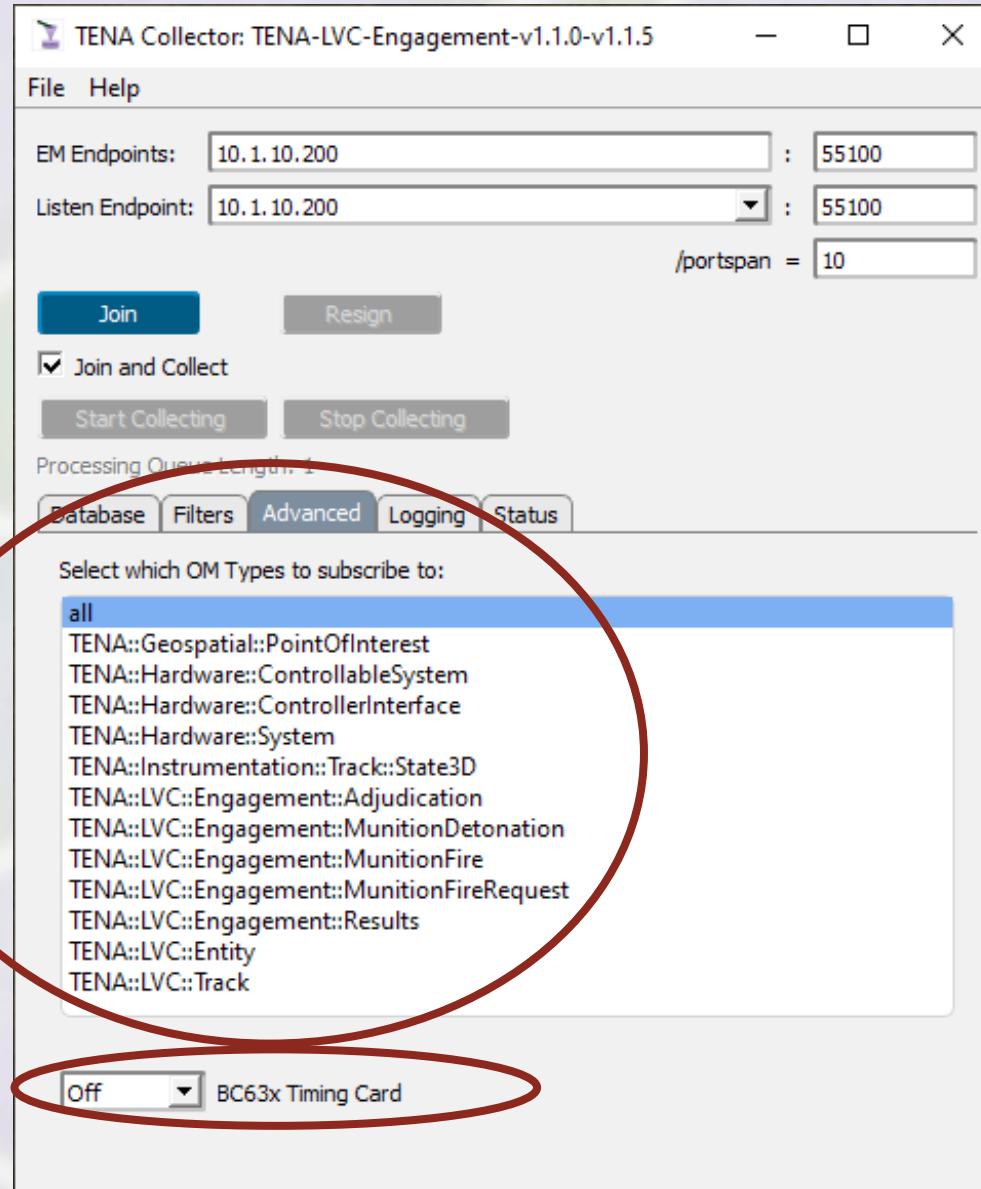
Other collector options

- Filter options
 - Tag filtering
 - Tag filtering – IP only



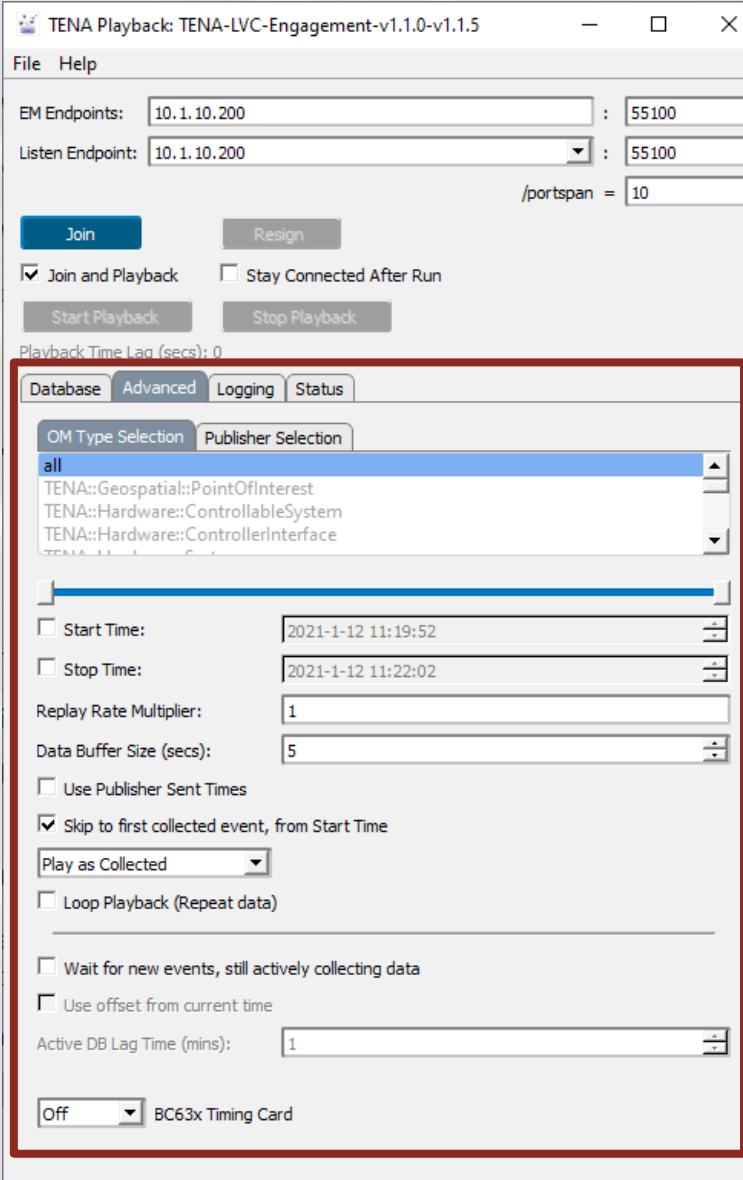
Other collector options

- Advanced options
 - OM SDO/Message type selection
 - BC63x timing card



Other playback tool options

- Advanced options
 - OM SDO/Message type selection
 - Publisher selection
 - Start/stop time selection
 - Replay rate
 - Use publisher sent times
 - Skip to first event
 - Loop playback
 - BC63x timing card
 - etc.





dbExporter

```
Command Prompt

> C:\TENA\dbExporter-v1.4.2\start.bat --database c:\TENA\vehicle-v1-DataCollection-v1.1.4\TDCS-20201022022107.sqlite
--outfile c:\temp-tdcs\demo.xml
Connected to C:\TENA\vehicle-v1-DataCollection-v1.1.4\TDCS-20201022022107.sqlite
Loading Enums into memory
Loading Vectors for cross-referencing
writing out SDO and Message work-sheets
writing out the Vector work-sheets
writing out work-sheet index
Done writing c:\temp-tdcs\demo.xml

>
```

