Galleon Whitepaper

NGVA Data Model and OpenDDS

Practical example on how to use the NGVA Data Model and OpenDDS

Revision history:

Rev.	Date	Changes	Sign
1.0	7-JUNE-21	Initial release	ØAS

Abstract

This document describes how to use the NGVA Data Model in OpenDDS, and provides a practical example using OpenDDS and the NGVA Data Model to control the Galleon Video Recorder. Everything related, from converting the IDL files provided by the NGVA Data Model to C++ code, building OpenDDS and creating applications to control the Galleon Video Recorder using OpenDDS, is discussed.

1 Introduction

There are few, if any, existing online resources covering DDS and the NGVA Data Model. This whitepaper depicts a practical example of how to make use of the NGVA Data Model and OpenDDS in a sub-system in a military land vehicle. In this example the NGVA Data Model and OpenDDS will be used to control the Galleon Video Recorder. More specifically to start a video recording, and start/stop a video stream.

This document is aimed at developers seeking information on how to use the NGVA Data Model and DDS, but the main concepts should be possible to understand without any software developer background. First a short introduction on NGVA is given, followed by an introduction of DDS, OpenDDS and the NGVA Data Model. Then, the work completed to build OpenDDS and to generate C++ code from the NGVA Data Model is presented, followed by example applications displaying how to use OpenDDS and the Data Model. At the end of the document a short summary is found.

NGVA, or NATO Generic Vehicle Architecture, is defined in STANAG 4754. It is an open standard that aims to provide simpler, more cost effective and more agile integration of sub-systems in military land vehicles. It also aims reduce integration risks and deployment time. NGVA defines and standardizes the:

- Architecture Approach
- Power Infrastructure
- Data Infrastructure
- Crew Terminal Software Architecture
- Data Model
- Safety
- Verification and Validation

This document will only deal with the data exchange mechanism (which is a part of the Data Infrastructure) and the Data Model parts of the standard.

DDS – the data exchange mechanism

NGVA utilize DDS, or Data Distribution Service, to exchange data between sub-systems. DDS is a middleware for data-centric sharing of data between different machines and applications, which implements a publish-subscribe pattern. DDS handles message addressing, data marshalling (and de-marshalling), flow control, retries, etc. Different implementations of

DDS exist, some commercial, and some open-source/free. NGVA requires the DDS implementation to follow the DDS Interoperability wire protocol, DDSI-RTPS, to ensure communication between sub-systems using different DDS vendors.

Important DDS concepts to be familiar with are publisher, domain, subscriber and topic. A publisher is publishing data (of a specific type) to a specific topic in a given DDS domain. A DDS domain can be viewed as a subnet on the network. Only applications belonging to the same domain can interact. A subscriber is subscribing to a specific topic in a given DDS domain and reads data from that topic every time new data is written to it. A topic has a unique name within the domain and is of a specific data type(structure).

OpenDDS

OpenDDS was selected as the DDS implementation for this project. This implementation supports DDSI-RTPS and is open source and free of licensing fees. OpenDDS places no obligation on users to redistribute any of their source code. For full licensing information visit OpenDDS' website (https://opendds.org/about/license.html).

NGVA Data Model

The NGVA Data Model defines the data structures and formats to be used by the sub-systems communicating via DDS in the military land vehicle. The Data Model consists of several modules, e.g. Alarms, Power, LDM Common, Video, Automatic Weapon, Usage and Condition Monitoring and many more. Each sub-system will usually only implement a subset of the modules. Each module contains an UML model that can be translated into an Interface Description Language (IDL) file using translators provided by NGVA. This IDL file can later be converted to code (typically C++ code), by compilers provided by the DDS implementation supplier. This document will use the Video module and the LDM Common module from the NGVA Data Model.
The LDM Common module contains data structures shared between all the modules, while the Video module contains data structures used to control video devices.

2 Work package

In order to control the Galleon Video Recorder over DDS using the NGVA Data Model the following work had to be completed:

- Build OpenDDS
- Generate C++ code from the Video module and the LDM Common module in the NGVA Data Model.
- Create three applications publishing and subscribing to different DDS topics
    - One video streaming application
    - One video recording application
    - One controller application that communicates with the video streaming application and the video recording application over DDS

An important note is that all of this work is conducted on a system running CentOS Linux 7.8.

