WHITEPAPER

DDS: The Data Backbone of SDV Interoperability

AUTHOR

Neil Puthuff
Staff Application Engineer
Real-Time Innovations (RTI)

Software-defined vehicle (SDV) development, while still promising, has hit a speed bump. Much like the concept of SAE Level 4+ autonomy, the ideal of a data-driven SDV is fairly easy to describe yet considerably more difficult to build. One big reason: the complexity of systems integration.

At its core, the SDV is driven by data. It involves massive amounts of data from countless sources (control signals, sensors, passenger input, internal components, external networks, etc.), each with their own specific requirements to be sent exactly where and when they're needed in real time.

Why is this so challenging?

First, there's the data itself: How is it represented? As a text string, an integer, a floating-point number? What are the units? Is it absolute or relative? Is there an offset or scaling factor?

Second, where does this data come from? Ideally the data would be available anywhere it may be needed in the vehicle, but this also requires that its source be discoverable and understood.

Third, how is this data accessed? Is it done using a service-oriented architecture (SOA), a vendor-specific API, or one of the community-developed frameworks such as AUTOSAR, Eclipse SDV, or others? Does it require tight coupling between the components, where a small change in one interface can require a cascading assortment of changes elsewhere in the system?

Vehicles are built of components sourced from an ecosystem of suppliers and integrators, each having their own preferences and each vying to create a better solution and win the most business. While this sounds like a healthy, Darwinian way of producing the best systems, in reality these 'best of' subsystems must be digitally welded together, posing new complexities and costs. While it's common to create a robust, reliable system from different components that use different data types and have different methods of access, it comes at a cost.

Automotive industry organizations have started to address this by creating a variety of common software API standards for component communications. Yet these approaches may not address the format and type issues for the underlying data, and they still leave developers with many of the same system integration challenges as before.

These API's can be viewed as layers of abstraction atop the underlying data that is crucial to an SDV. Focusing on the data itself via a data-centric approach can help streamline the complexity of system integration. This work to build and incorporate open source, common data centric models for present and future vehicle design is now underway at COVESA.

A DATA-CENTRIC APPROACH

Let's get back to the ideal of a data-driven SDV, but with three important additions:

1. **All data uses a common type and format standard.** This includes all data in and around the vehicle — sensors, actuators, software data objects — for all systems: Cabin, Chassis, ADAS, Infotainment, V2X, etc.

2. **The data itself is the interface between components.** For instance, a perception system might expect input from camera, LiDAR, radar, GNSS and IMU using the standard data types, and would export a list of perceived objects and their location and motion while also using the standard data type definitions.

3. **There is a 'Global Data Space' inside the vehicle.** This means that every ECU and processing node in the vehicle has full access to all of the data in the global data space.

In other words, every component and application in the vehicle 'speaks' the same common data language used in all vehicles. With these additions, the task of creating and integrating a SDV becomes a lot easier:

- Software functions that work with these data types can be placed anywhere in the vehicle – any ECU, any compute node.

- Components can be freely swapped or upgraded without the usual integration headaches.

- Components that work for today's vehicle can continue to work in future vehicles.

- Components can be created independently of any make or model – they are built to the data, not the vehicle.

In this framework, the character and capabilities of a vehicle can be completely defined by the software, now and in the future, as long as the common data language is still used.

Such a system is not only possible, but actively being worked on today. With COVESA and DDS™, these data-centric systems are on their way to being created, tested and deployed. COVESA is taking a leadership role in moving data-driven SDVs forward and expanding their capabilities via open source/open standards and with commercial support.

COVESA, or the Connected Vehicle Systems Alliance, is an open, collaborative technology alliance that brings together organizations globally to develop common approaches and innovative technologies for connected vehicles. It is also a founding member of the SDV Alliance, along with AUTOSAR, Eclipse SDV and SOAFEE. One of COVESA's main focal areas is the definition of a catalog of common data types contained in its Vehicle Signal Specification (VSS). Using the COVESA VSS enables software components to be created independently, while leveraging the interoperability of speaking a common data language.

DDS, or Data Distribution Service, is an open Object Management Group® (OMG®) standard for data-centric communications. With DDS, the data itself is the interface between system components. It uses a publish-subscribe or request-reply pattern, with automatic discovery, independence from the underlying transport (network, memory, radio, serial), granular security and a rich set of Quality-of-Service (QoS) capabilities that enable it to operate in the harshest of environments. DDS offers extremely high performance using an efficient binary data encoding, and is able to take advantage of the underlying transport capabilities (i.e., multicast, zero-copy, lossless compression, etc.) without affecting code portability.

