optimize! softing

WHITE PAPER

UNLOCKING SERVICE NETWORK POTENTIAL

How to Configure and Maintain a Repair Shop Tester with Cloud Access and Seamless IT Backend Integration

On-Board Diagnostics (OBD) systems support the protection of our environment against harmful pollutants such as carbon monoxide (CO), nitrogen oxide (NOx), hydrocarbons (HC), and particulate matters (PM) emitted by combustion engines. OBD regulations require passenger cars and light-, medium-, and heavy-duty trucks to support a minimum set of diagnostic information to external (off-board) “generic” test equipment for diagnostic purposes.

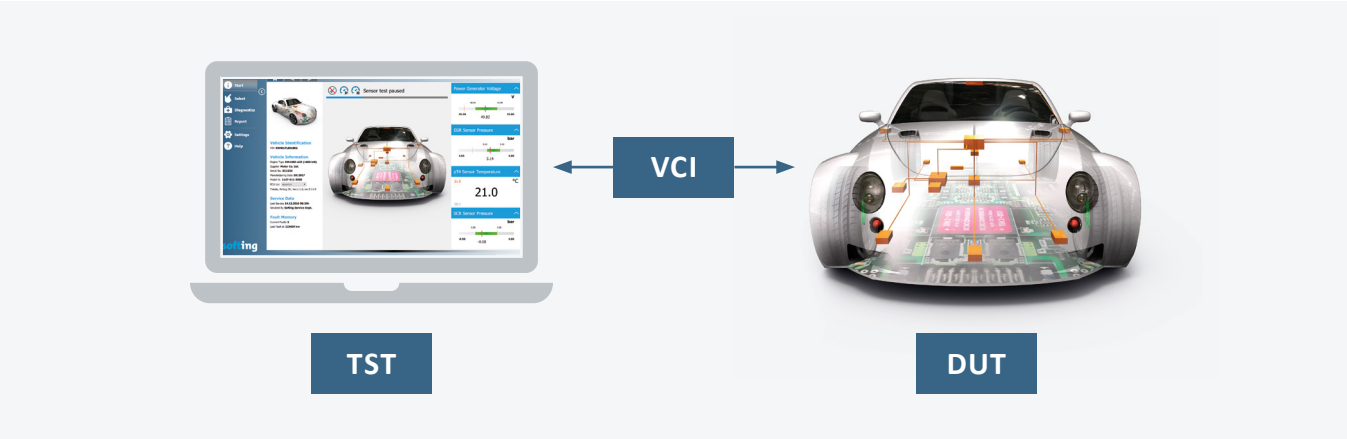
For the purpose of diagnostic communication, both the test equipment and the vehicle must support the same communication protocol stack. The communication protocol SAE J1979 aka ISO 15031 that has been in use for decades will be replaced by SAE J1979-2 for vehicles with combustion engines and by SAE J1979-3 for Zero Emission Vehicle (ZEV) Propulsion systems. Both SAE J1979-2 and -3 are based on ISO 14229-1 UDS.

This white paper explains the technology of a cloud-based diagnostic tester for vehicle service and repair shops, and the tools to create and maintain the components.

TABLE OF CONTENTS

1. INTRODUCTION TO DIAGNOSTIC COMMUNICATION

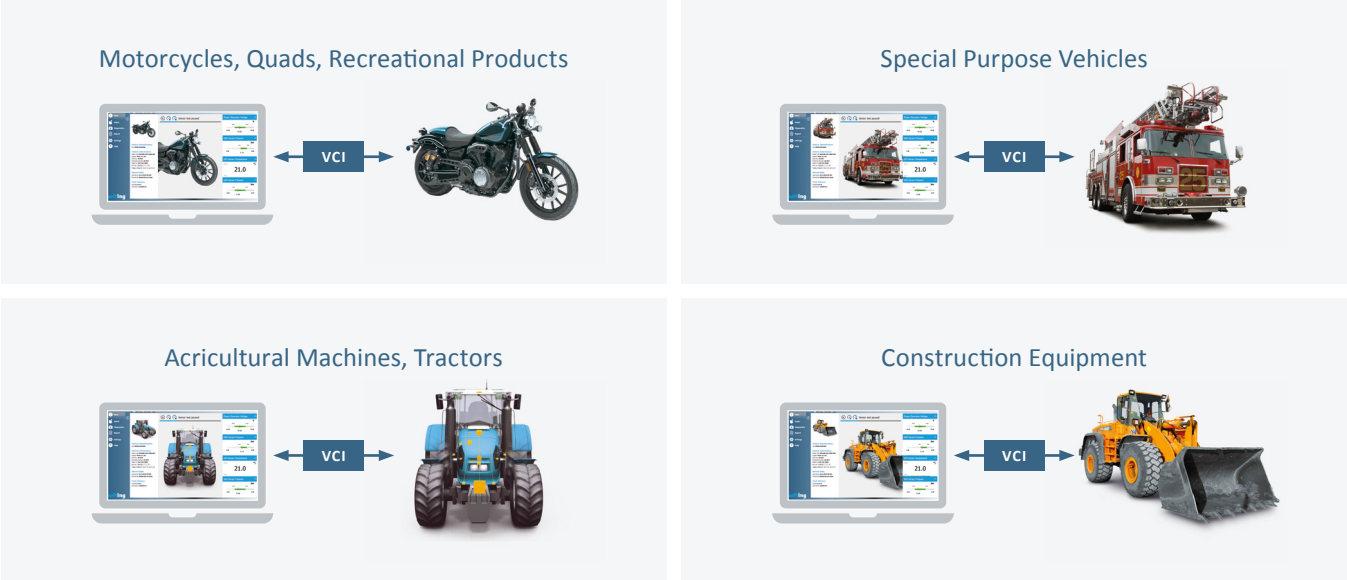
Figures 1 to **4** serve as an introduction to diagnostic communication and the most important terms and abbreviations.



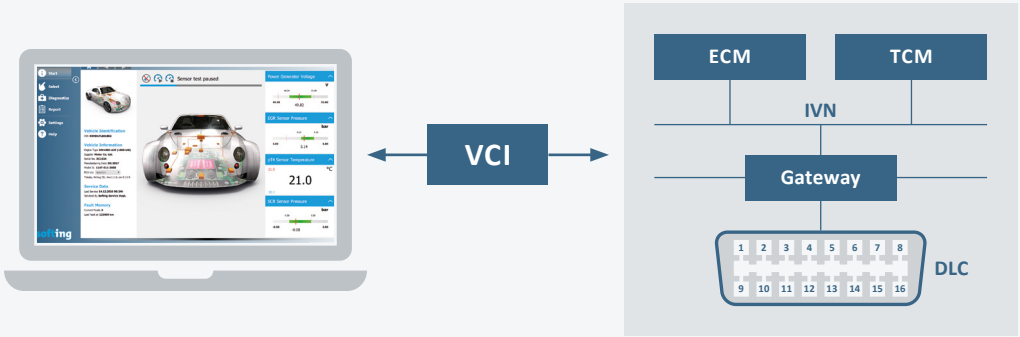
▲ **Figure 1:** Classical service and repair shop setup

Figure 1 shows a classical vehicle service and repair shop setup. A Vehicle Communication Interface (VCI) connects the tester (TST) with a passenger car (DUT = Device Under Test). For diagnostic purposes, the TST sends diagnostic service requests to the DUT and receives diagnostic service responses from the DUT. Requests and responses are specified in diagnostic communication protocols, for example SAE J1979 (OBD II), ISO 14229 (UDS) or SAE J1979-2 (OBD on UDS).

Figure 2 illustrates that the DUT can be any road-vehicle or non-road mobile machinery that supports diagnostic communication and comes with a plug to connect the VCI.