Exporting data to an XML spreadsheet format



Viewing XML spreadsheet in Microsoft Excel

Screenshot of Microsoft Excel showing an XML spreadsheet named "dm.xml". The spreadsheet contains data from rows 1 to 40, with columns A through U. The columns represent various metadata fields such as timestamp, location, event type, and state charge.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	
1	Metadata,TimeOfCommit	Metadata,TimeOfReceipt	Metadata,EntityType	name,String	team	location,xminYmin,Float64	location,yminXmax,Float64	Metadata,SDCm,hostIpAddress	Metadata,SDCm,processID	Metadata,SDCm,processCreateTimestamp												
2	2021-10-22 02:21:11.971227	2020-10-22 02:21:11.981054	Discovery	Team_Req		0	0	16777343	22168	1613383271										3 Reliable		
3	2021-10-22 02:21:12.973045	2020-10-22 02:21:12.973079	StateChange	Update#1_Team_Req		1	1	16777343	22168											3 Reliable		
4	2021-10-22 02:21:13.973282	2020-10-22 02:21:13.974070	StateChange	Update#2_Team_Req		2	2	16777343	22168										3 Reliable			
5	2021-10-22 02:21:14.973594	2020-10-22 02:21:14.974533	StateChange	Update#3_Team_Req		3	3	16777343	22168										3 Reliable			
6	2021-10-22 02:21:15.974061	2020-10-22 02:21:15.974030	StateChange	Update#4_Team_Req		4	4	16777343	22168										3 Reliable			
7	2021-10-22 02:21:16.974225	2020-10-22 02:21:16.975052	StateChange	Update#5_Team_Req		5	5	16777343	22168										3 Reliable			
8	2021-10-22 02:21:17.974766	2020-10-22 02:21:17.975555	StateChange	Update#6_Team_Req		6	6	16777343	22168										3 Reliable			
9	2021-10-22 02:21:18.975611	2020-10-22 02:21:18.976417	StateChange	Update#7_Team_Req		7	7	16777343	22168										3 Reliable			
10	2021-10-22 02:21:19.975892	2020-10-22 02:21:19.976736	StateChange	Update#8_Team_Req		8	8	16777343	22168										3 Reliable			
11	2021-10-22 02:21:20.976065	2020-10-22 02:21:20.977035	StateChange	Update#9_Team_Req		9	9	16777343	22168										3 Reliable			
12	2021-10-22 02:21:21.976725	2020-10-22 02:21:21.977025	StateChange	Update#10_Team_Req		10	10	16777343	22168										3 Reliable			
13	2021-10-22 02:21:22.977276	2020-10-22 02:21:22.978027	StateChange	Update#11_Team_Req		11	11	16777343	22168										3 Reliable			
14	2021-10-22 02:21:23.977482	2020-10-22 02:21:23.978315	StateChange	Update#12_Team_Req		12	12	16777343	22168										3 Reliable			
15	2021-10-22 02:21:24.978611	2020-10-22 02:21:24.978611	StateChange	Update#13_Team_Req		13	13	16777343	22168										3 Reliable			
16	2021-10-22 02:21:25.979327	2020-10-22 02:21:25.980314	StateChange	Update#14_Team_Req		14	14	16777343	22168										3 Reliable			
17	2021-10-22 02:21:26.979435	2020-10-22 02:21:26.980614	StateChange	Update#15_Team_Req		15	15	16777343	22168										3 Reliable			
18	2021-10-22 02:21:27.980146	2020-10-22 02:21:27.980912	StateChange	Update#16_Team_Req		16	16	16777343	22168										3 Reliable			
19	2021-10-22 02:21:28.980864	2020-10-22 02:21:28.981058	StateChange	Update#17_Team_Req		17	17	16777343	22168										3 Reliable			
20	2021-10-22 02:21:29.981356	2020-10-22 02:21:29.981356	StateChange	Update#18_Team_Req		18	18	16777343	22168										3 Reliable			
21	2021-10-22 02:21:30.981413	2020-10-22 02:21:30.982287	StateChange	Update#19_Team_Req		19	19	16777343	22168										3 Reliable			
22	2021-10-22 02:21:31.981444	2020-10-22 02:21:31.982451	StateChange	Update#20_Team_Req		20	20	16777343	22168										3 Reliable			
23	2021-10-22 02:21:32.981482	2020-10-22 02:21:32.982471	StateChange	Update#21_Team_Req		21	21	16777343	22168										3 Reliable			
24	2021-10-22 02:21:33.982671	2020-10-22 02:21:33.983559	StateChange	Update#22_Team_Req		22	22	16777343	22168										3 Reliable			
25	2021-10-22 02:21:34.983495	2020-10-22 02:21:34.984352	StateChange	Update#23_Team_Req		23	23	16777343	22168										3 Reliable			
26	2021-10-22 02:21:35.984521	2020-10-22 02:21:35.985313	StateChange	Update#24_Team_Req		24	24	16777343	22168										3 Reliable			
27	2021-10-22 02:21:36.984777	2020-10-22 02:21:36.985759	StateChange	Update#25_Team_Req		25	25	16777343	22168										3 Reliable			
28	2021-10-22 02:21:37.985231	2020-10-22 02:21:37.986254	StateChange	Update#26_Team_Req		26	26	16777343	22168										3 Reliable			
29	2021-10-22 02:21:38.985501	2020-10-22 02:21:38.985930	StateChange	Update#27_Team_Req		27	27	16777343	22168										3 Reliable			
30	2021-10-22 02:21:39.986564	2020-10-22 02:21:39.987402	StateChange	Update#28_Team_Req		28	28	16777343	22168										3 Reliable			
31	2021-10-22 02:21:40.987512	2020-10-22 02:21:40.988272	StateChange	Update#29_Team_Req		29	29	16777343	22168										3 Reliable			
32	2021-10-22 02:21:41.987563	2020-10-22 02:21:41.988017	StateChange	Update#30_Team_Req		30	30	16777343	22168										3 Reliable			
33	2021-10-22 02:21:41.987961	2020-10-22 02:21:41.988617	Destruction	Update#30_Team_Req		30	30	16777343	22168										3 Reliable			
34																						
35																						
36																						
37																						
38																						
39																						
40																						



TDCS Demo



- Live demo



Part 2: Using DataView with TDCS

- What is DataView?
- Acquiring & installing DataView
- DataView demonstration



What is DataView?

- DataView displays TDCS data in a human-readable graphical user interface (GUI)
 - While data is still being collected
 - Or after data has been collected
- Similar look and feel compared to the Wireshark packet analyzer app
- Provides capabilities that assist with
 - Analysis, visualization, and plotting
 - Data filtering and CSV export
 - Testing and troubleshooting
- Runs on Windows and Linux



DataView interface

TENA DataView (v1.4.0)

File Help

Auto-Scroll Display Detail Data Limit Event Cache Set Event Cache Size 10000 Expand SDO Pointers Show Optional Unset Attributes Reload Display Attributes Settings Plot Data Switch to Advance Filtering

OM Type: All Select All Select All Filter By Time Of Receipt Start Value Filter By Time Of Receipt End Value Clear All

Object Name	TimeOfReceipt (SDOs/Messages)	ApplicationId (SDOs/Messages)	Endpoint (SDOs/Messages)	MulticastAddress (SDOs/Messages)	StateVersion (SDOs)	EventType
TENA::PlatformDetails	2019-08-19 11:04:21.535908-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::Platform	2019-08-19 11:04:21.538837-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::Platform	2019-08-19 11:04:21.542741-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::PlatformDetails	2019-08-19 11:04:21.542741-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::Platform	2019-08-19 11:04:21.545669-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::Platform	2019-08-19 11:04:21.545669-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::PlatformDetails	2019-08-19 11:04:21.546645-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::PlatformDetails	2019-08-19 11:04:21.546645-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::Platform	2019-08-19 11:04:21.546645-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::PlatformDetails	2019-08-19 11:04:21.547620-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::Platform	2019-08-19 11:04:21.547620-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::PlatformDetails	2019-08-19 11:04:21.547620-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::Platform	2019-08-19 11:04:21.547620-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::PlatformDetails	2019-08-19 11:04:21.548597-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::Platform	2019-08-19 11:04:21.548597-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::PlatformDetails	2019-08-19 11:04:21.548597-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::Platform	2019-08-19 11:04:21.549573-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::Platform	2019-08-19 11:04:21.549573-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::PlatformExtension	2019-08-19 11:04:21.550548-0400	16	127.0.0.1:52406	N/A	1	Discovery
TENA::PlatformDetails	2019-08-19 11:04:21.550548-0400	16	127.0.0.1:52406	N/A	1	Discovery

Object Type Event Count
TENA::Platform (SDO) 180
TENA::PlatformDetails (SDO) 180
TENA::PlatformExtension (SDO) 180
TENA::Track (SDO) 180

Event (Metadata) Panel

Object Types Event Count Status

TENA::Platform

+ Metadata Information
- Attributes
+ platformID (Constant)
+ platformType.typeIndex (Constant) <Optional>
+ platformType (Constant)
+ platformType.disEntityType (Constant) <Optional>
- designator: !# (Constant)
- affiliation: Affiliation_Hostile (3)
- damageState: DamageState_Communication (2)
- damageInPercent: 0.0280000027269125 <Optional>
+ tspi.time
+ tspi.position.geodetic_asTransmitted <Optional>
+ tspi.velocity.geocentric_asTransmitted <Optional>
+ tspi.acceleration.geocentric_asTransmitted <Optional>
+ tspi.orientation.frWRTgeocentricBodyFixedZYX_asTransmitted <Optional>
+ tspi.angularVelocity.geocentric_asTransmitted <Optional>
+ tspi.angularAcceleration.geocentric_asTransmitted <Optional>
+ sendTime <Optional>
- TENA::PlatformDetails,pPlatformDetails (No Data)

Detail Attribute Panel

Result Count for Filter. Also placing the mouse over the text will show itemized list of counts per object type.

Start/Current/End Time and Duration

Database Name
Configuration File Name
Messages Displayed Here

Filter Event Count: 720
Start Time: 2019-08-19 11:04:21.535908-0400
Last Update: 2019-08-19 11:04:37.929337-0400
Duration: 00:00:00:16.393429

C:/Java/SQLite DB-NS1/Platform-10 Objects.sqlite Config: C:/Java/SQLite DB-NS1/csv/platFormAllAFRows.txt



Acquiring & installing DataView

- DataView is acquired from the TENA repository in the TDCS group
- Let's get DataView...

Home - TRMC - TRMC Website +

This website is for process, or transmit any CLASSIFIED information.

Spaces Create ... Search ?

Dashboard Edit Save for later Watch ...

Home

Test Resource Management Center

 **JMETC**

Joint Mission Environment Test Capability

The JMETC mission is to provide a persistent capability for linking distributed facilities, enabling DoD customers to develop and test warfighting capabilities in a Joint Context. JMETC provides a test infrastructure consisting of the components necessary to conduct Joint distributed test events by cost-effectively integrating live, virtual, and constructive (LVC) test resources that are configured to support the users' needs. The JMETC program provides its

 **TENA**

Test and Training Enabling Architecture

The United States Office of the Secretary of Defense Test Resource Management Center (TRMC) has developed a common architecture to support effective integration and reuse of testing, training, and simulation capabilities that require real-time collaboration between distributed computer systems operating within diverse testing and training environments. Through the establishment of the Test

 **National Cyber Range Complex**

The National Cyber Range Complex (NCRC) is a holistic, integrated cyber range capability comprising the National Cyber Range (NCR), Regional Service Delivery Points (RSDP), and the Joint Mission Environment Test Capability (JMETC) Multiple Independent Levels of Security (MILS) Network (JMN). The NCRC provides Department of Defense (DoD) and other government and industry users with representative environments, a distributed network infrastructure and other tools and services to support

Click on the TENA link

Home - TENA - TRMC Website +

https://www.trmcosd.mil/wiki/display/TENA/Home

This website is for UNCLASSIFIED USE ONLY. Do not discuss, enter, transfer, process, or transmit any CLASSIFIED information.

Spaces Create ... Search ? 

Dashboard  Edit Save for later Watch ...

Test and Training Enabling Architecture

TENA News & Events

- JMETC Tech Talks will begin in June - visit the JMETC Tech Talks page to see upcoming talks
- Proceedings & helpdesk cases for previous JTEX meetings (JTEX-01 thru -05)
- "Bringing Speed to DoD Cybersecurity: Applying Best Practices for Agile RDT&E Environments" (ITEA Journal, Dec 2020)
- JMETC Brings Readily-Available Persistent Connectivity for Joint Distributed Test Events (2020-09-03)
- JMETC Provides Persistent Connectivity and Support to Enable Air Force System Interoperability Test (AFSIT) (2019-11-14)
- JMETC Support of Multiple Joint Interoperability Tests (JITs) (2019-10-29)
- TENA Overview Briefing (2019-10-15)
- TRMC Big Data Knowledge Management (BDKM) Enterprise Overview Briefing (2019-08-01)
- JMETC Overview Briefing (2019-06-05)
- TENA Middleware 6.0.7 released, added computer platform support and made various improvements (2019-05-13)

TENA-team members can add news items on the [News](#) wiki page.

Favorite User Groups

- TENA: Utilities and Tools   



Click on the TENA Repository link



TENA Repository

Access the TENA repository for browsing and building object models.



TENA Project Information

Access TENA project and project related information.



TENA Middleware

Obtain information concerning the TENA Middleware.



TENA Training

Obtain information concerning TENA training.



Object Models

Obtain information about TENA Object Models.



TENA Helpdesk

Access the TENA Helpdesk to submit issues or search for solutions.



TENA User Profile

Review and update your user profile.

TENA - Test and Training Enabling Architecture

TENA Website Repository Home Components Help

Welcome
The TENA Repository is a web-based storehouse for TENA interoperability solutions.



