High-Performance Computing Platforms in the Automobile
Mastering OTA: Automotive and IT domains are converging

Over-the-air application cases such as software updates, live diagnostics and data collection promise enormous savings potentials for automotive OEMs and offer new opportunities for bolstering customer loyalty. However, the required effort is huge. One crucial factor in successfully handling the task is good integration into the existing process world.

The entry of high-performance computing platforms (HPCs) into vehicles is accompanied by transformation of the E/E architectures being used. Currently, multiple domain controllers are interconnected via a central gateway [Figure 1a]. The functions are already allocated to specific ECUs (microcontrollers) at production time. The software used there is primarily written in embedded C code which has been optimized for minimal use of resources. The programming paradigm does not permit dynamic memory management. One representative of this architecture is the AUTOSAR Classic platform [Figure 1]. For future vehicle generations, more complex functions are being implemented centrally in HPCs (microprocessors). These functional clusters communicate with one another via relevant middleware. Traditional ECUs are still used for the sensors and actuators [Figure 1b]. The goal is a service-oriented architecture which permits better maintainability and extendibility of the overall system – especially after the Start of Production (SOP). Many POSIX-compatible operating systems are used in these HPCs. Technologies and tools originating from the traditional IT world can be used for these platforms. It is possible to use object-oriented, high-level languages like C++, and a new programming paradigm enables dynamic memory management. One example of such an architecture is the AUTOSAR Adaptive Platform.

Software Update
In telecommunications, “over the air” software updates have been the usual practice for many years now. Today’s smartphones get an update to one or more applications nearly every day. However, these mechanisms are not used universally in the automotive field yet. Generally, the ECU software is still updated in a workshop. One reason is that an ECU update in a vehicle network has a higher level of complexity than an update of an app on a smartphone. Today’s vehicles can contain more than 100 ECUs which are based on different platforms and have different updating mechanisms. They communicate with one another in a heterogeneous network of multiple bus systems in a multi-layered topology. Furthermore, the vehicle must be in a safe
The specific language used to implement these sequences is of secondary importance. Relevant here are the flash-specific and diagnostic-specific library functions that are available to the application. The software update application of the automotive OTA solution from Vector, vConnect, uses flash containers which were created with the vFlash tool. Reuse of these flash containers is possible in the OTA context, because – along with flash sequences – they also contain all of the data, communication parameters and address information required for reprogramming. In particular, flash scripts are based on a diagnostic or flash library which is shared by the two applications. This means that the flash containers can be developed and validated locally with standard tools.

**Live Diagnostics**

To read out data or execute routines via diagnostics, it is often necessary to execute multiple diagnostic services in specific sequences. Examples are reading out the error memory or executing tests via diagnostics. The same diagnostic library used to describe these sequences in the Indigo diagnostic tool and in the software update case is also used in the live diagnostic application of vConnect. This library is used universally for the diagnostic and flashing application cases and is used both offboard and onboard. The complexity of these sequences continues to approach that of classic application software. It therefore makes sense to provide relevant tools to developers of such sequences. Specifically, these include a professional IDE and a test framework. This boosts efficiency and elevates the quality of the created sequences. For the runtime environment in the vehicle, on the other hand, aspects such as resource efficiency and safety are crucial [Figure 2]. An example of
collection query is valuable. These configurations must therefore be subjected to version management as well. This also applies to logger configurations which have been created and validated with classic tools. The signals transmitted over the vehicle buses already provide deep insight into a vehicle’s behavior. These signals can also be captured with minimal side effects. This assumes that the HPC system on which the application for collecting data is running is connected over Ethernet. Libraries which are generally available can be used to conveniently record Ethernet communication. This generates practically no additional load on the bus. For signals which are on CAN behind a DoIP gateway, a standardized bus mirror component has existed since AUTOSAR 4.4. It has an API (Application Programming Interface), which permits instrumentation at runtime. It enables packing selected CAN frames into DoIP packets and make these contained signals available over Ethernet ([Figure 3](#)). The data can then be filtered and aggregated or logically combined with triggers centrally in the data collection application of the HPC before transport to the backend. In the backend, the received data sequences might be converted to standardized MDF data format, for example. This format is supported by many of the measurement and analysis tools on the market.

### Framework

Security and data protection are fundamental requirements of an OTA solution. End-to-end validation of the communication is absolutely essential. Here, it makes sense to provide the communication and safety components that are needed for all OTA applications together in one framework. The automotive OTA solution vConnect offers such a security and communication API in its framework. This API makes complex communication patterns available, such as the layout of an encrypted session or request/response communication which are easy to use. Various security mechanisms have already been integrated in this API as well. It offers replay protection, for example, which allows the receiver to recognize and reject logged messages that

---

**Figure 2:** The complexity of live diagnostic sequences continues to approach that of classic application software. Developers need the right tools to boost efficiency and quality. One example is the Vector Diagnostic Scripting Library (VDS).

**Figure 3:** For signals which are on CAN behind a DoIP gateway, a standardized bus mirror component exists since AUTOSAR 4.4.

---
are being sent again. On the backend and in the vehicle, the framework provides standard mechanisms such as secure storage of certificates and keys or validation of signatures. The framework’s application programming interfaces are public. It can therefore also be used to implement new customer-specific OTA applications [Figure 4]. Due to its container-based backend architecture, the framework enables many different options for a deployment. On the vehicle side, multiple abstraction layers make it easy to adapt to different POSIX derivatives and hardware environments for communication, data storage and the human-machine interface, for instance.

**Figure 4:** End-to-end validation of the communication is absolutely essential for OTA. Here, it makes sense to provide the communication and safety components that are needed for all OTA applications together in one framework.

**Conclusion**
The advent of HPCs with POSIX systems in vehicles allows the use of tools and the adoption of programming languages and paradigms from classic IT. On this basis, OTA applications can be implemented more efficiently and with higher quality. Maximum reliability is achieved by developing portable software components that can be used both in the vehicle and in the offboard tools. Together with the use of identical input and output data, a common data creation process and further processing of the data in already introduced tools, a complete solution can be created that optimally covers both offboard and onboard requirements.

Translation of a German publication in Automobil Elektronik, issue 01-02/2020