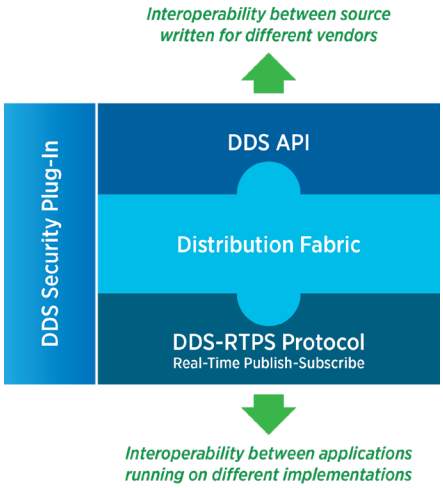

Figure 1: Data Distribution Service (DDS)

DDS: DATA-CENTRIC FRAMEWORK FOR SYSTEM INTEROPERABILITY

The basic operating tenets of DDS include:

- **A common set of user-defined data types** that embody all of the data that flows through the system, including radar tracks, driver controls, video feeds, sensors, actuators and much more. These same data type definitions are used throughout the system.
- **Publishers and subscribers** of the above data types. Software applications will either produce this data for publication or will subscribe to this data for consumption. In either case, the application will only interface to the data, regardless of its source, destination, or how it gets there. It could be on the same ECU, a different ECU, simulated, recorded or emulated, and the software applications don't need to know.
- **Discovery** of other DDS participants to locate and match the publishers and subscribers of every data type used in the system, creating the connections to enable data flow. Discovery works across all supported transports (UDP, Shared Memory, TCP, etc.) thus avoiding the need to make changes to the applications when the system is reconfigured.

The block diagram on page 3 will help in describing how DDS works.

The **DDS Core** libraries handle the **Discovery** of other DDS participants, implementing the **QoS** configuration, and **Security** if enabled, which can individually handle authentication, encryption and access control for each individual data flow.

The DDS Core then interfaces with the **Transport Abstraction** layer, which enables DDS to work over nearly any transport, including IP networks, shared memory, serial, backplane, etc. This is the final interface to the **Global Data Space** created by DDS: an abstraction wherein all data in the system is current and available; it provides a single point of truth for what's happening in the system.

This arrangement in DDS enables software components to be created without worrying about the underlying hardware, operating system, network or programming language. Complex systems can be built by completely independent teams, each taking their own best approach for each component while having the standards-based foundation of interoperability and performance.

More than a dozen implementations of DDS have been created, both open source and commercially supported, and these standards-conforming implementations are completely interoperable with each other. DDS is available for most popular programming languages and operating systems/RTOSs, and is one of the two communications frameworks of the AUTOSAR standard.

DDS offers the following capabilities:

- **Extreme performance** DDS operates at close to wire speeds over IP networks and even faster over shared memory.
- **Advanced QoS capabilities** to fine-tune the data flows and to ensure the system runs at peak performance under most conditions.
- **Standards-based interoperability** to ensure that different implementations of DDS can work together, avoiding vendor lock-in.