TENA Repository
Discover, reuse and share
TENA components

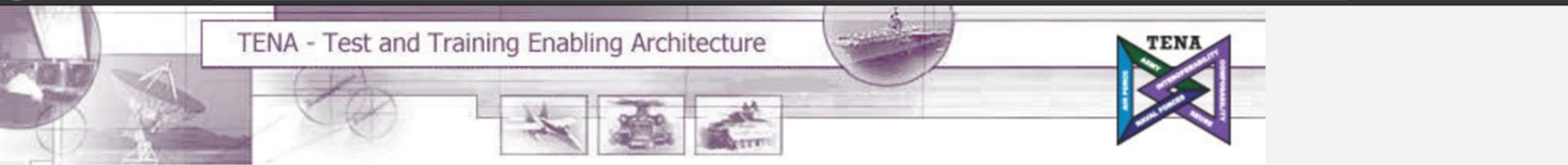
Click on the “All products and Object Models” link

Getting Started			
Quick Start Guide Information describing how to use the TENA Repository.	All Products and Object Models Browse products and object models from all groups.	TENA Standard Object Models A Collection of C++ SDKs for the TENA Standard Object Models	TENA Canary The TENA Canary is a simple application that can connect to an execution to help test the network.
Featured Products and Object Models Featured Products and Object Models	Submit Object Model Submit or revise an object model	TENA Middleware The TENA Middleware is a software framework that enables rapid and reliable development of scalable distributed systems.	TENA Console The TENA Console is a tool that assists in the troubleshooting and monitoring of TENA executions.

3.2.5.2trunk

Products and Object Models in PMRF		
REPO Products and Object Models in REPO	RMAST Products and Object Models in RMAST	ProLogic Products and Object Models in ProLogic
RTTC Products and Object Models in RTTC	RelayNode Products and Object Models in RelayNode	RTC Products and Object Models in RTC
SATIN Products and Object Models in SATIN	SIMDIS Products and Object Models in SIMDIS	SAIC Products and Object Models in SAIC
STAT Products and Object Models in STAT	SUMS Products and Object Models in SUMS	SPAWAR Products and Object Models in SPAWAR
TA Products and Object Models in TA	TACE Products and Object Models in TACE	Simmetry Products and Object Models in Simmetry
TDCS Products and Object Models in TDCS	TSPIL Products and Object Models in TSPIL	TCS Products and Object Models in TCS
TRCE Products and Object Models in TRCE	VENGEANCE Products and Object Models in VENGEANCE	TENA Products and Object Models in TENA
USNTTR Products and Object Models in USNTTR	WebBinding Products and Object Models in WebBinding	TVDS Products and Object Models in TVDS
WSMR Products and Object Models in WSMR	tmp Products and Object Models in tmp	VPEF Products and Object Models in VPEF
YTC Products and Object Models in YTC		Weibel Products and Object Models in Weibel

Navigate to the TDCS group



TENA Website Repository Home Components Help

Browse | Search



Show archived

Found: 8 Showing: 8

Name	Description	Type	Attributes
TDCS-DataView-v1.4.0	The TENA DataView Tool displays data stored on an SQLite database populated by TENA Data Collection System. It requires database generated by TDCS v1.1.5 or later.	Product	Released Public
Select the latest version of DataView		ObjectModel	Released Private
TDCS-QuickTest-v2		ObjectModel	Draft Private
TDCS-QuickTest-v2.1		ObjectModel	Draft Private
TDCS-Tests-Coverage-Complex-v1		ObjectModel	Released Private

TDCS-DataView-v1.4.0

The TENA DataView Tool displays data stored on an SQLite database populated by TENA Data Collection System. It requires database generated by TDCS v1.1.5 or later.

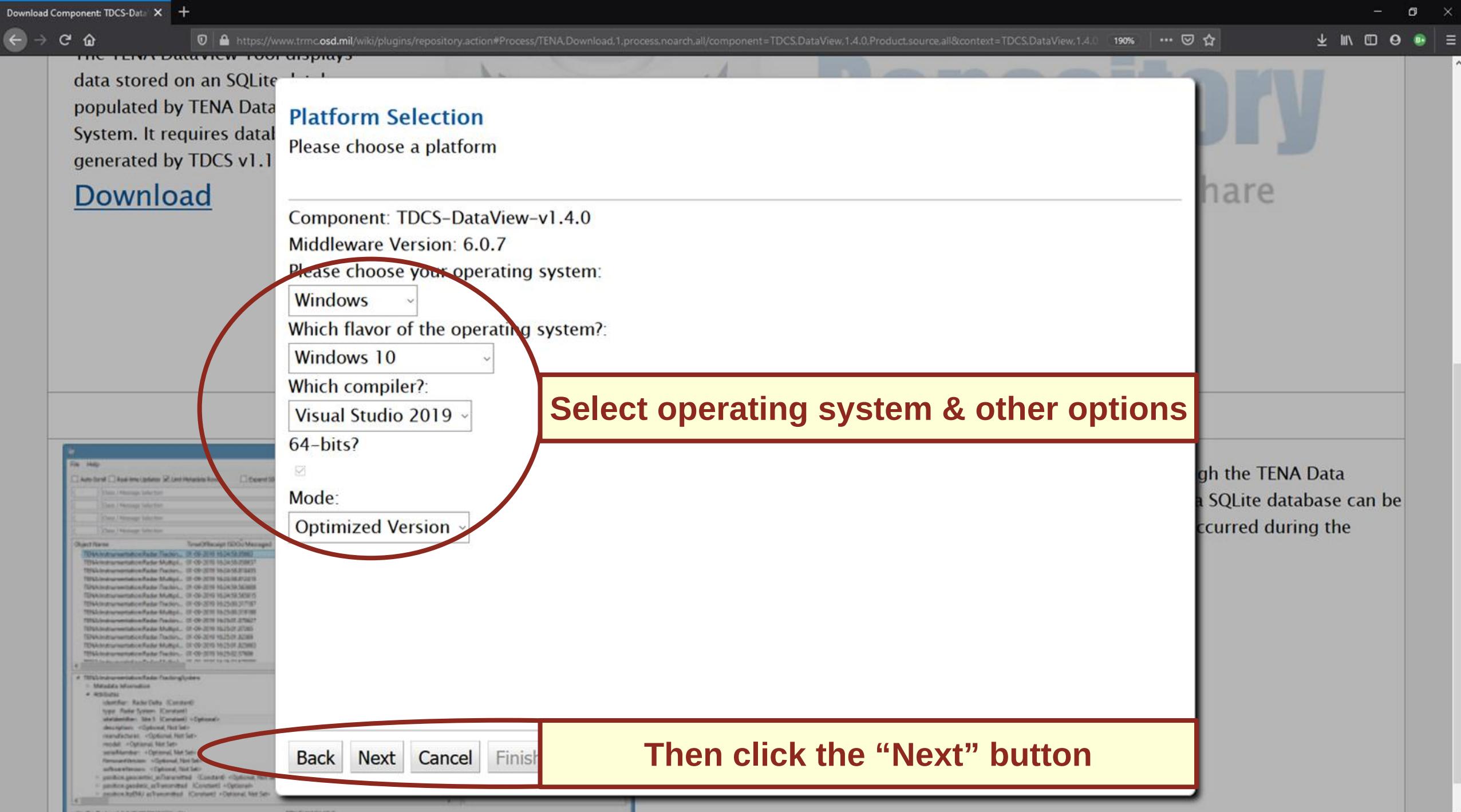
[Download](#)

Click the “Download” link



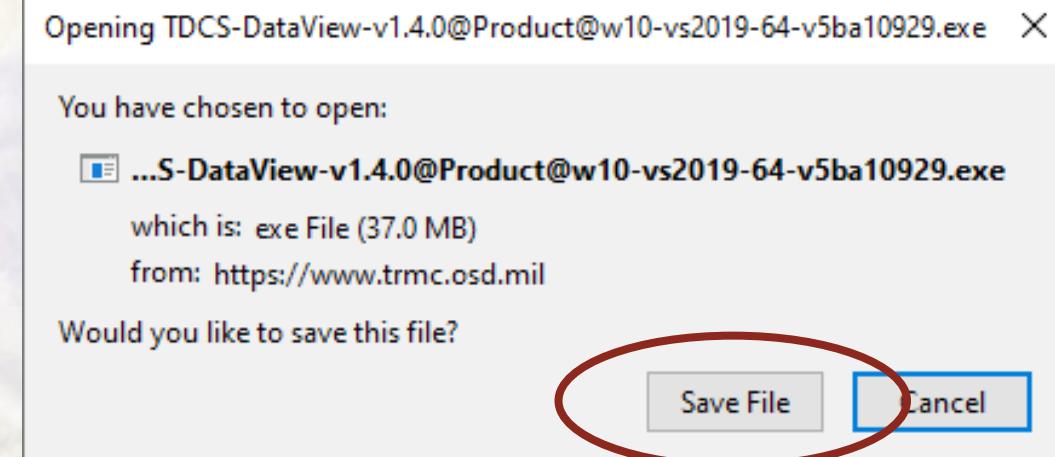
[Overview](#) | [Documentation](#) | [Help](#) | [Versions](#) | [Contents](#) | [History](#)

The TENA DataView tool can view data that is collected through the TENA Data Collection System (TDCS). Event data captured by TDCS into a SQLite database can be seen in TENA DataView to show detailed data of the events occurred during the event.