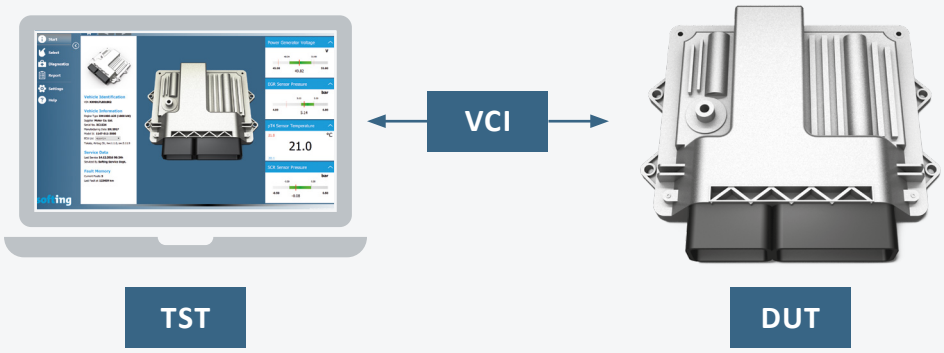


▲ **Figure 2:** Road-vehicles and non-road mobile machinery



▲ **Figure 3:** Diagnostic communication with an E/E system

Figures 3 and **4** illustrate that the TST is physically connected to a vehicle, but the communication takes place between the TST and the E/E-system of the vehicle (**Figure 3**). The E/E-system consists of Electronic Control Units (ECUs), such as the Engine Control Module (ECM), the Transmission Control Module (TCM) and Gateways connected to each other via an In-Vehicle Network (IVN), but also of sensors, actuators, wiring and connectors, such as the Diagnostic Link Connector (DLC). Examples of IVN include but are not limited to CAN, LIN, and Automotive Ethernet.



▲ **Figure 4:** In reality, the DUT is an Electronic Control Unit (ECU)

Independent of the complexity of the E/E-system, at a certain point in time, diagnostic communication takes place between the tester and one, selected ECU (**Figure 4**). In ISO standards and SAE Recommended Practices, the TST is referred to as the Client, and the ECU as the Server.

Standardized diagnostic communication protocols are based on the Open Systems Interconnection (OSI) Basic Reference Model[1] that structures communication systems into seven layers. **Figure 5** illustrates that Requests and Responses are specified on the Application Layer (L7). If an ECU is not able to support the requested service, it shall send a negative response, which is accompanied by a Negative Response Code (NRC) that includes information of the reason for not sending a positive response. The most common reason for a negative response is NRC 0x11 = Service Not Supported.

OSI Layer	Name	Content
7	Application Layer	Requests, pos./neg. Responses, SIDs, NRCs
6	Presentation Layer	DIDs, DTCs, FTBs, FMI, SPN
5	Session Layer	
4	Transport Layer	DoCAN, DoIP
3	Network Layer	
2	Data Link Layer	UTP, CAN, 100BASE-TX, Ethernet
1	Physical Layer	

SID Service Identifier
NRC Negative Response Code
DID Data Identifier
DTC Diagnostic Trouble Code

FTB Failure Type Byte
FMI Failure Mode Identifier
SPN Suspect Parameter Number
UTP Unshielded Twisted Pair (of copper wires)

DoCAN/ DoIP Diagnostics on CAN / Internet Protocol

▲ **Figure 5:** Open Systems Interconnection (OSI) Basic Reference Model

Each Request and Response is assigned a unique Service-Identifier (SID), such as the OBD request 0x09 = Request Vehicle Information or the UDS request 0x22 = Read Data By Identifier (**Figure 6**).

Since the introduction of On-Board Diagnostic (OBD II) Systems in California (1989), several standardized, mandated, and proprietary diagnostic communication protocols have come and gone. SAE J1979 aka ISO 15031 and its “OBD Modes” has been in use for decades, but due to regulatory adoption of OBDonUDS, SAE J1979 now becomes a multiple part document series. SAE J1979 will be replaced by SAE J1979-2 for vehicles with combustion engines and by SAE J1979-3 for Zero Emission Vehicle (ZEV) propulsion systems. For ZEV, emission-related failures will be replaced by ZEV propulsion related failures.

[1] ISO/IEC 7498-1

Both SAE J1979-2 and -3 are variants of the protocol stack UDS[2] with the specification of 26 Unified Diagnostic Services (**Figure 6**) but limited to OBD-related failures, meaning that these new diagnostic communication protocols are required by law but do not support advanced diagnostic functions, such as flash programming. For performance reasons of the flash process, the deployment of UDSonIP as it is standardized in ISO 14229-5 became state-of-the art.

The good news is that the application layer protocols SAE J1979-3 (ZEVonUDS) and ISO 14229-5 (UDSonIP) reference the same set of the lower OSI layer protocols all the way down from L6 to L1 (Ethernet).

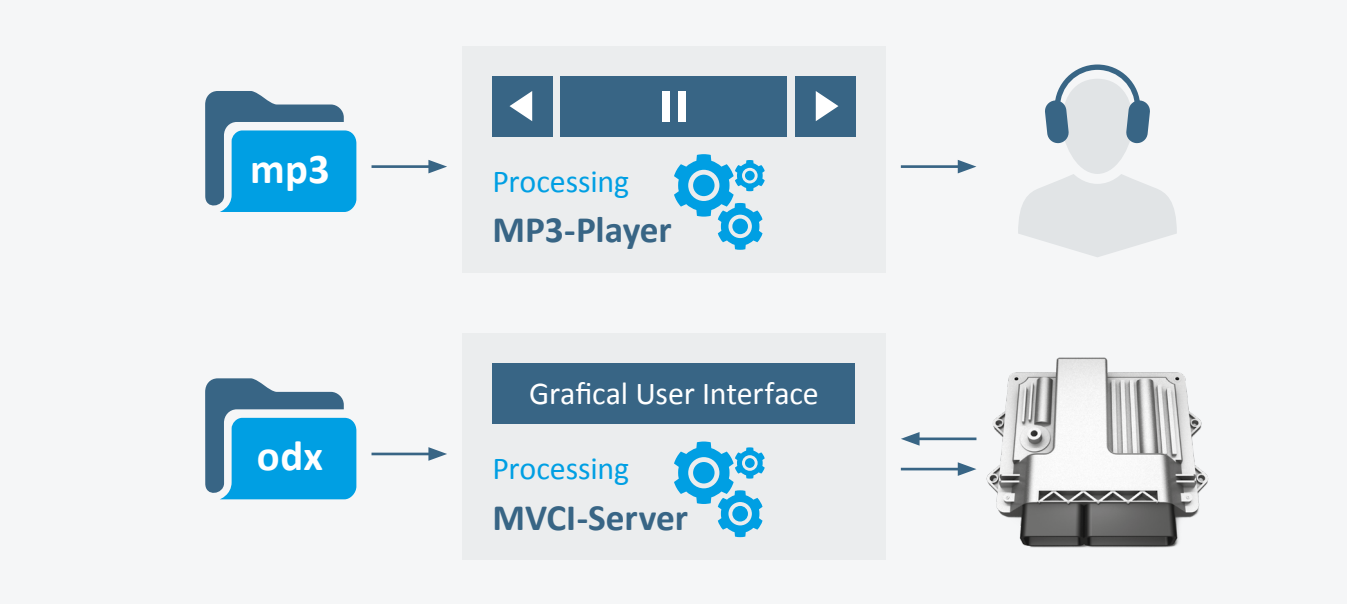
Request SID	Service	Request SID	Service
0x10	Diagnostic session control *	**0x31**	Routine control *
0x11	ECU reset *	**0x34**	Request download
0x14	Clear diagnostic information	**0x35**	Request upload
0x19	Read DTC information *	**0x36**	Transfer data
0x22	Read data by identifier	**0x37**	Request transfer exit
0x23	Read memory by address	**0x38**	Request file transfer
0x24	Read scaling data by identifier	**0x3E**	Tester present *
0x27	Security access *	**0x3D**	Write memory by address
0x28	Communication control *	**0x84**	Secured data transmission
0x29	Authentication	**0x85**	Control DTC setting *
0x2A	Read data by peridoc identifier	**0x86**	Response on Event *
0x2C	Dynamically define data identifier *	**0x87**	Link control *
0x2E	Write data by identifier	**0x2F**	Input/output control by identifier

* marks services that are parameterized by a sub-function byte

▲ **Figure 6:** ISO 14229-1 (UDS) = 26 Application Layer Services

[2] ISO 14229-5

2. EXTERNAL (OFF-BOARD) GENERIC TEST EQUIPMENT FOR DIAGNOSTIC PURPOSES



▲ **Figure 7:** Data-driven systems

Figure 7 shows two simplified examples of data-driven systems. MP3 is a data format. MP3-files contain compressed audio data and are processed by an MP3-Player, which converts binary data streams like 01000111000 back to Metallica´s “Nothing Else Maters”.