2.1 Build OpenDDS

Version 3.16 of OpenDDS was downloaded from the official OpenDDS website (https://opendds.org/) and copied to an appropriate folder on the system. The following commands were executed in order to build it:

- Navigate to the folder containing the OpenDDS release (OpenDDS-3.16.tar.gz)
- *$ tar –zxvf OpenDDS-3.16.tar.gz*
- *$ cd OpenDDS-3.16*
    - The path to this folder is later referred to as <opendds>
- *$ su*
    - Type 'root' password
- *$ yum install perl perl-Data-Dumper perl-Env*
- *$ ./configure*
- *$ make*

2.2 Generate code for the Video and LDM Common module in the NGVA Data Model

When the Video and LDM Common module is downloaded from the official NATO GVA website (https://www.natogva.org/data-model/), the modules always contain the UML models needed to generate IDL files and later code in the language of your choice. Usually, the modules also contain IDL files for one or more DDS suppliers. The modules used here did not contain IDL files for OpenDDS, but they had IDL files for RTI Connext. It was possible to use these IDL files with OpenDDS by implementing some small modifications. “@topic” had to be added before each of the data structures that was used as topics in the applications. Figure 1 depicts “@topic” added before a data structure.

```
@topic
struct C_Actual_Video_Source
{
    P_LDM_Common::T_IdentifierType A_sourceID;
    P_LDM_Common::T_DateTimeType A_timeOfDataGeneration;
    T_EncodingType A_colourModel;
    sequence <P_LDM_Common::T_IdentifierType> A_providesVideoStream_sourceID;
    P_LDM_Common::T_IdentifierType A_specification_sourceID;
};
```

Figure 1. Example on "@topic" added before struct

The IDL files for the Video and LDM Common module, Video_PSM.idl and LDM_Common.idl, were copied to a folder on the system (path to folder later referred to as <idl>) and “@topic” was added before
the following structures: C_Actual_Video_Source_setColourModel, C_Actual_Video_Source, C_Actual_Video_Stream_cancelVideoStream, C_Actual_Video_Stream_requestVideoStream, C_Actual_Video_Stream, C_Actual_Video_Sink_setVideoStream, C_Actual_Video_Sink, C_Video_Source_Specification and C_Video_Stream_Specification. To see the fields in each structure, open the IDL files and have a look.

Because the Galleon Video Recorder has a C++ API, the IDL files needed to be converted to C++ code. The IDL files were converted by executing the following commands:

- *$ <opendds>/ACE_wrappers/bin/tao_idl --idl-version 4 -o <output> <idl>/LDM_Common.idl*
- *$ <opendds>/ACE_wrappers/bin/tao_idl --idl-version 4 -o <output> -I<idl> <idl>/Video_PSM.idl*
- *$ <opendds>/dds/idl/opendds_idl --idl-version 4 -o <output> <idl>/LDM_Common.idl*

- *$ <opendds>/dds/idl/opendds_idl --idl-version 4 -o <output> -I<idl> <idl>/Video_PSM.idl*

- *$ <opendds>/ACE_wrappers/bin/tao_idl --idl-version 4 -o <output> -I<opendds>/ -I<idl> <output>/Video_PSMTypeSupport.idl*

<output> is the path to the folder for the output (C++ code).

For an in-depth description of the IDL processing above, have a look in the OpenDDS Developer's Guide found on the OpenDDS website (http://download.objectcomputing.com/OpenDDS/OpenDDS-latest.pdf).

2.3 Create applications publishing and subscribing to DDS topics

As mentioned earlier, three applications were created. One for controlling the video stream, publishing information about it to DDS topics and subscribing to DDS topics in order to receive start/stop requests for the video stream. Another application for controlling the video recording, publishing information about it to DDS topics and subscribing to DDS topics in order to receive start requests for the video recording. The third application subscribes to DDS topics to receive information about the available video stream and recording, and based on this information it publishes to DDS topics in order to request to start recording and start/stop streaming. Figure 2 illustrates how the applications communicate using DDS.