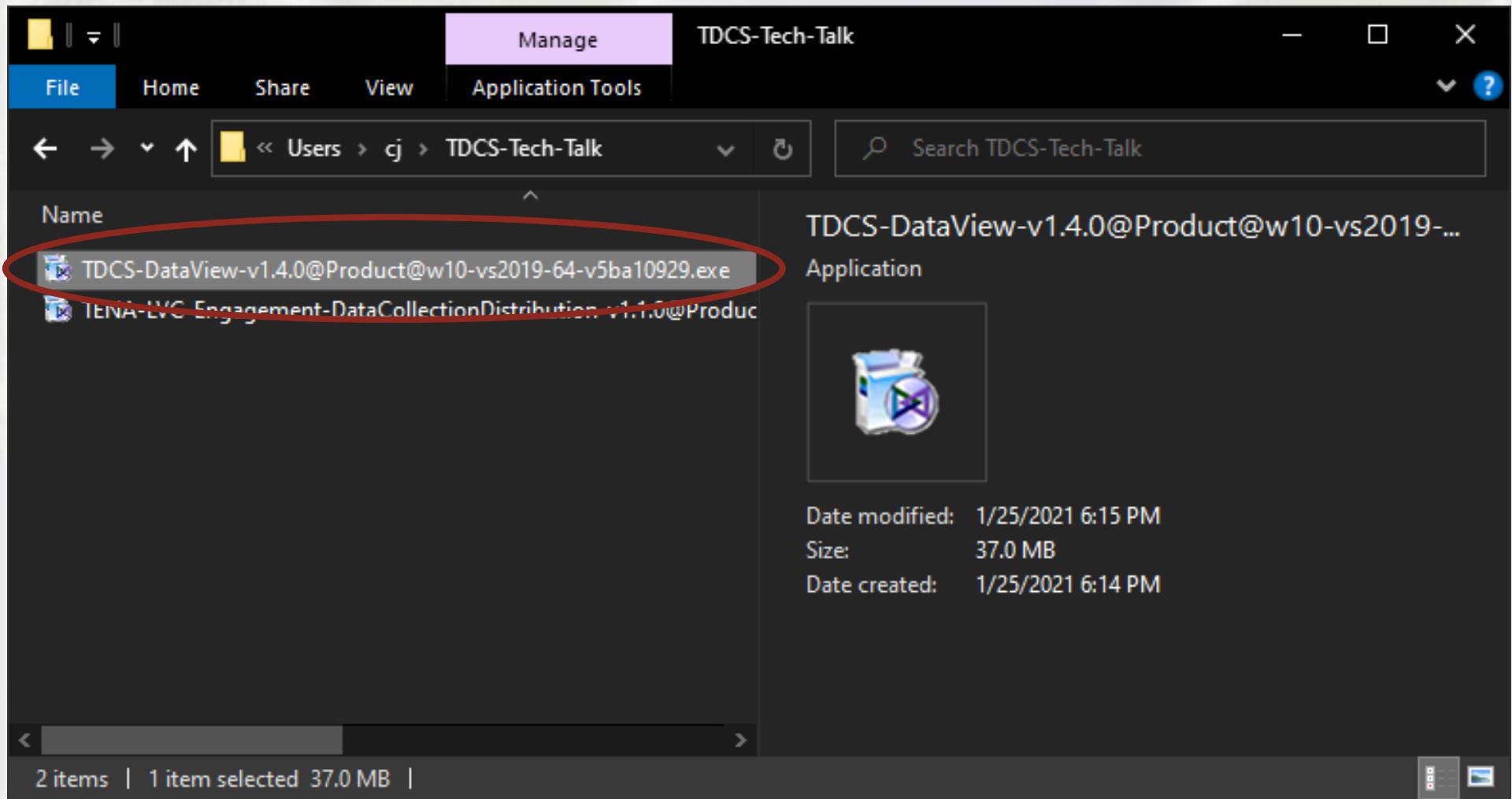


Save the installer

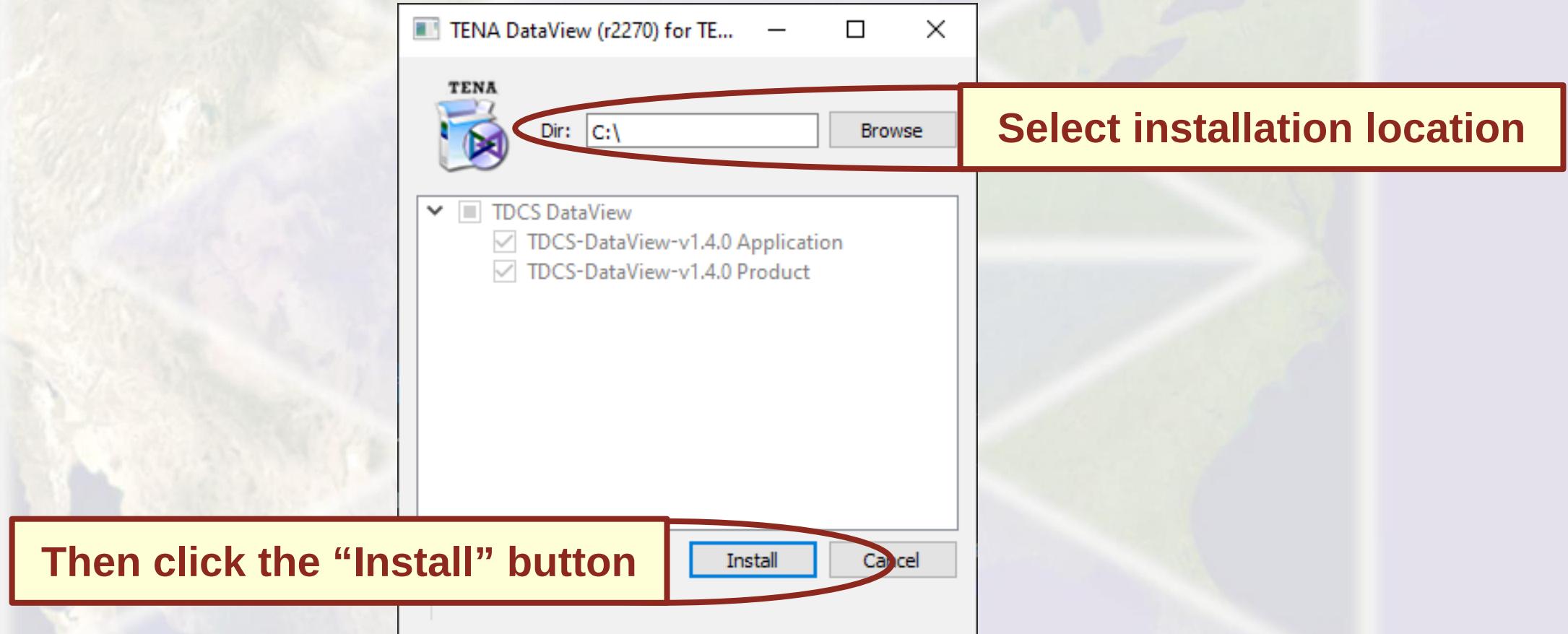




Run the installer



DataView installer

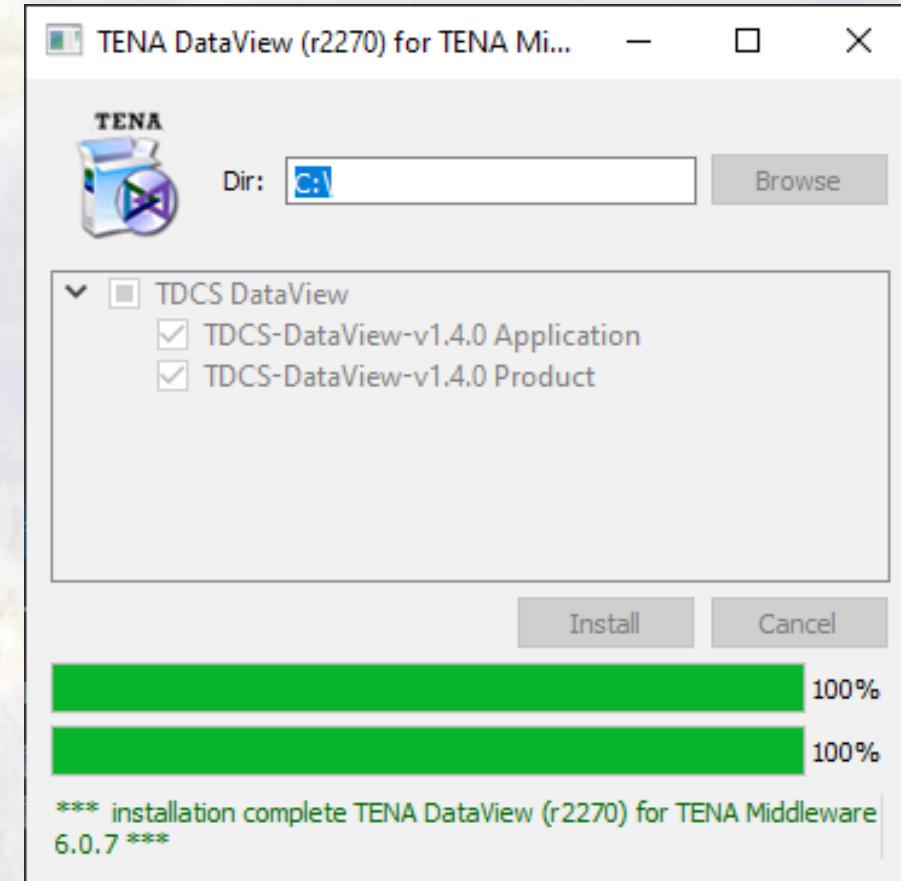


Then click the “Install” button

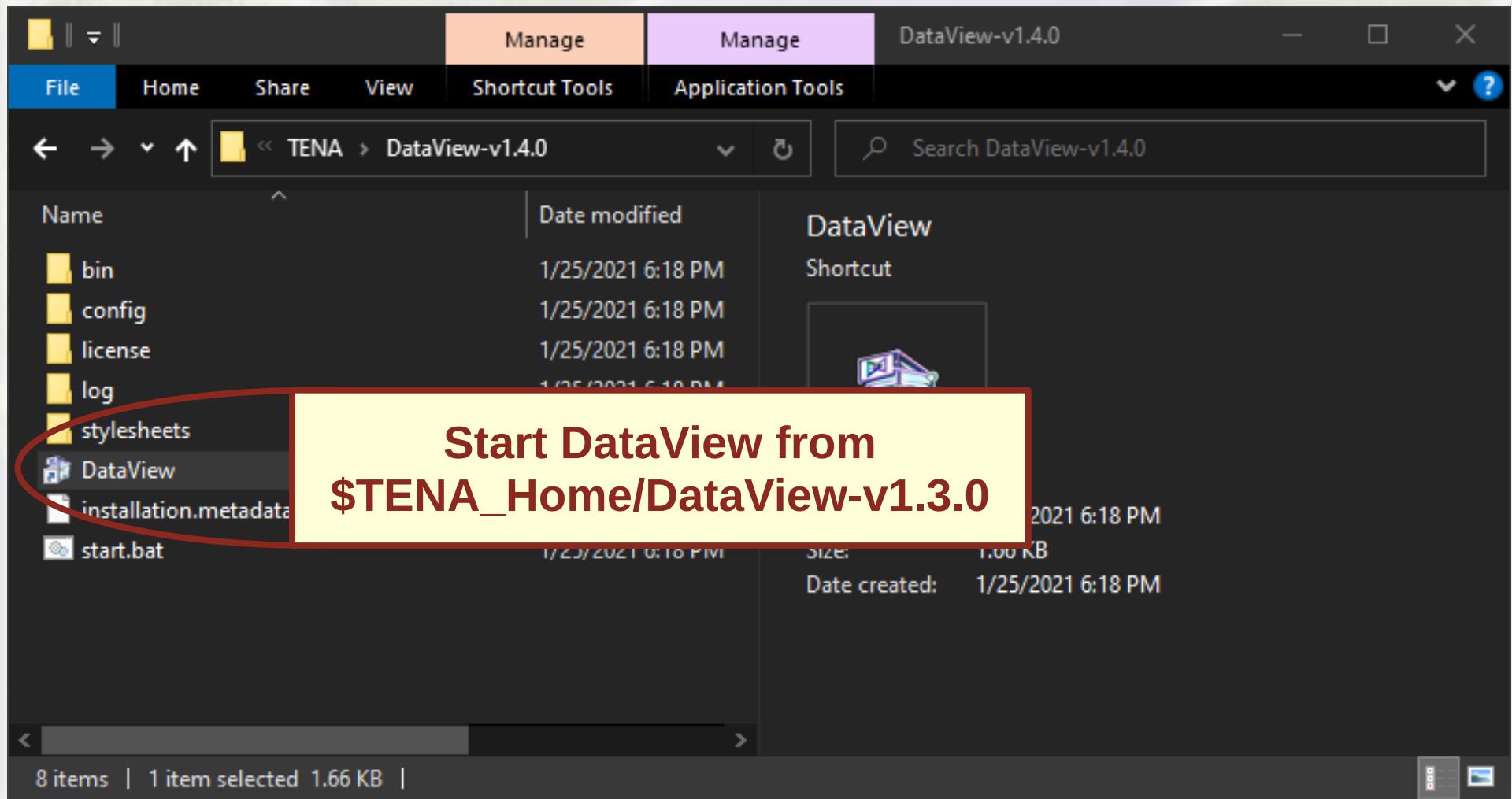
Select installation location



Installer will close when done



DataView shortcut



OM Type: All							Object Type	Event Count
Object Name	TimeOfReceipt (SDOs/Messages)	ApplicationId (SDOs/Messages)	Endpoint (SDOs/Messages)	MulticastAddress (SDOs/Messages)	StateVersion (SDOs)	EventType (SDOs)	Object Type	Event Count
TENA::LVC::Entity	2021-01-25 15:27:34.975554-0400	8	10.1.10.200:55103	N/A	364	StateChange	TENA::LVC::Entity (SDO)	9864
TENA::LVC::Entity	2021-01-25 15:27:34.985590-0400	8	10.1.10.200:55103	N/A	364	StateChange	TENA::LVC::Engagement::MunitionDeto...	10
TENA::LVC::Entity	2021-01-25 15:27:34.996585-0400	8	10.1.10.200:55103	N/A	364	StateChange	TENA::LVC::Engagement::MunitionFire ...	10
TENA::LVC::Entity	2021-01-25 15:27:35.006775-0400	8	10.1.10.200:55103	N/A	365	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.016784-0400	8	10.1.10.200:55103	N/A	365	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.025359-0400	8	10.1.10.200:55103	N/A	365	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.035715-0400	8	10.1.10.200:55103	N/A	365	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.044801-0400	8	10.1.10.200:55103	N/A	365	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.055839-0400	8	10.1.10.200:55103	N/A	365	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.065808-0400	8	10.1.10.200:55103	N/A	365	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.076151-0400	8	10.1.10.200:55103	N/A	365	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.085293-0400	8	10.1.10.200:55103	N/A	365	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.094763-0400	8	10.1.10.200:55103	N/A	365	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.106765-0400	8	10.1.10.200:55103	N/A	366	StateChange		
TENA::LVC::Engagement::MunitionFire	2021-01-25 15:27:35.108768-0400	13	10.1.10.200:55105	N/A				
TENA::LVC::Entity	2021-01-25 15:27:35.108768-0400	13	10.1.10.200:55105	N/A	1	Discovery		
TENA::LVC::Entity	2021-01-25 15:27:35.109770-0400	13	10.1.10.200:55105	N/A	2	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.117069-0400	8	10.1.10.200:55103	N/A	366	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.127423-0400	8	10.1.10.200:55103	N/A	366	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.135412-0400	8	10.1.10.200:55103	N/A	366	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.145001-0400	8	10.1.10.200:55103	N/A	366	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.157016-0400	8	10.1.10.200:55103	N/A	366	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.166015-0400	8	10.1.10.200:55103	N/A	366	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.175742-0400	8	10.1.10.200:55103	N/A	366	StateChange		
TENA::LVC::Entity	2021-01-25 15:27:35.185							
TENA::LVC::Entity	2021-01-25 15:27:35.196							
TENA::LVC::Entity	2021-01-25 15:27:35.205							

Viewing our collected data in DataView

TENA::LVC::Entity

- Metadata Information
- Attributes
 - identifier: 1979-921-20 (Constant)
 - sourcelIdentifier: sittingduck (Constant)
 - entityID (Constant)
 - lvIndicator: LVCIndicator_Constructive (2) (Constant)
 - entityType (Constant)
 - tspi.time
 - tspi.position.geodetic_asTransmitted <Optional>
 - tspi.velocity.geocentric_asTransmitted <Optional>
 - tspi.orientation.frdrWRTltpENUbodyFixedZYX_asTransmitted <Optional>
 - affiliation: Affiliation_Hostile (3)
 - damageState: DamageState_NoDamage (1)
 - deadReckoningAlgorithm: DeadReckoningAlgorithm_None (1)
 - sensorPedigree.count: 0
 - trackPedigree.count: 0
 - appearance.landPlatformAppearance