ODX[3] is a data format. ODX-files contain the machine-readable specification of diagnostic protocol stacks and diagnostic data. ODX-files are processed by an MVCI-Server[4] that converts binary data streams like 01110000 to meaningful information to help the technician to do his job right.

OTX[5] is used for the machine readable desription of diagnostic sequences.

The combination of the MVCI Server and the OTX Runtime can be understood as a generic data-driven diagnostic operating system. It is accompanied by custom-specific data and an application with a Grafical User Interface (GUI) that consists of widgets.

Qt 6 Creator by The Qt Company[6] lets you develop applications with intuitive user interfaces for multiple devices and platforms, such as Windows, MAC OS X, iOS, Android, and LINUX.

[3] ODX = Open Diagnostic Data Exchange Format (ISO 22901)
[4] MVCI = Modular Vehicle Communication Interface (ISO 22900)
[5] OTX = Open Test Sequence Exchange Format (ISO 13209)
[6] Qt GROUP (NASDAQ Helsinki, Finland: QTCOM) https://www.qt.io/group

3. TDX AT A GLANCE

Vehicle service and repair shop testers provide a highly specialized functionality for a heterogeneous group of users. Depending on their education, employees of service and repair shops can be anything between highly specialized technicians and unskilled workers. Especially the later need support and tools with user guidance.

Although legislators and standardization bodies use the term “external (off-board) generic test equipment for diagnostic purposes”, there is no generic diagnostic tester. Diagnostic test equipment differs depending on the vehicle manufacturer (Corporate Design requirements, IT organization, …), the vehicles (model, type, year, propulsion, type of transmission, accessories, …), branded dealers and workshops (number, international presence, staff education, …), and so on. On the other hand, there are several tester components and functions that, once created, can be improved, approved and then reused. Many components of external test equipment are based on international standards.

Examples for such tester components include but are not limited to

- Diagnostic operating system (ISO 22900 MVCI Server, OTX Runtime)
- Diagnostic communication protocol specification templates in ODX (ISO 22901)
- OTX sequence templates (ISO 13209)
- Language files (XLIFF)
- Graphical elements and UIs
- Web services
- …

Examples for reusable functions include but are not limited to

- Selection of the DUT
- Graphic display of the ECUs and the IVN
- DTC and freeze frame handling
- ECU (flash-)programming
- Graphic display of diagnostic data and measurement values
- …

There is no generic diagnostic tester, but there are several TST components that, once created, can be reused.

4. TDX CONTENT ENGINEERING PROCESS

A TDX-based diagnostic tester consists of a framework (TDX.workshop) with predefined functionalities and customer-specific components (TDX content). The TDX content is based on templates that can be customized with content development tools (Chapter 5).

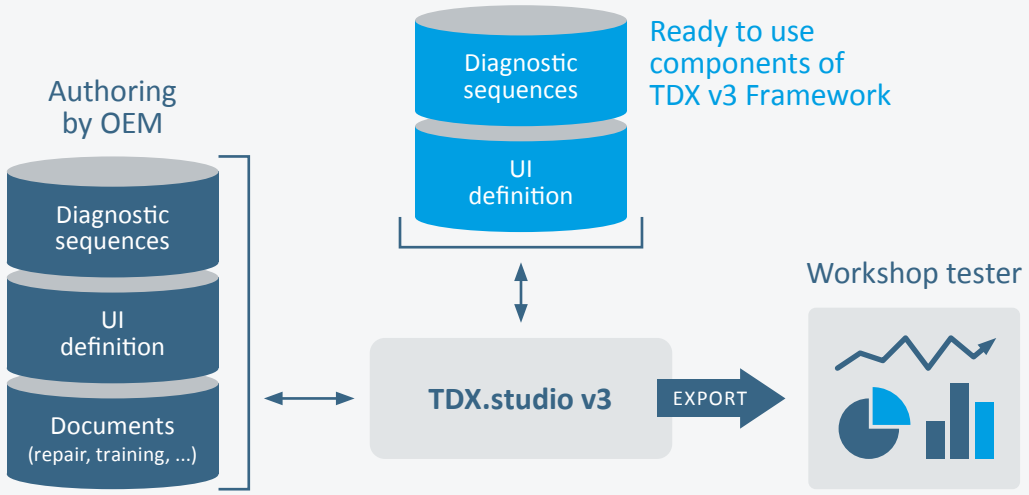
In the context of this paper, the engineering of TDX content consists of the creation/development of content from scratch, the editing and adaptation of templates, and the maintenance of already existing content. To start with the development of a diagnostic tester, the "TDX template" (**Figure 8**) is used to create a TDX project.

The TDX template comes with ready-to-use tester functions such as

- User login
- Navigation through the product range
- Vehicle overview
- Diagnostic function selection
- Update mechanism
- User feedback dialog (bug tracking)
- VCI selection and settings dialog

The TDX project is the container for the entire content of the workshop tester. The TDX project is created, developed, and maintained with TDX.studio. For the execution on the tester, the TDX project gets exported by TDX.studio in a special runtime format.

TDX project represents the part of the tester which can be customized by the OEM. The MVCI Server, OTX Runtime, D-PDU API and other parts of our binary delivery cannot be modified. The TDX project is that container which is uploaded to the TDX.server as a tester release. That release is what gets automatically or manually downloaded to the tester side.