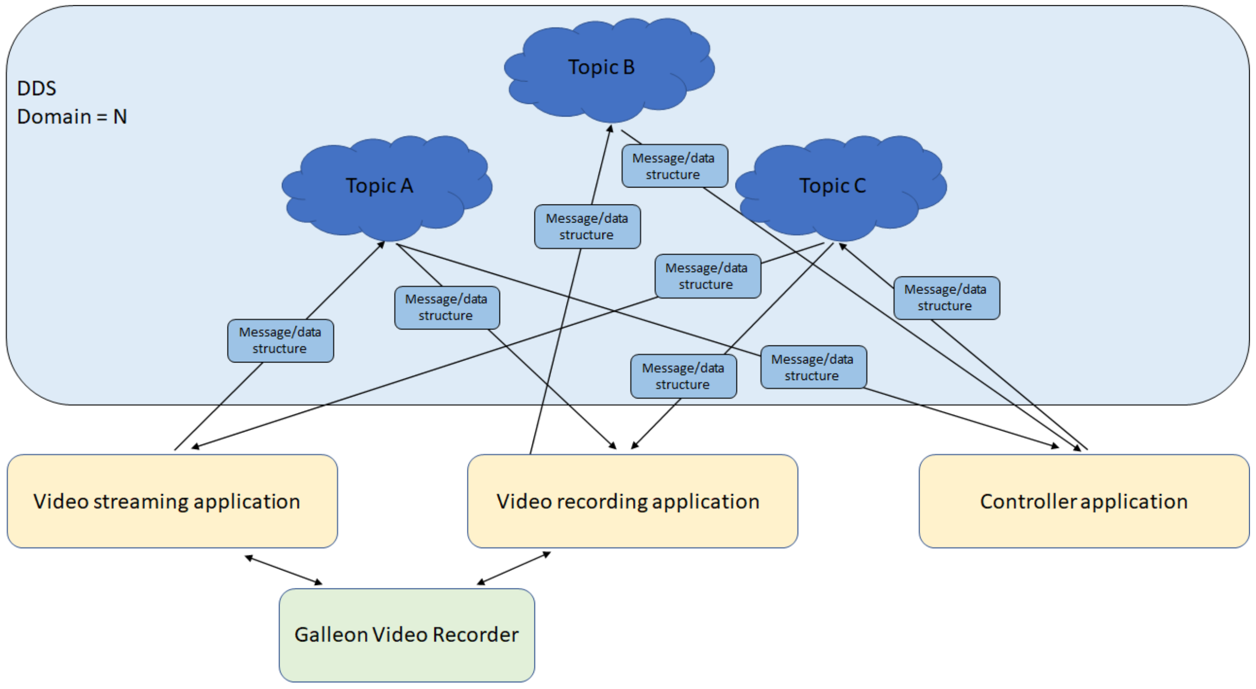

Figure 2. System overview

Each existing topic was named identically to its data structure, as required by NGVA.

Video streaming application

This application publishes to the following topics; C_Actual_Video_Source, C_Video_Source_Specification, C_Actual_Video_Stream and C_Video_Stream_Specification with given ID's. This makes it possible for other applications in the same DDS domain that subscribes to the same topics to find the information and make use of it.

The application does also subscribe to the C_Actual_Video_Stream_cancelVideoStream and C_Actual_Video_Stream_requestVideoStream topics. If an application is publishing to the C_Actual_Video_Stream_requestVideoStream topic with an ID matching the video streaming application's stream ID, the video streaming application will make the Galleon Video Recorder start streaming video. Similar, if an application is publishing to the C_Actual_Video_Stream_cancelVideoStream topic with an ID maching the video streaming application's stream ID, the video streaming application will make the Galleon Video Recorder stop streaming video.