 - appearance.airPlatformAppearance
 - appearance.munitionAppearance
 - appearance.humanLifeFormAppearance
 - LifeFormPosture,appearance.humanLifeFormAppearance
 - appearance.humanLifeFormAppearance
 - LifeFormWeaponOrImplement,appearance.humanLifeFormAppearance
 - EntityCamouflageType,appearance.humanLifeFormAppearance
 - appearance.humanLifeFormAppearance
 - appearance.environmentalAppearance
 - appearance.expendableAppearance



DataView demonstration

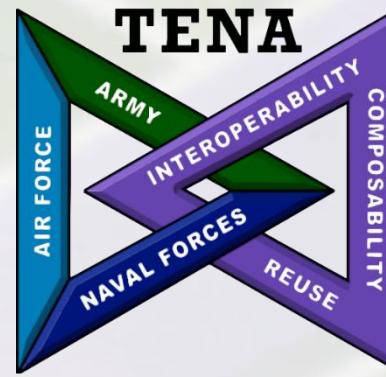
- Opening data files
- Overview of the GUI
- Navigating through data
 - Basic & advanced filtering
- Visual plotting capabilities
- CSV export capabilities
- Configuration files
- etc.



TENA DataView (TDV) Demo



- Live demo



Middleware configuration



Middleware configuration – mechanism

- Mechanism to control runtime behavior through
 - Command line arguments
 - Configuration files
 - Environment variables
- Used by the Middleware, EM, and C++ example apps
- C++ app developers can use the configuration mechanism for custom configuration requirements

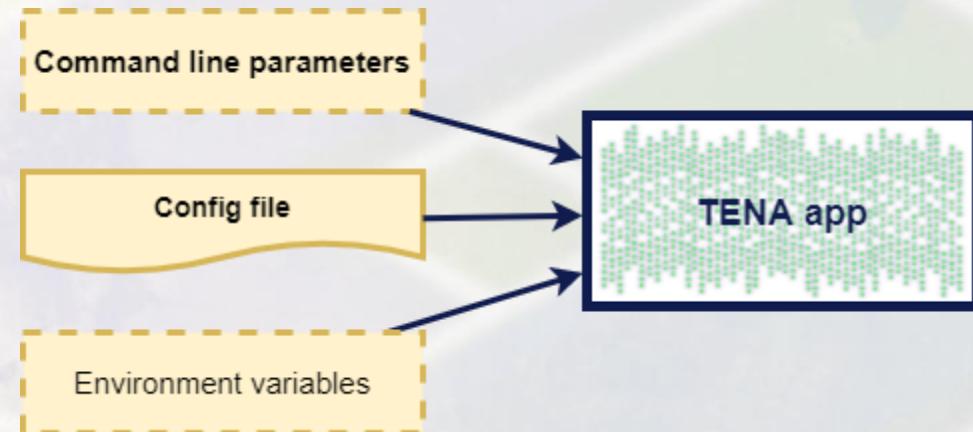
Middleware configuration – mechanism

- Each configuration parameter is controlled by a “Setting” (C++ object)
- A collection of Settings is managed by a “Configuration” (another C++ object) that supports
 - Adding settings
 - Controlling the parsing of settings values from multiple input sources
 - Retrieving the setting values.
- Input sources include
 - Command line arguments (argc, argv)
 - Configuration files
 - Environment variables
 - Key value pair list (useful in GUIs)

Middleware configuration – mechanism – configuration classes

- **TENA-Middleware-Utils-BasicConfiguration**
 - Provides basic services
 - Not recommended to be used directly
- **TENA-Middleware-Configuration**
 - Derived from BasicConfiguration
 - Contains the parameters associated with the Middleware
 - Provided to the TENA-Middleware-init() function to initialize Middleware
 - The only configuration class exposed via language bindings other than C++
- **C++ helper class: TENA-Middleware-ApplicationConfiguration**
 - Derived from BasicConfiguration
 - Can be extended for custom C++ application configuration
 - Parses input sources (command line parameters, config file, and environment variables)
 - Provides tenaConfiguration() method to obtain the TENA-Middleware-Configuration

Extending TENA-Middleware-ApplicationConfiguration (C++ helper class)





Example – using command line, config file, and/or environment

- Example apps provide an example ApplicationConfiguration class that extends TENA::Middleware::ApplicationConfiguration
- Custom settings can be added in ApplicationConfiguration.cpp, in the defineSettings() method

```
//...
using TENA::Middleware::Utils::Value, std::string;

this->addSettings()
    ("amigo1", Value<string>(), "Amigo 1 of 3")
    ("amigo2", Value<string>(), "Amigo 2 of 3")
    ("amigo3", Value<string>(), "Amigo 3 of 3")
//...
```



Example – setting value on command line



```
myapp \
    -listenEndpoints 127.0.0.1 \
    -emEndpoints 127.0.0.1:55100 \
    -amigo1 ned
```



Example – setting value in config file

```
# techtalk.conf  
amigo2 = lucky  
verbosity = 0  
#...
```



Example – setting value in environment

```
set DEMO_amigo3=dusty
```





Example – using command line, config file, and/or environment

- Config values can be accessed with the subscript operator (“[]”)

```
using std::string, std::cout, std::endl;

ApplicationConfiguration config(
    argc, argv, "demo.conf", "", "DEMO");

string amigo1(config["amigo1"].getValue<string>());
string amigo2(config["amigo2"].getValue<string>());
string amigo3(config["amigo3"].getValue<string>());

cout << "\namigo1: " << amigo1
    << "\namigo2: " << amigo2
    << "\namigo3: " << amigo3 << endl;
```