▲ **Figure 8:** TDX project

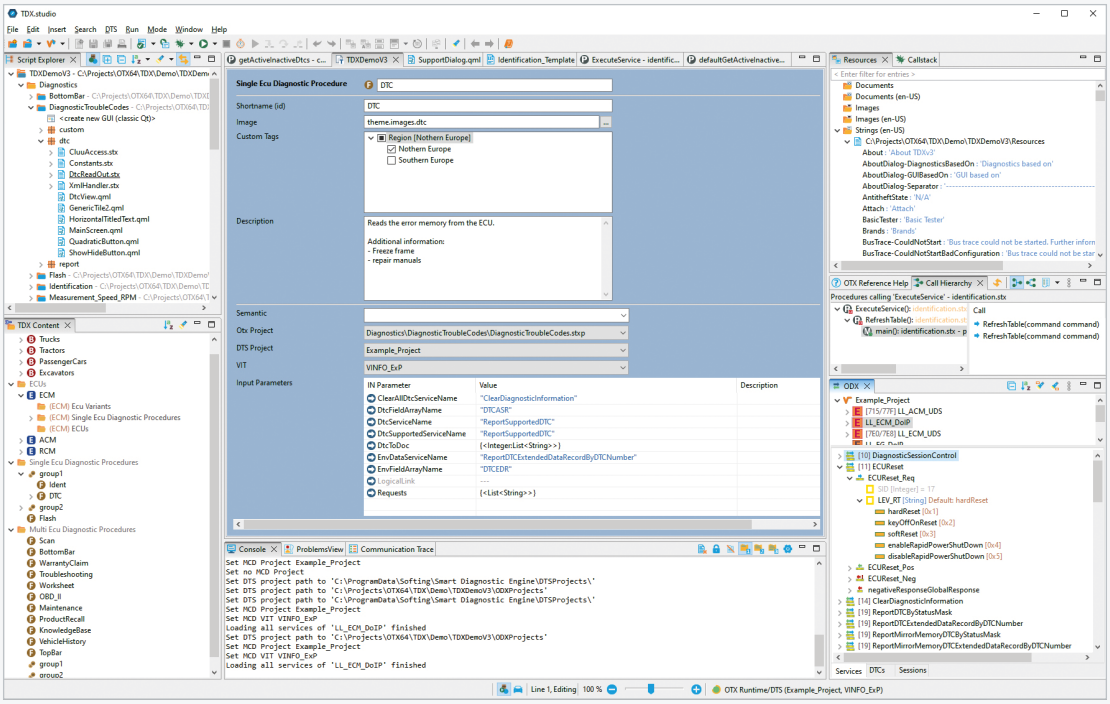
5. TDX CONTENT ENGINEERING TOOLS

5.1 TDX.STUDIO

TDX.studio (**Figure 9**) is the central tool to create a TDX project and its customer-specific content.

TDX.studio supports

- Developing OTX sequences for diagnostic functions
- Configuring of OTX templates
- Maintaining of foreign languages
- Modelling the product range of the vehicle manufacturer (brands, vehicle types, ECUs)
- Mapping diagnostic functions to the product range
- Tagging content for different user levels or license types
- Applying and customizing different themes for the tester
- Integrated development environment for OTX
- Debugging and deployment of diagnostic functionality for the tester



▲ **Figure 9:** TDX.studio for the engineering of TDX content

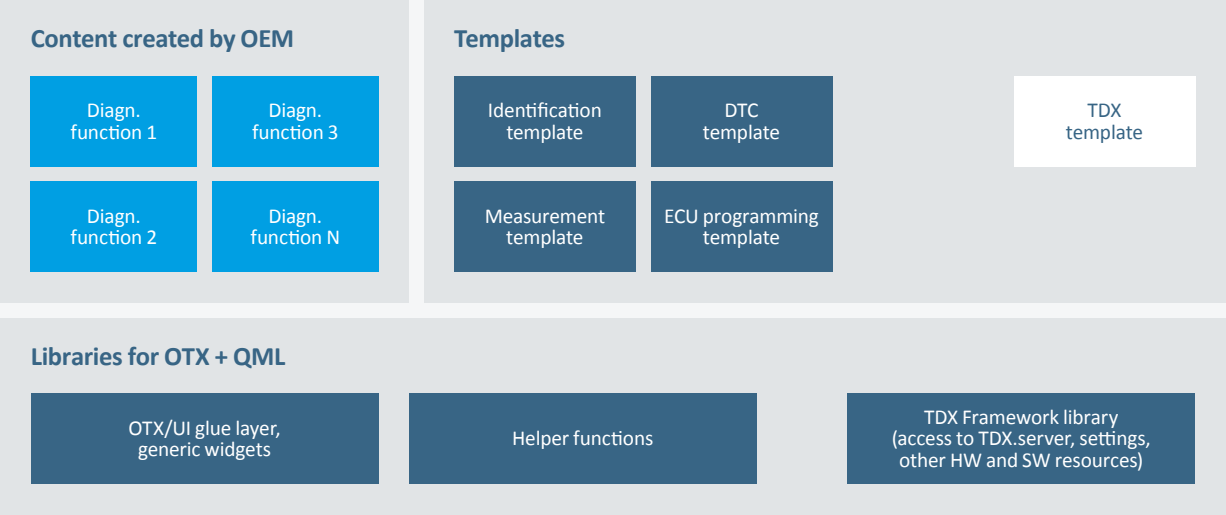
Executing the tester in debug mode

- Release export onto TDX.server of the TDX content
- Modeling the product range in terms of ECUs and vehicles

TDX.studio comes with ready-to-use OTX sequences, such as

- ECU identification
- Display of measurement values
- Error memory handling
- ECU programming

Libraries are another building block which helps the content engineer to perform his tasks faster by means of well tested code. Other advantages are a beter maintainability of the engineer's code as standard tasks are done the same way every time. OEM content, templates and libraries usually consist of OTX and QML.



▲ **Figure 10:** Libraries for OTX and QML development libraries

Libraries exists for a variety of use cases. The most important ones deal with generic UI widgets, access to different software and hardware resources and a glue layer between the UI and OTX.

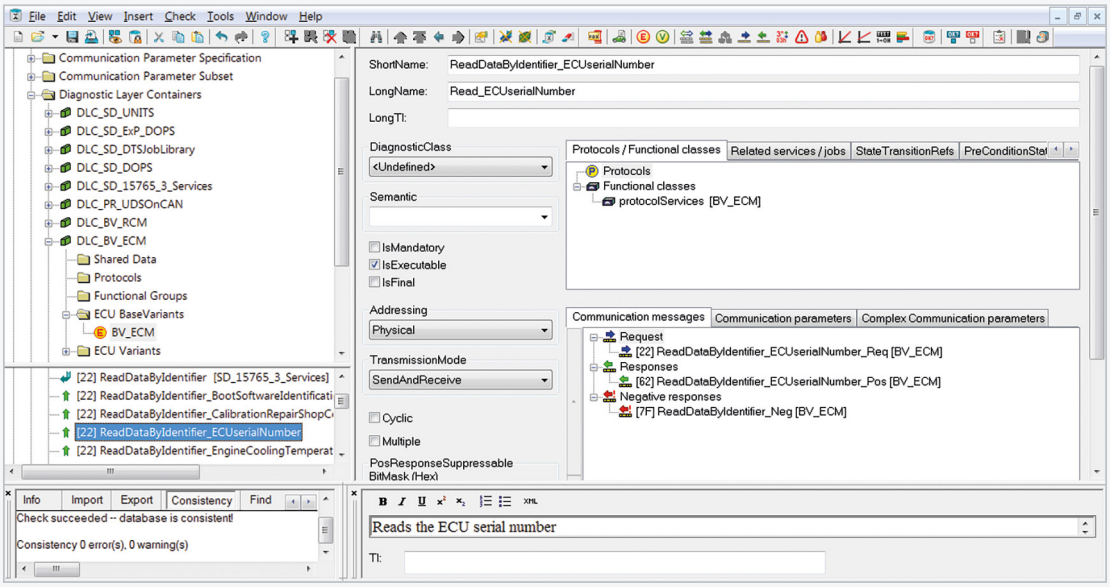
TDX.studio provides a syntax highlighting editor for QML engineering. This is helpful if small changes on the QML should be done quickly. Also, an experienced QML developer will appreciate this shortcut to QML files.

5.2 DTS.VENICE (ODX)

If you do not have the ODX files, they can be created with an ODX editor such as DTS.venice. We deliver ODX files with templates of diagnostic communication protocol stacks such as UDSonCAN, UDSonIP, OBD II, OBDonUDS, ZEVonUDS and (on demand) SAE J1939. The protocol stack templates are generic and must be adapted to the respective ECU. The required information is part of the ECU´s diagnostic specification.

ISO 22901 (ODX) and ISO 13209 (OTX) only specify the data formats, not the content. ODX data and the OTX sequences are “married to each other”, meaning that there must be a set of rules that facilitates the development of diagnostic sequences considerably due to the same service definitions across all ECUs and vehicles.

ODX-Authoring Guidelines (AGL) are part of the delievery.