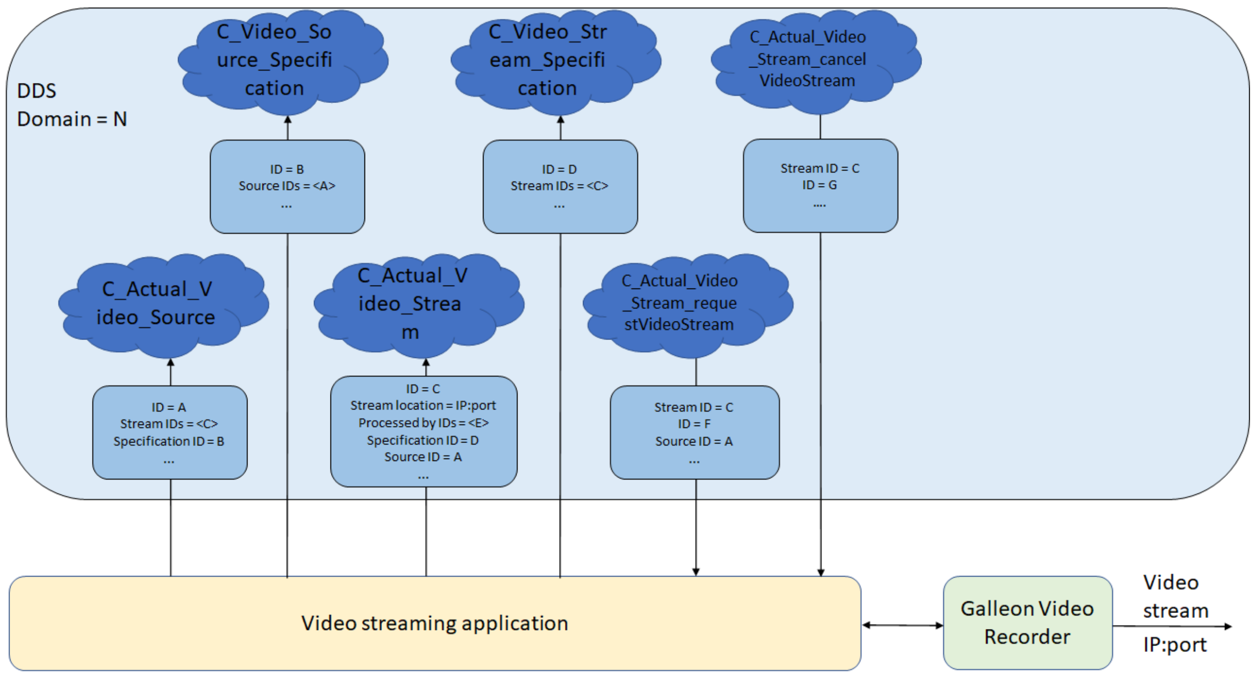

Figure 3. Video streaming application overview

Video recording application

This application publishes to the C_Actual_Video_Sink topic with a given ID. This makes it possible for other applications in the same DDS domain that subscribes to this topic to find the information and make use of it.

The application does also subscribe to the C_Actual_Video_Sink_setVideoStream, C_Actual_Video_Stream and C_Video_Stream_Specification topics. If an application is publishing to the C_Actual_Video_Sink_setVideoStream topic with an ID matching the video recording applications sink ID, the video recording application will read from the C_Actual_Video_Stream and C_Video_Stream_Specification topics to find information about the video stream with an ID matching the stream ID received from the C_Actual_Video_Sink_setVideoStream topic. When the information is found, the recording is created and started using the Galleon Video Recorder.

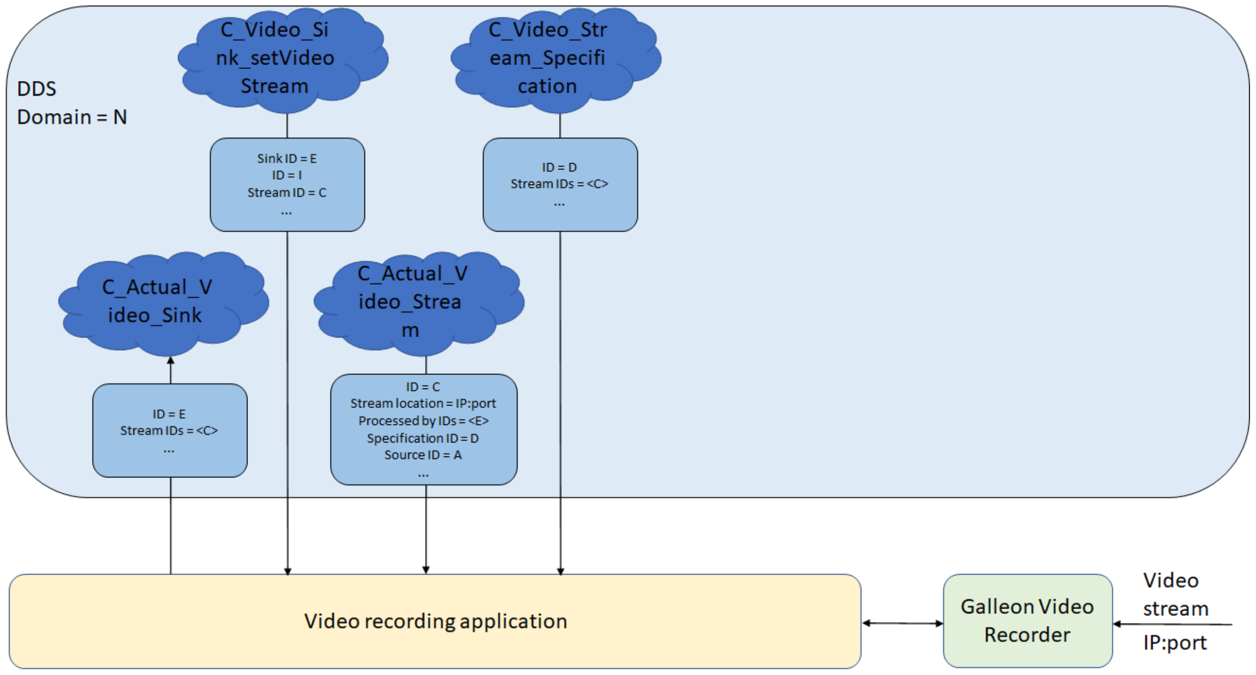

Figure 4. Video recording application overview

Controller application

This application subscribes to the C_Actual_Video_Sink and C_Actual_Video_Stream topics, and publishes to the C_Actual_Video_Sink_setVideoStream, C_Actual_Video_Stream_requestVideoStream and C_Actual_Video_Stream_cancelVideoStream topics.