Example – using command line, config file, and/or environment

Output

```
> myApp -listenEndpoints 127.0.0.1 -  
emEndpoints 127.0.0.1:55100 -amigo1 ned
```

```
amigo1: ned  
amigo2: lucky  
amigo3: dusty
```

- amigo1 (“ned”) came from the command line
- amigo2 (“lucky”) came from the config file (techtalk.conf)
- amigo3 (“dusty”) came from an environment variable (DEMO_setting3)





Example – using KeyValueList (e.g., in a GUI app)

```
using namespace TENA::Middleware::Utils;

// Instantiate key/value list
BasicConfiguration::KeyValueList kvList;

// Add key/value pairs to list
BasicConfiguration::KeyValuePair endpoints("listenEndpoints", endpointsString);
kvList.push_back(endpoints);

BasicConfiguration::KeyValuePair appName("applicationName", "myApp");
kvList.push_back(appName);
// ...

// Instantiate app config with key/value list
TENA::Middleware::ApplicationConfiguration config(kvList, usageHeader);

// Use app config to initialize Middleware and instantiate the runtime
runtime = TENA::Middleware::init(config.tenaConfiguration());
```

Middleware configuration parameters

- TENA apps support config parameters that control runtime behavior
 - Where possible, default values are set to provide optimal WAN behavior
- All apps must define the value for `listenEndpoints`

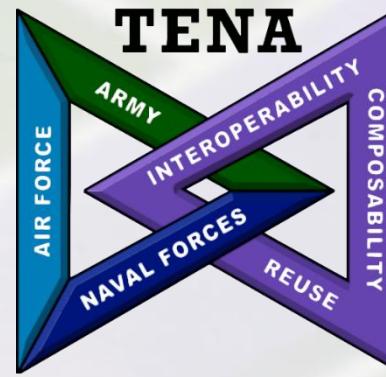
- `listenEndpoints`
- `configFile`
- `optionalConfigFile`
- `configFilePrefix`
- `applicationName`
- `multicastInterface`
- `multicastTTL`
- `multicastSendBufferSize`
- `multicastReceiveBufferSize`
- `bestEffortFragmentSize`
- `connectionTimeoutInMilliseconds`
- `twoWayTimeoutInMilliseconds`
- `logDir`
- `noErrorLog`
- `enableStructuredExceptionTranslation`
- `enableTENAdiagnostic`
- `enableDIMEtrace`
- `disableAlertOnError`
- `disableAlertOnWarning`
- `disableAlertOnNote`
- `popupWindowsDisabled`
- `windowsPeriodicTimerResolutionInMilliseconds`
- `dispatchDurationInSeconds`
- `corbaTransientRetries`
- `transientCommunicationAttempts`
- `announceTransientRetries`
- `transientAnnounceAttempts`
- `cannotContactEMretryTimeLimitInSeconds`
- `requestUserOnewayAck`
- `requestPublisherAck`
- `requestDataAck`
- `waitForMissingEventsIntervalInSeconds`
- `numORBthreads`
- `ftPrimaryIDwaitIntervalInSeconds`
- `ftPrimaryIDmaxWaitCount`
- `mwPushTimeoutInMilliseconds`
- `disconnectTimeoutInMilliseconds`



Middleware configuration – Links

- Configuration Mechanism

- </wiki/display/MW/Configuration+Mechanism>
- Middleware Configuration Parameters
 - </wiki/display/MW/Middleware+Configuration+Parameters>
 - Recommended Parameters for Working LAN Environment
 - </wiki/display/MW/Recommended+Parameters+for+Working+LAN+Environment>



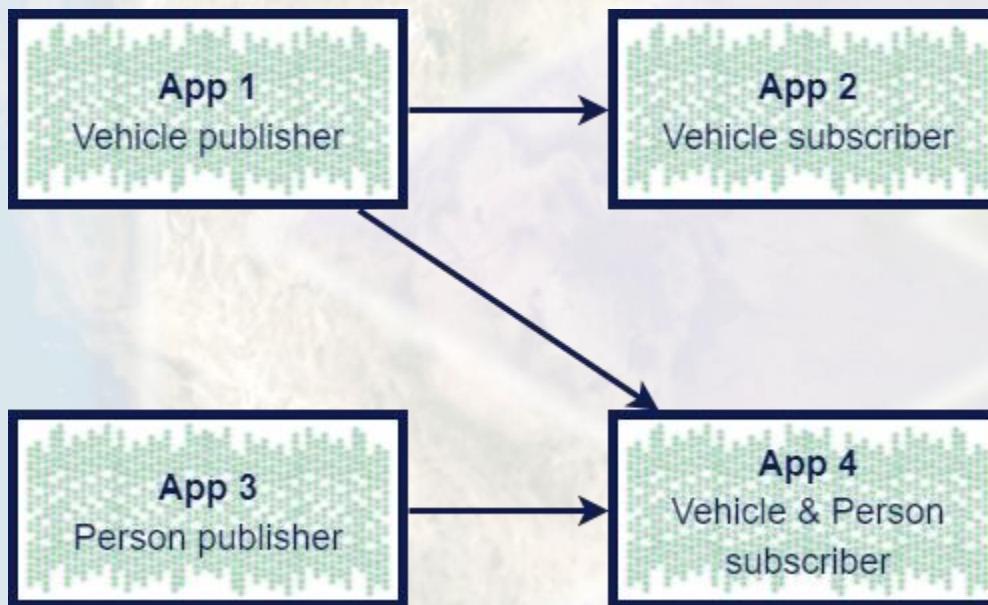
Advanced filtering

Advanced filtering

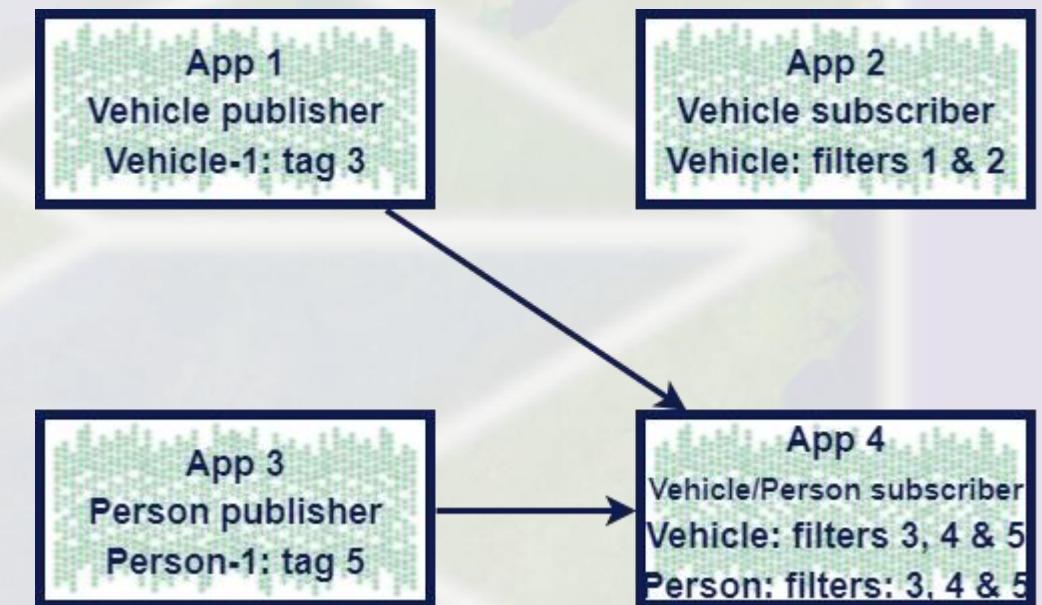
- Publish/subscribe behavior operates on matching the SDO/Message type being published with the type interest for the subscriber
- Advanced filtering extends type-based filtering to allow publishers and subscribers to provide additional filtering criteria
 - May improve network utilization and minimize unnecessary processing
- Publication “tags” & subscription “filters”
 - Publishers using advanced filtering apply a “tag” when creating objects
 - Subscribers using advanced filtering apply a “filter” when subscribing
 - Tags and filters are implemented using a simple integer value

Type-based publication/subscription

- Without tags/filters



- With tags/filters





Publication tags

```
std::unique_ptr<Example::Vehicle::SDOTag>  
tag(std::make_unique<Example::Vehicle::SDOTag>(42));
```

```
Example::Vehicle::ServantPtr servant =  
pServantFactory->createServant(  
*initializer,  
std::move(tag),  
communicationProperties);
```

Note1: SDO tags can be changed after servant creation, using the “setTag()” method.

Note2: SDO tags can also be automatically computed based on state changes:

See <https://www.trmc.osd.mil/wiki/display/MW/Computing+Tag+based+on+State+Update> for more details

Note 3: Message tags are specified when a message sender is created.



Subscription filters

- Subscribers using advanced filtering pass a “filter” to the subscription
- The filter object can be one of three kinds:
 - Wildcard – an all-pass filter (default – purely type-based subscription)
 - Tag – matches objects having the given tag
 - No tag – matches objects that have no tag
 - (Use “TENA::Middleware::Filter::MatchUntagged” as the filter value)

Note: The current Advanced Filtering capability does not support the ability to provide a set of filtering tags in a subscription, but users can perform multiple subscriptions if necessary.