▲ **Figure 11:** DTS.venice with UDS request 0x22 = read data by identifier > ECU serial number

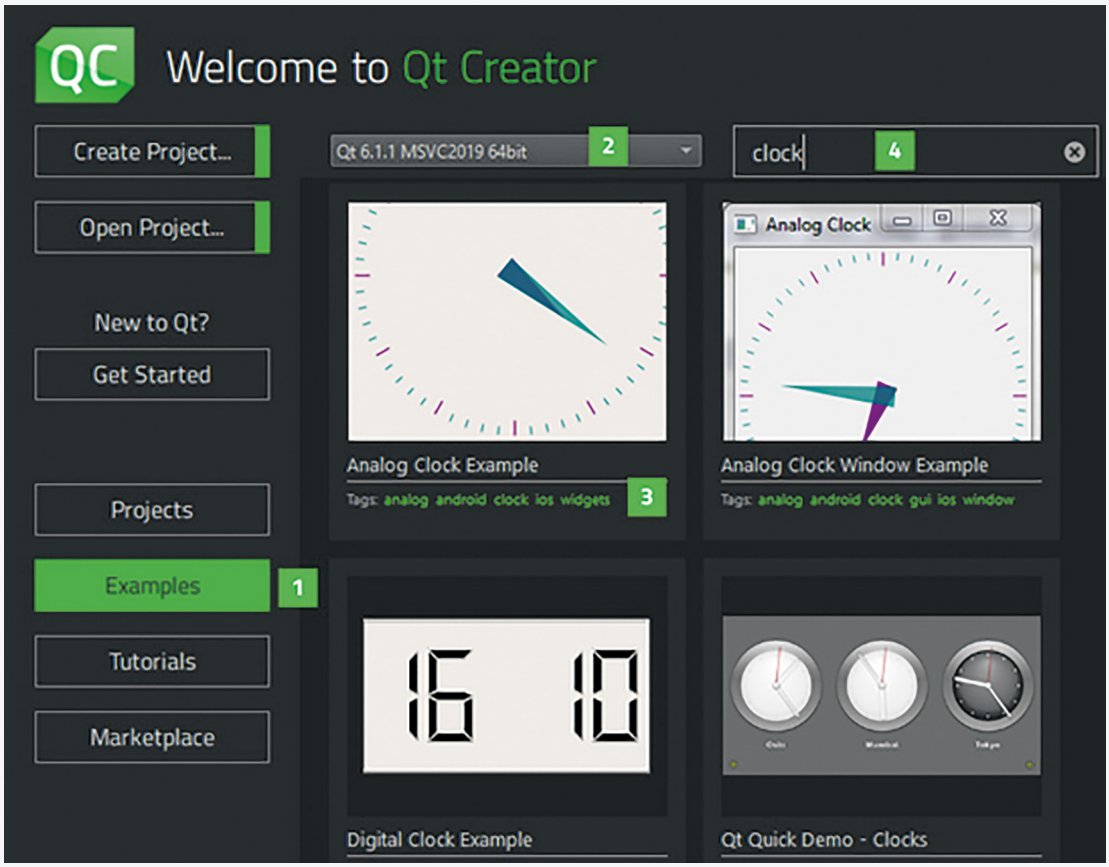
DTS.venice comes with an ODX consistency checker. Only if an ODX file passed the consistency check, it can be executed by the MVCI Server.

5.3 QT CREATOR

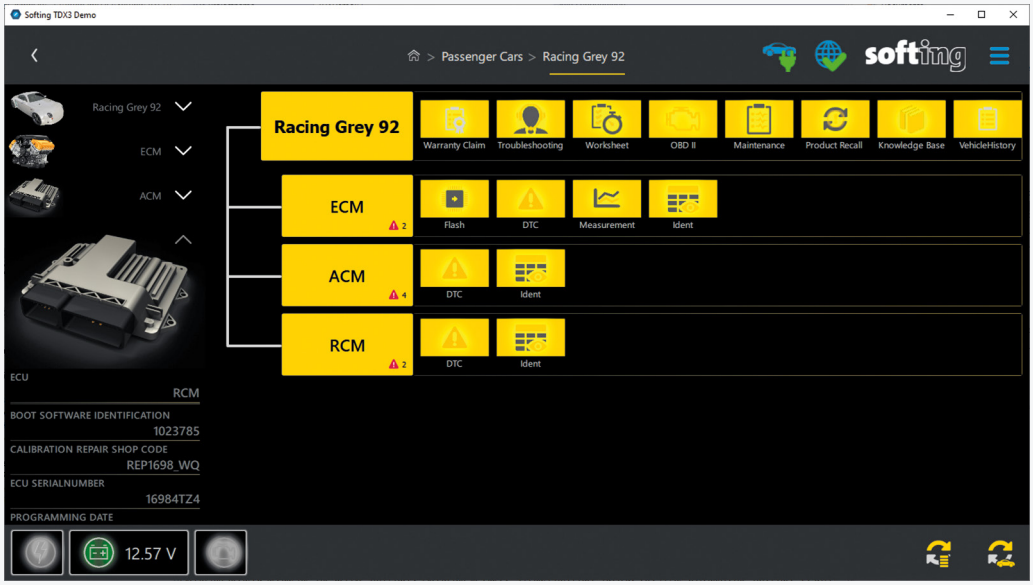
Qt Creator (**Figure 12**) is a cross-platform C++, JavaScript and QML integrated development environment (IDE) which simplifies GUI application development. QML (Qt Modelling Language) is like HTML, platform-independent and performant due to its usage of the GPU. It is part of the SDK for the Qt GUI application development framework and uses the Qt API. It includes a visual debugger and an integrated WYSIWYG GUI layout and forms designer. The editor has features such as syntax highlighting and autocompletion.

Qt Creator includes a code editor and integrates Qt Designer for designing and building graphical user interfaces (GUIs) from Qt widgets. The code editor in Qt Creator supports syntax highlighting for various languages. In addition to that, the code editor can parse code in C++ and QML languages and as a result code completion, context-sensitive help, semantic navigation are provided.

Qt Designer is a tool for designing and building graphical user interfaces (GUIs) from Qt widgets. It is possible to compose and customize the widgets or dialogs and test them using different styles and resolutions directly in the editor. Widgets and forms created with Qt Designer are integrated with programmed code, using the Qt signals and slots mechanism.



▲ **Figure 12:** Welcome screen of the Qt Creator (© The Qt Company Ltd.) https://www.qt.io/group



▲ **Figure 13:** Example of a QML-based GUI

6. TDX COMPONENTS

6.1 TDX.WORKSHOP

A TDX-based diagnostic tester consists of a framework (TDX.workshop aka TDX.ws) with predefined functionalities and customer-specific components (TDX content). Each TDX-based workshop tester requires a TDX.ws license.

6.2 TDX.SERVER

There is no doubt that data management is a must for every company, especially in aftersales use cases. Many companies have redundant data sources (data silos) that are separated from each other. As a result, data cannot be linked to each other and the right information that an employee needs to perform his or her task efficiently is very difficult to obtain or not available at all.