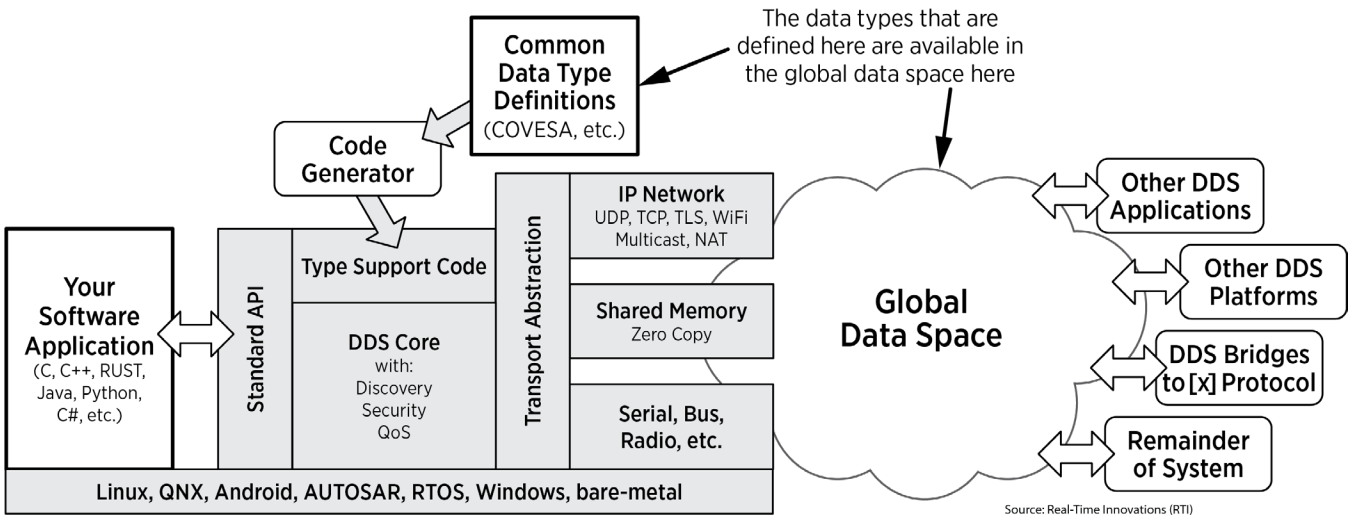

*Figure 2. How DDS works (left-to-right): **The Common Data Types** are sent to a **Code Generator** at software build-time, to produce the **Type Support Code** for those custom data types. This implements the **Software API** used by **Your Software Application** to create the **Publishers/Subscribers** of those custom data types.*

- **Easy to adopt** due to its extensive platform support; DDS runs on desktop (Linux, Windows, Apple), real-time (QNX, SafeRTOS, INTEGRITY, etc.) and automotive (AUTOSAR Classic and AUTOSAR Adaptive) platforms.
- **Easy to understand** due to its focus on the data itself. Complex systems are implemented as topic-based flows of data, without the layered complexity of Services, Registries, Protocols, etc.
- **Easy to use** due to its automatic discovery, global data space, and data centricity.
- **Easy to secure** due to its granular authentication / encryption / access control for each data flow, over secured (TLC, SSL) or unsecured (TCP, UDP, memory) transports.

When paired with a common set of data types like COVESA VSS, DDS can reduce system complexity and streamline development of SDVs and of achieving SAE L2-L5 autonomy. Here's how.

Ease of Integration via Data as the Common Interface

Since the data itself is the interface, DDS-based systems do not require the tight coupling between applications that is necessary with most other approaches.

For example, a perception component that imports camera and radar data and exports a list of identified objects would typically require substantial coding and testing effort to adapt the component API and data formatting to the rest of the system.

Using DDS for the middleware layer eliminates much of this work. The integration work between DDS and a common set of data types is minimized, as each component produces and is supplied with exactly the type of data it needs.

This loosely-coupled data centricity gives complete freedom to the component suppliers to use whatever technology works best: different programming languages, different processor types, different operating systems and different application frameworks. This approach helps to avoid vendor lock in for OEMs, while providing suppliers with the ability to use their preferred technology components. With DDS, the data is the common interface.

Application Portability

DDS creates a 'Global Data Space', where all data is available to all participants that are connected to the network. This makes it easier to relocate software applications within the system, which is an advantage when evolving a system from distributed control units through zonal controllers to centralized high-performance computing, and beyond.

Interoperability and the Automotive Ecosystem

Let's look at a scenario where components are built to use a common data model and DDS. This could enable drop-in, plug-and-play interoperability of components created through completely independent efforts. These components could be freely swapped and upgraded between suppliers and vehicles with fast, easy integration efforts that involve no custom engineering. Therefore, suppliers could build these components with ever-increasing value and reduced cost, instead of undertaking custom work for the specific makes/models that exist today. This model creates a robust and efficient ecosystem that maximizes value for all stakeholders.

With DDS, this scenario works now. One example is the ROS 2 project. ROS 2 uses DDS as its underlying communications framework and has both a very stable and rapidly expanding set of common data types. These two factors have enabled ROS to grow a vibrant ecosystem of thousands of interoperable software packages from low-level sensing and actuation to advanced perception, path planning, SLAM, and more.

Another example is AUTOSAR Adaptive, where the language-independent, service-oriented semantics of ara::com are mapped to the DDS standard type system and APIs via a DDS Network Binding. This integrates the rich set of DDS QoS policies into the AUTOSAR service-oriented architecture. This approach enables new interoperability scenarios in which AUTOSAR systems can be integrated into larger system-of-systems (SoS) environments through the use of the DDS databus, which distributes and manages real-time data in intelligent distributed systems.