Subscription filters



// Declare app's interest in Vehicle SDO's with tag 42

```
TENA::Middleware::Filter filter(42);
```

```
Example::Vehicle::subscribe(  
    session,  
    vehicleSubscription,  
    false /* no self-reflection */,  
    filter);
```

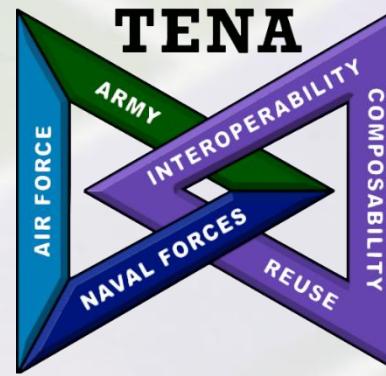


Advanced filtering – Links



- Advanced Filtering

- [/wiki/display/MW/Advanced+Filtering](#)
- Filter
 - [/wiki/display/MW/Filter](#)
- Tag
 - [/wiki/display/MW/Tag](#)
 - Computing Tag based on State Update
 - [/wiki/display/MW/Computing+Tag+based+on+State+Update](#)



Middleware IDs



Middleware IDs

- Many of the TENA API objects contain IDs that can be used for app programming needs
- These IDs provide an interface to support:
 - Comparison operations
 - Conversion to numerical & text representations
- Apps can use IDs to keep track of various TENA-related objects
 - Because they support comparison, they can be used in sorted containers



Middleware IDs

- Runtime ID
 - struct of 4 uint32
- Execution ID
 - struct of 4 uint32
- Session ID
 - uint32
- (SDO) Object ID
 - struct of 4 uint32
- Message sender ID
 - struct of 4 uint32
- Message ID
 - struct of 4 uint32 and uint32
- Type ID
 - uint64
- Application ID
 - uint32



Middleware IDs – common operations

- IDs have methods
 - `getValue()`: Returns the underlying representation (e.g., `uint32`)
 - `getBytes()`: Returns the value as a byte array
 - `toHex()`: Returns a hexadecimal string representation of the ID
- IDs can be constructed from:
 - Instance of their underlying type
 - Array of bytes (array of `uint8`)



Middleware IDs - `getID().toHex()` examples

```
using std::cout, std::endl, TENA::Middleware::SessionPtr;

SessionPtr session0(execution->createSession("session0"));
SessionPtr session1(execution->createSession("session1"));
SessionPtr session2(execution->createSession("session2"));

cout << " Runtime ID: " << runtime->getID().toHex() << endl;
cout << "Execution ID: " << execution->getID().toHex() << endl;
cout << "Session-0 ID: " << session0->getID().toHex() << endl;
cout << "Session-1 ID: " << session1->getID().toHex() << endl;
cout << "Session-2 ID: " << session2->getID().toHex() << endl;
```

Output

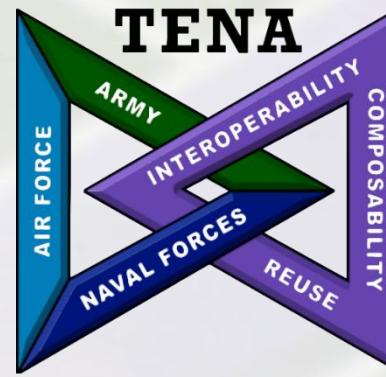
```
Runtime ID: 0xc80a010a00005448625ec66600000000
Execution ID: 0xc80a010a00004ce062589ae100000000
Session-0 ID: 0x00000000
Session-1 ID: 0x00000001
Session-2 ID: 0x00000002
```



Middleware IDs – Links

- Middleware IDs
 - [/wiki/display/MW/Middleware+IDs](#)





Middleware Metadata

Middleware Metadata

- Information related to Middleware status
 - Associated with the various objects that correspond to the API
- Related to the state of:
 - The app as whole
 - Specific SDOs and messages managed by the app
- Available for the following TENA Middleware classes
 - Runtime
 - Execution
 - Session
 - SDO
 - MessageSender
 - Message



Runtime metadata – examples

- `getID()`
 - ID that represents the runtime
- `getMiddlewareVersion()`
 - version of the middleware
- `getApplicationEndpointsString()`
 - Listen endpoints that the app uses to listen for incoming communication
- `getProcessID()`
 - ID used by the OS to represent the process associated with the app

Runtime metadata – examples

```
TENA::Middleware::Runtime::MetadataPtr rmd(runtime->getMetadata());  
cout << "\nSome runtime metadata..."  
    << "\n    Endpoints: " << rmd->getApplicationEndpointsString()  
    << "\n            ID: " << rmd->getID().toHex()  
    << "\n    MW version: " << rmd->getMiddlewareVersion()  
    << "\n    Process ID: " << rmd->getProcessID() << endl;
```

Output...

Runtime metadata

Endpoints: {[10.1.10.200:29729]}

ID: 0xc80a010a00008344625edddff00000000

MW version: 6.0.8

Process ID: 33604

Execution metadata – examples

- `getID()`
 - ID representing execution
- `getApplicationID()`
 - ID that represent the app that is joined to the execution
- `getUserSiteID()`
 - App-specified the user site ID (default zero (0) if not specified)
- `getUserApplicationID()`
 - App-specified the user site ID (default set to middleware-generated app ID if not specified)
- `getExecutionManagerEndpointsString()`
 - String representation of EM endpoints
- `getMulticastProperties()`
 - Multicast address/port range used by the execution

Execution metadata – examples

```
TENA::Middleware::Execution::MetadataPtr emd(execution->getMetadata());  
cout << "\nSome execution metadata..."  
    << "\n                ID: " << emd->getID().toHex()  
    << "\n                App ID: " << emd->getApplicationID().getValue()  
    << "\n                User site ID: " << emd->getUserSiteID()  
    << "\n                User app ID: " << emd->getUserApplicationID()  
    << "\n EM endpoints string: " << emd->getExecutionManagerEndpointsString()  
    << "\nMulticast properties: " << emd->getMulticastProperties() << endl;
```

Output ...

Execution metadata

ID: 0xc80a010a00004ce062589ae10000000

App ID: 23

User site ID: 0

User app ID: 23

EM endpoints string: 10.1.10.200:55100

Multicast properties: 225.25.2.1:55200:64

Session metadata - examples

- `getServantCountByType()`
 - Map that indicates the number of servants for each type
- `getProxyCountByType()`
 - Map that indicates the number of proxies for each type
- `getMessagesSentByType()`
 - Map that indicates the number of messages sent for each type
- `getMessagesReceivedByType()`
 - Map that indicates the number of messages received for each type



SDO metadata – examples

- `getTimeOfCreation()`
 - Time that the SDO was created
- `getTimeOfDiscovery()`
 - Time the SDO was discovered
- `getTimeOfCommit()`
 - Time the SDO was updated
- `getTimeOfReceipt()`
 - Time the update was received by the subscribing app
- `getStateVersion()`
 - Counter that starts at one and is incremented each time the servant's state is updated



Message sender metadata – examples

- `getEndpointsString()`
 - Network endpoints that the app uses to listen for incoming communication
- `getTypeIDpath()`
 - Vector for each type in the inheritance hierarchy for the message sender
- `getMessageCount()`
 - Counter for the number of messages sent by this `MessageSender`
- `getCommunicationProperties()`
 - Type of communication used when sending messages with this `MessageSender`



Message metadata – examples

- `getSenderID()`
 - ID representing the `MessageSender` that sent this message
- `getMessageCount()`
 - Message count of the `MessageSender` that sent this message
- `getMulticastAddressString()`
 - Multicast address used to send this message
- `getTimeOfTransmission()`
 - Time the message was sent by the sending app
- `getTimeOfReceipt()`
 - Time the message was received by the subscribing app

Application-specific clock

- By default, TENA uses the OS clock for timestamps, but app developers can create their own clock class for use by TENA
- The clock is used for creating timestamps associated with:
 - Logging
 - Diagnostics
 - Metadata
- Examples: IRIG, GPS, etc.

Application-specific clock

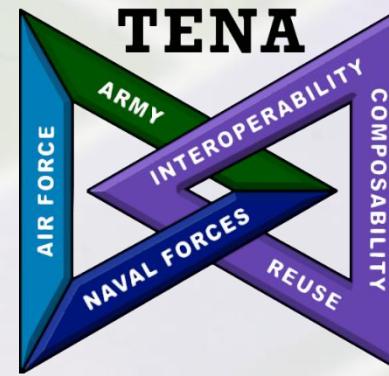
- App developers can create their own clocks by creating a class that inherits from the TENA MiddlewareTimeGenerator class
- Derived class only needs to implement the getTime() method
- Documentation and examples provided on the wiki



Middleware metadata – Links

- Middleware Metadata

- [/wiki/display/MW/Middleware+Metadata](#)
- Application-Specific Clock
 - [/wiki/display/MW/Application-Specific+Clock](#)



Handling exceptions

Runtime exception handling

- Handling exceptions is important for creating robust apps
- TENA uses exceptions to communicate situations requiring special handling back to the app
 - Apps need to “handle” these exceptions (using try/catch blocks)
 - Developers should determine if there are potential exceptions that should be wrapped in a try/catch block to prevent abnormal termination
- Review on wiki at
<https://www.trmc.osd.mil/wiki/display/MW/Runtime+Exception+Handling>

Runtime exception handling - examples

- **TENA::Middleware**
 - `init()` throws
 - `RuntimeAlreadyExists`
 - `joinExecution()` throws
 - `ExecutionManagerNotFound` (see example on next slide)
 - `NoLocalClassMethodsFactory`
 - `IncompatibleMiddlewareVersion`
 - `ObjectModelTypeInconsistencyError`
- **ServantFactory**
 - `createServant()` throws
 - `ExecutionNotConfiguredForBestEffort`

Runtime exception handling – example

```
TENA::Middleware::ExecutionPtr execution;
try {
    execution = runtime->joinExecution(epVector);
}
catch (TENA::Middleware::ExecutionManagerNotFound const & ex) {
    runtime.reset();
    cout << "EM not found: " << ex.what() << endl;
    return 0;
}

# Output ...
myapp -listenEndpoints 127.0.0.1 -emEndpoints 42.42.42.42:55100
```

EM not found: No executionManager could be found (after 3 attempts) listening to the endpoint(s) specified by {iiop://42.42.42.42:55100}. The endpoint(s) provided must reference an endpoint to which the desired executionManager is listening. Note that the executionManager prints out the listen endpoints. Please verify that the executionManager is running and listening to one of the specified endpoints. One reason this error may occur is that a firewall is preventing the application from establishing a TCP/IP socket connection. If the problem persists, report it to the TENA Helpdesk at <https://www.tena-sda.org/helpdesk/>.



Remote method exceptions – metamodel construct

- Allows SDO remote methods to throw custom exceptions
 - Exceptions can be caught – allowing the app to take appropriate action
- Should be used when an “exceptional situation” occurs



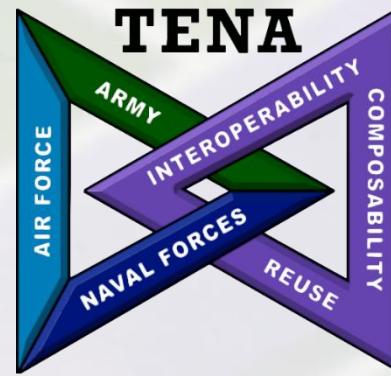
Remote method exceptions – metamodel construct

- A user defined exception example:
 - “exception MyException { string explanation; };
- Exceptions can only be structured with fundamental types (e.g., string)
- SDO remote methods can be declared to “raise” exceptions
 - “void myMethod() raises (MyException, MyOtherException);”



Handling exceptions – Links

- Runtime Exception Handling
 - [/wiki/display/MW/Runtime+Exception+Handling](#)
- Remote Method Exception Meta-Model Construct
 - [/wiki/display/MW/Remote+Method+Exception+Meta-Model+Construct](#)



Sending alerts



Next topic – Sending alerts

- The Middleware (associated with a TENA app) may generate alerts during some abnormal conditions
- By default, alerts will be...
 - Written to the standard output stream
 - Written to the app's log file
 - Sent as message to TENA Consoles in the execution



Sending alerts – configuration parameters

- noErrorLog
 - Disable error log file (messages will only go to output stream)
- logDir <location>
 - Directory for log files (default: TENA_HOME/TENA_VERSION/log)
- disableAlertOnError
 - Disables sending of alert error messages to consoles
- disableAlertOnWarning
 - Disables sending of alert warning messages to consoles
- disableAlertOnNote
 - Disables sending of alert note messages to consoles



Sending alerts – custom app alerts

- Apps can use alert mechanism to generate custom alerts
 - Errors
 - Warnings
 - Notes – used to report useful information that does not indicate a problem
 - “Out” (only works locally and does not send alert messages to TENA Consoles)
- Error, Warning, and Note alerts are automatically “decorated” with
 - The date and time the alert was issued
 - The file name and line number where the alert was generated
 - The “Out” construct does not provide any “decoration”

Sending alerts – example 1

```
TENA_OUT << "Heading toward the stairs" << endl;
TENA_NOTE << "Reached the top of the stairs" << endl;
TENA_WARNING << "Losing balance" << endl;
TENA_ERROR << "Help! I've fallen and can't get up!" << endl;
```

Output (terminal)...