This leads to challenges such as

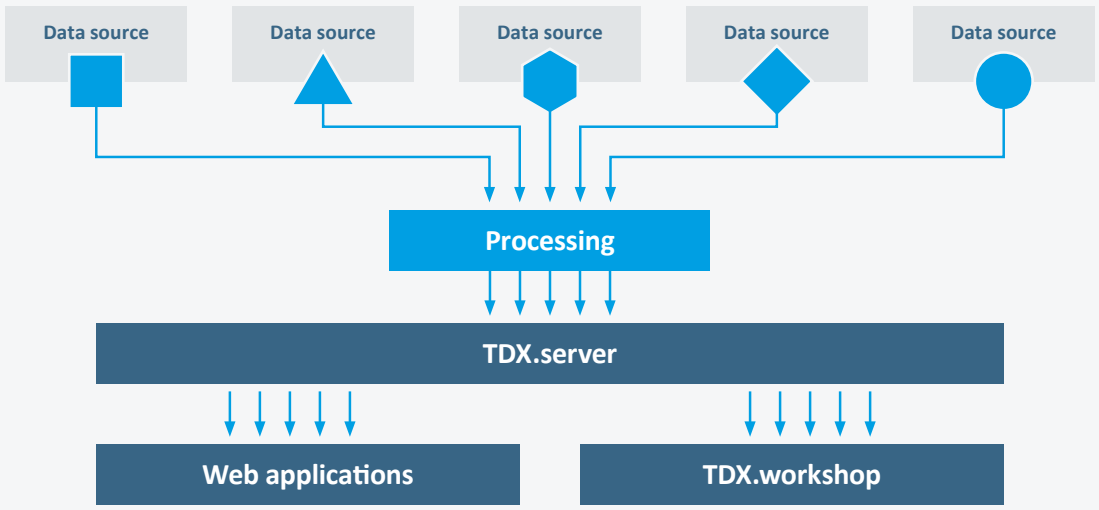
- Limited visibility into customer usage paterns and service requirements
- Inefficient service processes that result in high costs and long wait times
- Lack of insights into customer satisfaction and loyalty

These problems are solved by the use of a data platform. A data platform is a centralized system that collects, stores, processes and analyzes data from multiple sources. It provides a secure and scalable infrastructure for managing large volumes of data and allows organizations to gain insights and make informed decisions based on that data. A well-designed data platform can help improving operational efficiency, reduce costs, and drive innovation. For that purpose, data platforms are based on layered architecture to enable a continuous data flow.

The four basic layers are

- Ingestion
- Storage
- Processing
- Serving

The TDX.server is based on that architecture and is a data platform which collects, stores, and presents data from various sources (**Figure 14**).



▲ **Figure 14:** TDX.server connects different data sources

TDX.server comes with an SQL data base where all relevant data is being stored centrally as a “single point of truth”.

These data are (but not limited to)

- User credentials
- DTCs
- Documents
- Software content and releases
- Vehicle reports
- Customer feedback
- Licenses

For further processing, the data is being stored in an XML format.

Besides the SQL data base, TDX.server has immense capabilities in integrating and adapting to existing IT infrastructures, meaning that almost any system can be integrated as a data source by means of "connectors".

TDX.server can integrate user management systems e.g., Active Directory relate to existing user credentials. But also document management systems to handle additional content for the service tester can be made available.

There are three categories of connectors for the integration of existing systems and databases, as well as for new systems and databases to be added in the future:

- Java Database Connectivity (JDBC)
- Open Data Protocol (OData)
- Connector API

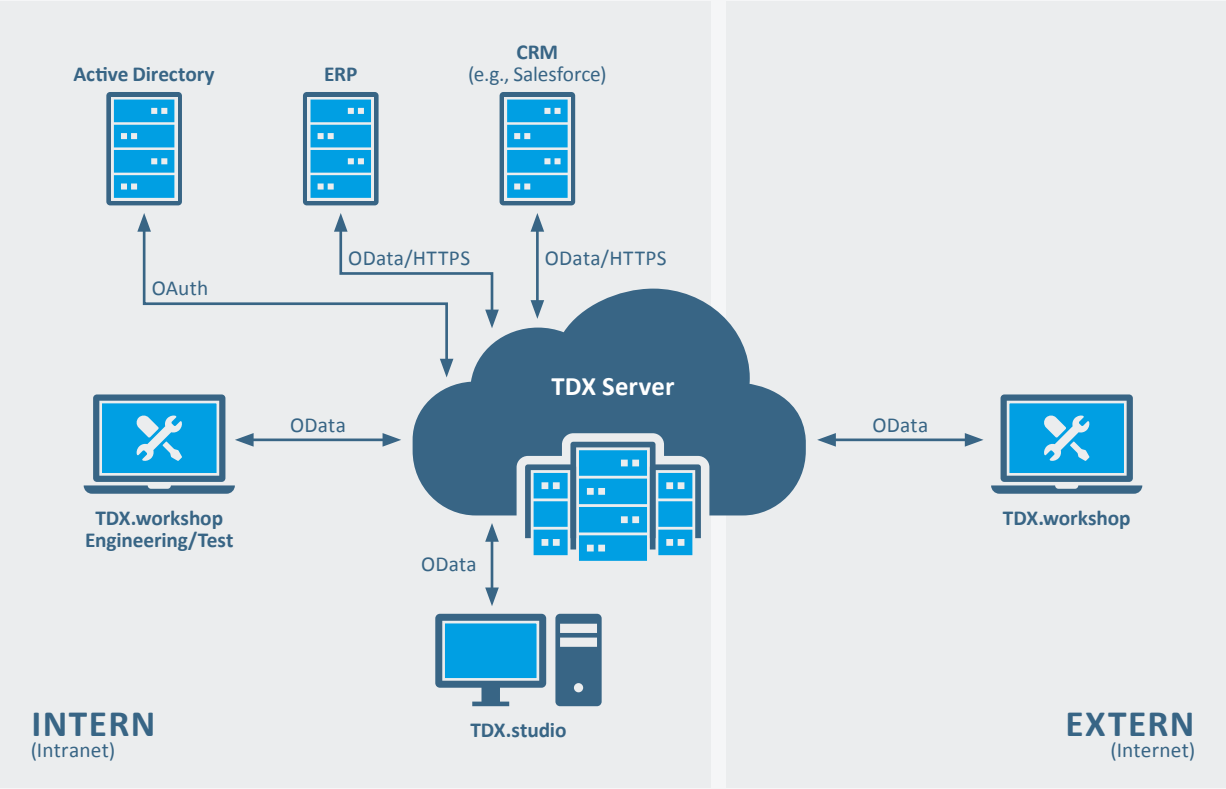
For CRM (e.g., Salesforce) or ERP systems, there are connectors that make the platform ready for immediate use. The JDBC Connector makes it possible to integrate almost any database. An interface is created that is independent of the database manufacturer and allows both reading of and writing to the database. Supported database types are MS-SQL, Postgres and Maria-DB, but also IBM DB2 and all other databases for which a JDBC driver exists.

With the Connector for OData, data can be integrated and processed quickly and easily via HTTPS/REST. The connector allows to access OData V2 and V4 compliant REST services. OData is the most widely used standard for this kind of communication and more comfortable to use than e.g., SOAP web services, because the protocol is based on known http requests.

oData is used to connect the TDX.server and the TDX.workshops. For authentication OAuth2 is being used.

In addition, even customized systems can be easily integrated using the Connector API: If a suitable connector is not available, it can be created very quickly and easily. Furthermore, TDX.server comes with a Web API to extract and load the data for further processing and analysis. The TDX.server Web API provides read and write access to the connected databases.

TDX.server not only serves as a backbone and IT integration layer. It provides all necessary functions and tools to setup und maintain aftersales environments and processes. Mainly, it serves as a central connection node to all diagnostic testers (TDX.workshop) located in the field, the integrated development environment (TDX.studio) and customer internal databases and management systems (**Figure 15**).



▲ **Figure 15:** TDX.server at a glance

The TDX.server runs on any cloud platform such as Microsoft Azure, Amazon Web Services (AWS) or the Google Cloud Platform (GCP). There are several deployment strategies customers can choose from.

ON-PREMISES

This involves deploying the data platform on your own hardware, either in your own data center or in a third-party data center. This gives you the full control over the hardware and software, but also requires you to manage and maintain the infrastructure yourself.