Reduced Cost and Time to Market

The benefits of using DDS and common data types like COVESA VSS for SDV development can have a large impact on cost, time and value. COVESA VSS can help:

- Save time by reducing extensive systems integration efforts.
- Add value by re-using components across different makes, models and model years.
- Reduce cost by avoiding short-lived custom engineering work during system evolution.

Additional benefits include the efficiency improvements of reduced complexity, the strength of a common ecosystem and a deeper talent pool from which to attract skilled developers.

COVESA VSS: DEFINING THE SDV DATA MODEL

Having a common and well-defined set of data types is crucial for a data-centric system — and this is where COVESA plays such a vital role. The COVESA Vehicle Signal Specification provides a community-created, independent reference of the data types and signals within a vehicle. Available today, the VSS covers more than 850 data signals in Body, Cabin, Chassis, ADAS, Powertrain, etc., and is continually expanding to cover more vehicle types, advancing autonomy, V2X, and more.

Part of this expansion work is with DDS, as described above. The COVESA VSS is exceptionally well-matched to DDS. Paired with this leading data-centric communications framework, COVESA VSS provides a stable and comprehensive data model for SDVs. This work represents one of the first collaborative efforts in applying DDS to specific automotive functionality in SDVs.

A growing ecosystem of solutions has formed around the COVESA VSS, enabling rapid prototyping, improved maintenance and enhanced user experience within the vehicle — all thanks to the data-centric interoperability of using common data types.

It's fair to say the race is on: vehicle architecture is rapidly advancing to previously unimaginable levels of complexity. As a result, COVESA is expanding the VSS to include these new capabilities, but there is much still to be done. As a community-driven standard, COVESA is actively supporting the expansion into these new realms of driver assistance, user experience, increasing levels of autonomy, V2X, privacy and security, and more.

CONCLUSION

The shift to software-defined vehicles is well underway, offering OEMs the advantage of unparalleled flexibility and an enhanced user experience. Yet the initial promise of rapid development has not materialized, in large part due to the complexity of integrating data among all sub-systems. By incorporating a data-centric architecture enabled by DDS, car manufacturers can accelerate time to market while still maintaining system security, flexibility and scalability. COVESA is leading the way through its work in providing automotive manufacturers with a standard set of interoperable data types, supported by tested, ready-to-use source code in enabling data-centric SDV functionality.

ABOUT COVESA

COVESA is a member-driven community that welcomes new ideas and use-cases. We invite you to help shape the future of SDV development, and to join with a group of automotive experts to help solve the most pressing challenges today. Join COVESA, or ask for an invitation to the weekly meetings of the working groups. Visit covesa.global to learn more.

ABOUT RTI AUTOMOTIVE

RTI is enabling the automotive transition towards a software-centric design through its collaboration in standards organizations such as AUTOSAR, AVCC®, COVESA, OMG® and SOAFEE, as well as with the leading automotive technology providers. This work enables interoperability across technology platforms for lower costs, productivity gains across global development teams and faster time-to-market. RTI Connext Drive® is an automotive-grade communication framework based on the DDS middleware standard that distributes vehicle dataflow in real time to ensure vehicle safety and performance. For more information, please visit www.rti.com.

ABOUT THE AUTHOR

Neil Puthuff is a Staff Application Engineer for Real-Time Innovations (RTI), the global leader in communication framework software based on the DDS data-centric standard. His primary focus is on vehicle autonomy and robotics. Neil's background includes software development technologies and low-power circuit design, which has resulted in more than a dozen U.S. patents and an EDN Innovation Award.

ABOUT RTI

Real-Time Innovations (RTI) is the infrastructure software company for smart-world systems. Across industries, RTI Connext® is the leading software framework for intelligent distributed systems. RTI runs a smarter world.

RTI is the market leader in products compliant with the Data Distribution Service (DDS™) standard. RTI is privately held and headquartered in Silicon Valley with regional offices in Colorado, Spain, and Singapore.

Your systems. Working as one.

CORPORATE HEADQUARTERS
232 E. Java Drive, Sunnyvale, CA 94089
Telephone: +1 (408) 990-7400
info@rti.com

rti.com
company/rti
rti_software
rti.com/blog
rtisoftware
rti.com/podcast