The application will first wait on data from the C_Actual_Video_Sink and the C_Actual_Video_Stream topics in order to find the IDs for the video stream and the video sink (recording). When the IDs are found the video stream is added to the video sink by publishing to the C_Actual_Video_Sink_setVideoStream topic with the received video sink ID and video stream ID. The video recording application will receive this message and start recording. After that this application publishes to the C_Actual_Video_Stream_requestVideoStream and C_Actual_Video_Stream_cancelVideoStream topics with the received stream ID. The video streaming application will then receive this message and start/stop the video stream using the Galleon Video Recorder.

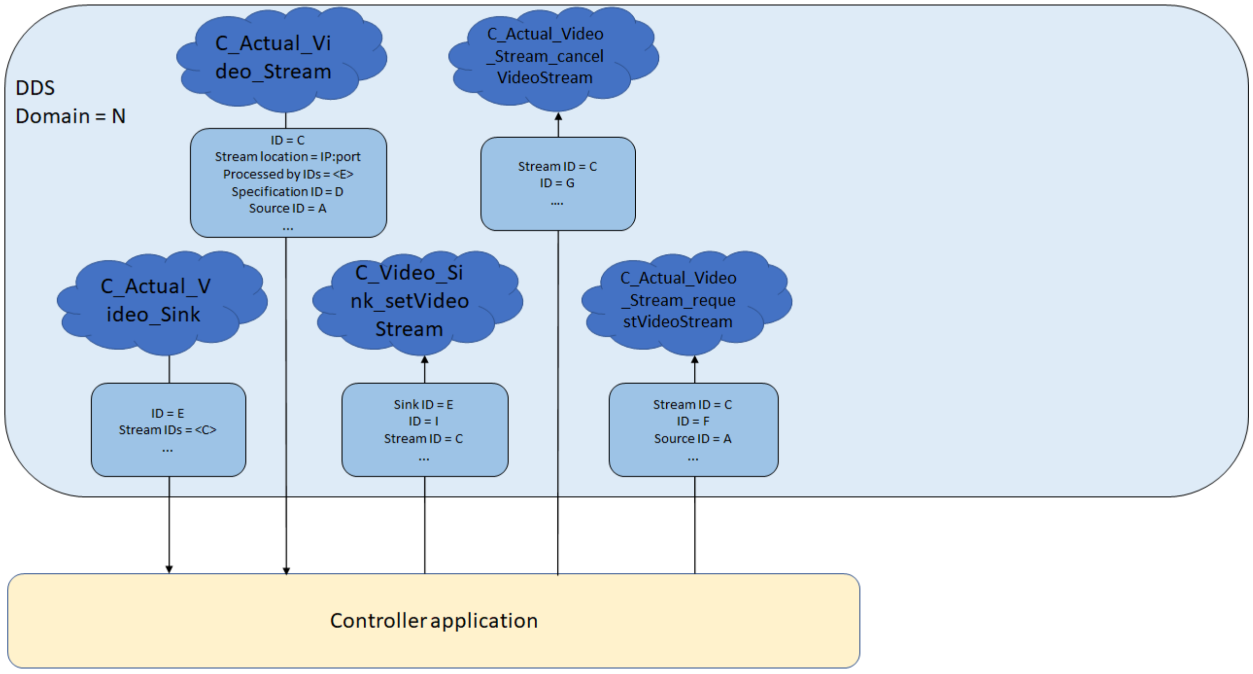

Figure 5. Controller application overview

3 Summary

This work shows that the NGVA Data Model and OpenDDS can be used to control a sub-system on a military land vehicle, like the Galleon Video Recorder. The concept of DDS is easy to understand, and it is easy to make use of DDS in an application. How complex it is to use the NGVA Data Model depends on whether the provided IDL files can be used, or if the IDL files have to be generated from the UML models using NGVA translators. This depends on the chosen DDS implementation.

4 Feedback and References

4.1 Feedback

Any questions or comments to the contents are welcome and appreciated. Please contact Espen Bøch, eboch@galleonec.com or send your feedback to info@galleonec.com

4.2 References

https://www.natogva.org/overview/
https://www.dds-foundation.org/what-is-dds-3/
https://www.natogva.org/data-model/
https://www.galleonec.com/project/xsr-hd-video-recorder/
http://download.objectcomputing.com/OpenDDS/OpenDDS-latest.pdf
https://opendds.org/about/license.html