CLOUD DEPLOYMENT

This involves deploying the data platform in a public cloud environment such as Azure, AWS or GCP. This approach provides scalability, flexibility, and ease of management, but also requires careful consideration of security and compliance issues.

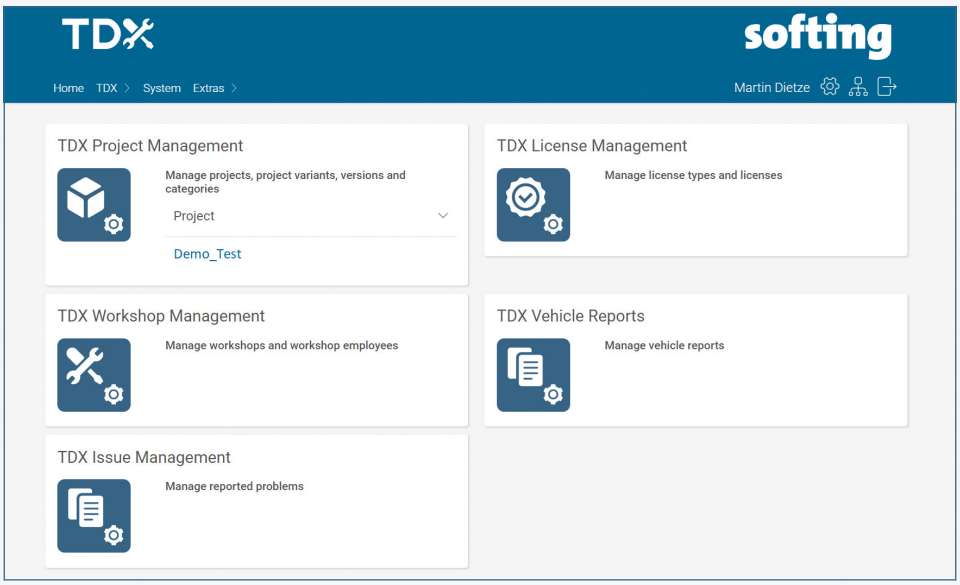
HYBRID DEPLOYMENT

This involves deploying the data platform across multiple environments, such as on-premises and cloud. This approach provides the best of both worlds, but also requires careful integration and management of the different environments.

TDX.server can be run as a Platform as a Service (PaaS) PaaS solution on platforms like Microsoft Azure or AWS, as well as installed on proprietary servers. This flexibility makes the solution highly scalable. New data sources can be connected at any time without having to redesign the platform.

Furthermore, TDX.server is equipped with multiple web applications (**Figure 16**) supporting users and administrators to implement and maintain aftersales use cases and processes, such as

- Project Management
- License Management
- Workshop Management
- Vehicle Reports
- Issue Management



▲ **Figure 16:** TDX web applications (apps)

TDX PROJECT MANAGEMENT

The project management app regulates the variants and versions of the diagnostic tester software (TDX.workshop). It helps administrators, release and product managers to keep track of the status of the entire software deployment process. In addition, individual releases can be linked to user-defined properties such as location, user authorizations, vehicle series and licenses. This enables user group specific software releases for example for beta testers as well as country or vehicle series specific versions of the tester. The coupling with a dedicated license provides additional security against unauthorized execution of the software.

Benefits

- Comprehensive overview on software deployment
- Easy release management
- User group specific software releases
- Additional protection against unauthorized execution of the software

TDX LICENSE MANAGEMENT

With the license management app licenses for individually defined categories such as region, brands or product lines or even license expiration time can be assigned to specific diagnostic tester software versions. Among others, this helps product managers to manage the tester functionalities. It offers a two-step approach starting with categories and breaking down to very specific tester functions. This makes it possible to configure almost any tester content in any granularity accessible only to authorized users and bind it to a license.

Benefits

- Comprehensive overview on software versions and licenses
- Easy editing and filtering of license categories
- Easy creation and assignment of license properties, specific to categories
- User-specific and license-based diagnostic tester functions

TDX WORKSHOP MANAGEMENT

The workshop management app helps to manage repair shops/dealerships, the assigned technicians and their permissions and licenses within the diagnostic tester. It helps OEM responsible to efficiently manage their dealership network and keep track of the maintenance and repair capabilities. At the same time, this strengthens cooperation with the individual repair shops and dealerships, as they can also gain access to the workshop management app. In addition, licenses (on- and offline usage) and authorizations for the workshops and their technicians can be issued here as well.

Benefits

- Comprehensive overview on repair shop and dealership network
- Easy management of entire network and single repair shops/dealerships
- Overview on worldwide available technician capacity
- Strengthens cooperation and information exchange with repair shops and dealerships
- Easy assignment of user-based on- and offline licenses and permissions

TDX VEHICLE REPORT

The vehicle-specific reports document the status of the vehicle and are stored in the vehicle database after each diagnostic session. In this way, the current status of a vehicle is recorded and can be viewed at any time. Among other things, this helps to recall repair measures that have already been carried out and to draw further conclusions. In addition, the central vehicle database can be used to create evaluations of multiple vehicles.

Benefits

- Document and keep track of vehicle conditions
- Elaborate on vehicle maintenance and repair measures
- Evaluations of vehicle fleet

TDX ISSUE MANAGEMENT

The TDX issue management simplifies collaboration with and among technicians. It helps to speed up repair procedures and other aspects of an aftersales tester environment. Entries and requests of any kind can be made directly from the diagnostic service tester TDX.workshop. For beter traceability, administration data such as creator, workshop, date etc. is atached to each entry. In addition, log files and screenshots of the tester are added directly when the entry is created. This makes the work of support teams much easier.

Benefits

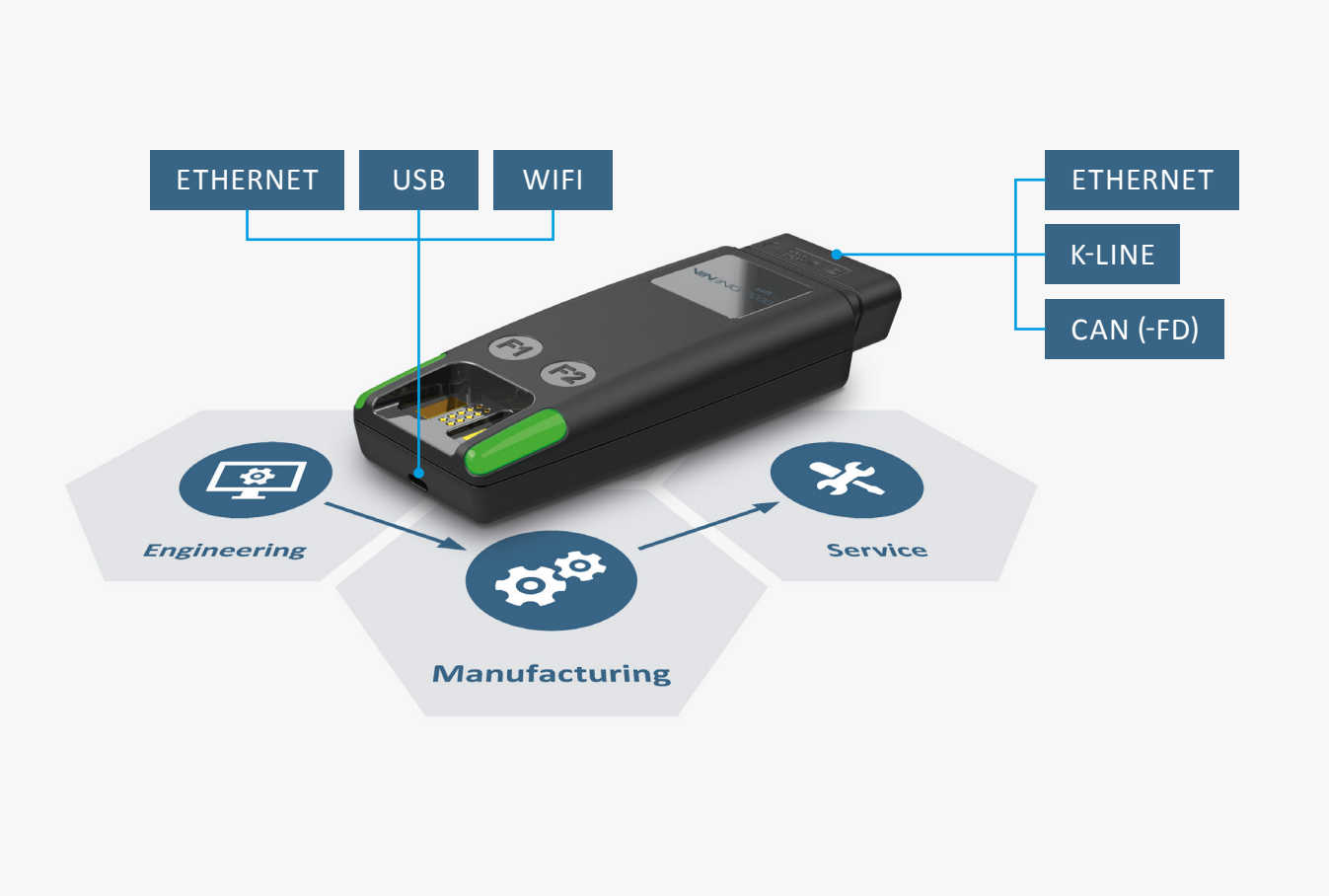
- Easy creation and processing of tickets
- Automated addition of log files and screenshots to a ticket