...

Heading toward the stairs

NOTE: Reached the top of the stairs [2022-04-21T16:51:21.586162Z,C:\tmp42\Example-Vehicle-Subscriber-v7.6.0\src\main.cpp:105:main,TID=21152]

WARNING: Losing balance [2022-04-21T16:51:21.586162Z,C:\tmp42\Example-Vehicle-Subscriber-v7.6.0\src\main.cpp:106:main,TID=21152]

ERROR: Help! I've fallen and can't get up! [2022-04-21T16:51:21.587158Z,C:\tmp42\Example-Vehicle-Subscriber-v7.6.0\src\main.cpp:107:main,TID=21152]

...

Output (log file)...

...

Heading toward the stairs

NOTE: Reached the top of the stairs [2022-04-21T16:51:21.586162Z,C:\tmp42\Example-Vehicle-Subscriber-v7.6.0\src\main.cpp:105:main,TID=21152]

WARNING: Losing balance [2022-04-21T16:51:21.586162Z,C:\tmp42\Example-Vehicle-Subscriber-v7.6.0\src\main.cpp:106:main,TID=21152]

ERROR: Help! I've fallen and can't get up! [2022-04-21T16:51:21.587158Z,C:\tmp42\Example-Vehicle-Subscriber-v7.6.0\src\main.cpp:107:main,TID=21152]

...



Sending alerts – example 1 – TENA Console

```
TENA_OUT << "Heading toward the stairs" << endl;  
TENA_NOTE << "Reached the top of the stairs" << endl;  
TENA_WARNING << "Losing balance" << endl;  
TENA_ERROR << "Help! I've fallen and can't get up!" << endl;
```

TENA Console v1.0.27, Connected to EM 0 at 10.1.10.200:55100

[File](#) [View](#) [Help](#)

[EM Running](#) [Execution](#) [Execution OM Stats](#) [Applications](#) [Network Monitoring](#) [Execution Managers](#) [Consoles](#) [Alerts](#)

Show Notes Show Warnings Show Errors
 Show Heartbeat Messages Apply search/replace to messages [\(click to change\)](#) [Select All](#) [Hide Selected](#) [Unhide All](#)

...	ID	Type	...	Source	Message
...	1	- Note	...	Console Alert	Connected to the Execution Manager
...	3	...	0	Note	...	Application Alert	Reached the top of the stairs [2022-04-21T16:51:21.586162Z,C:\tmp42\Example-Vehicle-Subscriber-v7.6.0\src\main.cpp:107: main,TID=21152]
...	3	...	0	Warning	...	Application Alert	Losing balance [2022-04-21T16:51:21.586162Z,C:\tmp42\Example-Vehicle-Subscriber-v7.6.0\src\main.cpp:107: main,TID=21152]
...	3	...	0	Error	...	Application Alert	Help! I've fallen and can't get up! [2022-04-21T16:51:21.587158Z,C:\tmp42\Example-Vehicle-Subscriber-v7.6.0\src\main.cpp:107: main,TID=21152]

Time Reported: 04/21/22 10:51:21.587-0600 Source: (Application Alert)
Type: Error Node ID: 3
Message:
Help! I've fallen and can't get up! [2022-04-21T16:51:21.587158Z,C:\tmp42\Example-Vehicle-Subscriber-v7.6.0\src\main.cpp:107: main,TID=21152]



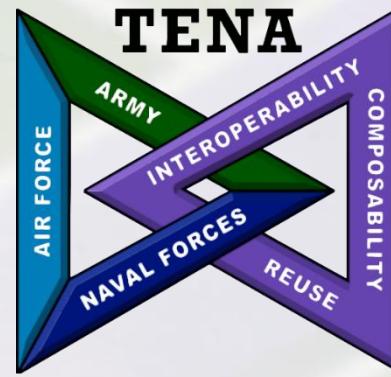
Sending alerts – example 2

```
catch(std::exception const & ex)
{
    TENA_ERROR << "Unexpected Standard Exception: "
                << ex.what()
                << std::endl;
}
```



Sending alerts – Links

- Alerts
 - trmc.osd.mil/wiki/display/MW/Alerts



Log files



Log files

- TENA apps create a log file whenever the app encounters an exception
- The exception message will also be
 - Written to the app's output stream
 - Sent to active TENA Consoles



Log files – configuration parameters



- **logDir <ARG>**
 - Directory for log files (default: TENA_HOME/TENA_VERSION/log)
- **noErrorLog**
 - Disables error log file (messages will only go to standard out)
- **applicationName <ARG>**
 - Specify app name as it appears in log file names



Log files - example 1

C:\TENA\6.0.8\log\InstrumentAssignment-GUI.exe-20220331-150351-17704-msgs.txt

Log file opened by 'C:/TENA/InstrumentAssignment-GUI-v3.3.0/bin/InstrumentAssignment-GUI.exe' at 2022-03-31T21:04:04.148260Z

...

Assigning POI "Boris" to R-2780-5331

Assigning POI "1979-921-32" to R-2780-5331

Calling stopFollowingThisPOI on "1979-921-32" for R-2780-5331

Assigning POI "1979-921-33" to R-2780-5331

Assigning POI "BadNews" to R-2780-5331

Assigning POI "1979-921-40" to R-2780-5331

...

Log file closed at 2022-04-01T15:49:34.070326Z



Log files - example 2

C:\TENA\6.0.8\log\TENA-StandardOMs-v2.0.0-DataCollection-v1.1.7-20220218-113715-16024-msgs.txt

Log file opened by 'C:/TENA/StandardOMs-v2.0.0-DataCollection-v1.1.7/bin/tenaCollector-TENA-StandardOMs-v2.0.0-v1.1.7.exe' at 2022-02-18T18:37:19.601796Z

NOTE: TENA-StandardOMs-v2.0.0 has started collecting data. [2022-02-18T18:37:19.600799Z,M:\autobuild\box712996\bld\332c91\TENA_StandardOMs-v2.0.0-DataCollection_2f104c\src\ApplicationMain.cpp:1673:TDCS::DataCollection::ApplicationMain::setupObservers,TID=19736]

NOTE: TENA-StandardOMs-v2.0.0 has stopped collecting data. [2022-02-18T18:38:13.881074Z,M:\autobuild\box712996\bld\332c91\TENA_StandardOMs-v2.0.0-DataCollection_2f104c\src\ApplicationMain.cpp:2727:TDCS::DataCollection::ApplicationMain::unsubscribe,TID=19736]

Log file closed at 2022-02-18T18:38:18.200307Z



Log files – example 3

C:\TENA\6.0.8\log\executionManager-20211207-104415-10976-msgs.txt

Log file opened by .../executionManager.exe ...*

...

NOTE: An application with the same endpoint ... attempting to register...*

WARNING: Application 7 -- No heartbeat received within 157.525 ...*

ERROR: Type consistency failure: Application listening at endpoint...*

...

NOTE: Shutting down..... [2021-12-08T19:40:13.330843Z]

NOTE: Shutdown complete. [2021-12-08T19:40:13.330843Z]

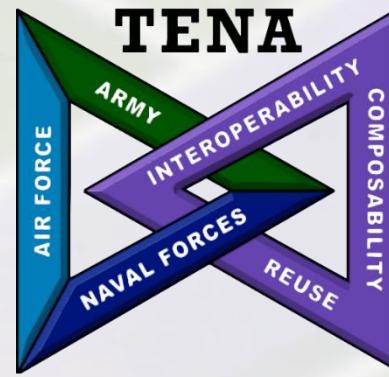
Log file closed at 2021-12-08T19:40:55.457286Z

* Details removed for brevity. Log files include details related to specific app IDs, endpoints, etc.



Log files - Links

- Log Files
 - [/wiki/display/MW/Log+Files](#)



TRMC website services



Topics

- Group basics – getting started
- Wiki/Confluence
- Email reflectors
- Other things that might be of interest
 - Helpdesk/Jira
 - Git/Bitbucket
 - Artifactory
 - Repository
 - Meeting registrations
 - Calendar



Group basics – getting started

- Website content is managed through user groups
 - All content belongs to one, and only one, user group
 - Content includes web pages, helpdesk cases, file attachments, repository products or OMs, source code, email reflectors, etc.
- Supports collaboration needs of an organization, company, organization, project, or product
- Classified data not supported
 - Do not include or attach classified data
- Supports requirements for Controlled Unclassified Information (CUI)



Group roles

- Members are assigned roles that are associated with permissions
 - Admin: can read, create, modify, delete, & restrict content – and manage group membership
 - Team: can read, create, modify, delete, & restrict content
 - User: can read content
- Admins can request additional roles to further restrict content



Open or closed access

- Open: all approved website users can access content
 - Recommend to promote awareness of various user group activities when appropriate
 - Team members can still restrict certain content in an open group
- Closed: group content limited to group members



Membership visibility



- Groups can be setup so that
 - Anyone may request membership (admins adjudicate requests), or
 - Membership is managed by admins



Wiki/Confluence

- Main link: <https://www.trmc.osd.mil/wiki/display/Training>
- Useful for sharing information
 - Create/organize web pages
 - Attach content
 - Restrict content as needed
 - “Watch” pages to be notified of changes
 - Wiki keeps track of page history



Email reflectors

- Email “reflectors” can be requested, for example:
 - training-team@trmc.osd.mil
- Not much to say
 - One person in the group sends an email to this address
 - Everyone else should receive that email

Other things that might be of interest

- Helpdesk/Jira
 - Issue tracker
 - Main link: <https://www.trmc.osd.mil/helpdesk>
- Git/Bitbucket
 - For managing software source code
 - Main link: <https://www.trmc.osd.mil/bitbucket>
- Repository
 - <https://www.trmc.osd.mil/wiki/plugins/repository.action>
- Artifactory
- Meeting registrations
- Calendar
- etc.