6.3 TDX VCIS

The standardized ISO 22900 MVCI Server includes the D-PDU API for the logical connection of VCIs. In addition, a selected range of SAE J2534 Pass-Thru VCIs can be connected without further ado. However, Softing recommends the use of tested and approved VCIs of the VIN|ING series.

The VIN|ING 800 (**Figure 17**) is a single-channel CAN interface that connects to the tester via USB. On the vehicle side, a CiA 9-pin D-SUB connector, a SAE J1962 (OBD II) connector or a SAE J1939-13 (HD OBD) connector is available.

▲ **Figure 17:** VIN|ING 800



▲ **Figure 18:** VIN|ING 2000

Figure 18 shows the high-performance VIN|ING 2000. It comes with an injection-molded SAE J1962 OBD II connector and supports K-Line, CAN / CAN FD and Ethernet on the vehicle side. The TDX tester is connected by Ethernet, USB or WiFi.

7. CYBER SECURITY MEASURES

7.1 INTRODUCTION

In addition to the requirements for the functionalities needed to fulfill the tasks in the diagnostic and development environment, security requirements are another focus of TDX. When securing the overall system, the security aspects relating to data communication are considered on the one hand, and the current security requirements relating to data storage and access control on the other.

All components and communication channels must meet security requirements. These are: The Cloud-based TDX.server with all its databases, the TDX.workshop clients that have access to the server, the VCI which realizes the connection to the vehicle itself and all the communication channels between these components.

7.2 SECURING THE SERVER

The Cloud-based TDX.server is the central component of the TDX system. It holds the data and is the central point of communication. Securing the data storage on the server and the communication channels is one aspect of securing the server. Another aspect is securing the Operating System (OS) of the server, for example Linux or Windows.

The Center for Internet Security (CIS) is an association of organizations which provides documents with 'best practices' concerning securing an OS. These so-called "benchmark documents" are available for many different OS and versions. Based on these benchmark documents it is possible to create scripts to check and if necessary, adjust the specific security settings of the particular OS and version. These scripts are applied during the installation process of the TDX system. Settings, which cannot be adjusted automatically, means via script, must be set manually during the setup process of the cloud infrastructure. Since these benchmark documents are updated regularly, TDX systems can be updated very fast.

7.3 SECURING DATA STORAGE

In TDX, data storage in various types of databases plays a central role. These databases can also hold data that is essential for the proper running of business processes. Therefore, it is very important to protect this data against unauthorized access, whether reading or writing. Both the data in the databases and the configuration data of the user software (TDX.studio, TDX.workshop) are stored in an encrypted way to protect them against atacks by unauthorized persons.

The ODX data is also stored in encrypted form to protect it from any unauthorized access.

The key used to encrypt the data is unique for each TDX customer, meaning that customer A has no access to the data of customer B, even if the data containers themselves – via whatever route – get from one customer to the other.

7.4 SECURING COMMUNICATION CHANNELS

TDX is a cloud-based system. This means that a significant part of the data traffic takes place via the Internet. Therefore, the focus regarding the security of the communication channels is on the internet-based data channels. To meet the security requirements at this point, the htps protocol is used. The HTTPS protocol represents the current state of the art in terms of secure communication via internet-based communication channels.

7.5 SECURING THE CLOUD

Very often, Microsoft Azure is used as a cloud solution for the TDX system. This allows Microsoft Azure Web Application Firewall (WAF) to be used in Microsoft Azure Application Gateway.

WAF provides centralized and extensive protection against common exploits and security risks. The features of WAF that are available in detail can be found in the Microsoft documentation. The configuration of WAF must be adapted per installation in each individual case.

This ensures maximum security of the cloud and the web applications.

7.6 SECURING THE VCI

The protection of the VCI against cyber-atacks cannot be described unambiguously at this point since it depends on the VCI used, which unauthorized accesses are detected and rejected or prevented.

If a VIN|ING 2000 is used, the safety aspects are already considered:

- Secure Transport Layer Security (TLS) connection for configuration: To prevent a potential atacker from gaining unauthorized access to the data traffic or the VCI itself in any form during the configuration process, the transmited data is encrypted during the configuration process. An analysis or modification of the transmited data is therefore no longer possible.
- Secure Shell (SSH) connection: A user or program may gain unallowed access to the VCI by using an SSH tool and cracking the password. For SSH public / private key approach you need to have a private key file on your computer to be able to access the VCI. Further a corresponding public key must exist on the VCI itself to be able to connect. SSH access via username and password is deactivated.
- Password protection for configuration: Unauthorized access is made more difficult by the fact that a password must be entered to access the VCI via the configuration software. The password is stored encrypted on the VCI and is configurable.

- User Management: If a potential atacker gains root user access to the VCI, he has full control over the VCI. The atacker can manipulate the system to its full extent. The use of the root user is prevented. The standard user has only limited possibilities and is not given full access to the VCI.
- Read only root partition: Users, scripts or programs can manipulate data in the root partition and thus render the VCI inoperable. Launch scripts can be manipulated to launch additional malware. If the root partition is read-only, a process or user cannot manipulate data stored on this partition of the device. Thus, important system data cannot be manipulated.
- Sign firmware update file: A user or program can tamper with the firmware update file or the files it contains. Signing and, if necessary, rejecting an unsigned firmware update file prevents manipulated firmware from reaching the VCI.

7.7 SECURING THE CLIENTS

The clients are the computers on which the TDX.workshop is executed, and the diagnostic tasks are performed. Securing the clients is a very special challenge since mostly the client computers are not in the responsibility of the OEM who offers the TDX infrastructure. The CIS benchmark documents mentioned in chapter 7.2, are also available for operating systems, which are installed on the client computers. Therefore, scripts can also be created for the client computer, with which the recommended security settings can be checked and, if necessary, adjusted in the same way as for the server. It is strongly recommended to apply comparable security measures to the client computers as they are applied to the server.

Softing Automotive | Richard-Reitzner-Allee 6, 85540 Haar – Germany
Phone +49-89-45656-420 | **Fax** +49-89-45656-499 | www.automotive.softing.com