ASAM MCD-2 D

ASAM MCD-2 D (aka ODX) allows the data-oriented specification of vehicle diagnostics. The standard defines a data model for the description of diagnostics capabilities of ECUs needed throughout the lifecycle of a vehicle from development, testing, production to after-sales and service. The standard facilitates the exchange of diagnostic information between partners in development process, e.g. between OEM and ECU supplier or between OEMs in a cooperation project. In detail, ODX covers the description of:

- Diagnostic communication via requests and responses, trouble codes (DTCs), parameters and other diagnostic data
- Communication parameters for different diagnostic protocols
- ECU memory programming
- ECU variant coding
- Function-oriented diagnostics

The standard defines a data model and description format, which is independent from specific vendors, busses or protocols, and which has well-defined semantics for all specification elements. ODX allows to store diagnostic data in a central location and to efficiently distribute the data to all involved parties from a single source. ODX data is serialized in a machine-readable XML format. Therefore, the standard enables the complete reuse of diagnostic data throughout all development phases of an ECU, e.g. for the design of diagnostic communication, the development of the ECU kernel and application software, configuration of diagnostic testers and the generation of the diagnostics documentation for the vehicle. Since the data originates from one source, ODX helps to prevent inconsistencies, errors and repetitive efforts.
ODX-compliant workshop testers or D-servers do not need to be re-programmed for every ECU or vehicle. The ODX-file is sufficient to configure such devices for handling diagnostic communication with an ECU or the vehicle network. The data in an ODX-file describes the communication between ECUs and external devices and allows the translation of the diagnostic data contained in the messages to human-readable values or text.
An ODX-compliant diagnostics stack of an ECU does not just allow to carry out basic diagnostics functions, but makes other functions available via the diagnostics system such as reading specific ECU-internal signals, reprogramming of the ECU, enabling or disabling of specific ECU features (variant coding) or identifying the currently flashed variant in the ECU. To prevent errors or misuse, ODX offers several means to ensure integrity, validity and authenticity of the diagnostic data access.

Datasheet

Title	Open Diagnostic Data Exchange
Category	AE
Current Version	2.2.0
Release Date	18.05.2008
Download	ASAM MCD-2 D V2.2.0
Application Areas	• Specification of diagnostic communication, data and capabilities of an ECU • ECU memory programming • ECU variant coding • Configuration of diagnostic devices, e.g. workshop testers or D-servers • Configuration of development tools

Specification Content	• Data model specification
	• Authoring guidelines
	• XML schemata
	• Communication parameters for:
	  ◦ KWP 2000 on CAN
	  ◦ KWP 2000 on K-Line
	  ◦ UDS on CAN

File Formats	odx, pdx

Content

- History
- Motivation
- Application Areas
- Technical Content
- Relation to Other Standards
- Industry Adoption
- List of Deliverables
- Further Reading

History

The diagnostic functions of early ECUs were usually just implemented in software and documented as part of the software documentation, typically as a control-flow-oriented description. In the beginning of the 90th, Daimler commissioned the development of a diagnostics runtime system, called CAESAR/Diogenes. The system required for the first time a data-oriented description of diagnostics functions and communication. This was the foundation of the development of the ODX standard (Open Data Diagnostics Exchange Format), which got later the official name ASAM MCD-2 D. After several years of development with multiple non-public versions, ODX version 1.2.2 was published by ASAM in December 2000 as the first public release.

This release contained the diagnostics communication data model and first attempts to describe ECU memory programming (*Flash*). Release 2.0 then mostly completed the data model for ECU memory programming. The data model got extended by vehicle identification information, network topology and the diagnostic plug pin-out (*VehicleInfo*). The latest release, ASAM MCD-2 D version 2.2 adopted the communication parameters from ISO 22900-2 (*Comparam*), included support for variant coding (*ECUConfig*) and functional diagnostics (*FunctionDictionary*).

Main contributors to the standard are Audi AG, BMW AG, Continental Automotive GmbH, Daim-ler AG, DSA Daten- und Systemtechnik GmbH, ETAS GmbH, General Motors Company, In2Soft - A KPIT Cummins Company, Porsche AG, Renault S. A., Robert Bosch GmbH, Siemens AG, Softing Automotive Electronics GmbH, SPX UK Ltd, Vector Informatik GmbH, Volkswagen AG.

Motivation

At the beginning of the computerization of cars, the development process for diagnostic functions and communication mostly consisted of writing code and documentation. Diagnostics got re-implemented multiple times in every new ECU project, in every test system, in every production system and in every workshop system. Besides the extra efforts of multiple implementations of the same diagnostic features, this caused inconsistencies, frequent communication errors and other costly time and quality issues. Furthermore, diagnostics specs were written in Word, PDF or proprietary ASCII formats, which were not machine-readable. Consequently, the relevant data had to be manually entered into tools of the ECU life-cycle. This waste effort multiplied, when distributed development groups within an OEM, or various suppliers and engineering providers of different companies are involved in the development life-cycle. This process of multiple implementation of the same functionality and multiple specification sources and formats for the same system was highly error-prone, expensive and impeded fast development cycles. Variants made this process even more complex. And pushing change requests through such a development process that is already in full swing quickly turned out as a mission impossible.

The main motivation for ASAM MCD-2 D was to consolidate all diagnostics specification data of an ECU or ECU network into one location, give it clear semantics that are understandable by all involved parties and make the specification machine-readable. This is achieved with the diagnostics data model of ASAM MCD-2 D and its serialization into the ODX data description file. The diagnostics system of a vehicle and its ECUs can be defined in one central location, exchanged via one standardized and machine-readable file and is the basis for all engineering activities at ECU development, production and service. Furthermore, ODX-specifications guarantee long-term availability of the data. It is very likely that computer systems are still able to import, process, understand and work with the data from ODX-files in the coming 10 to 20 years due to a stable and carefully advanced standard .

With ODX, the distribution of diagnostics specification is streamlined between different engineering centers of an OEM, its ECU suppliers and engineering service providers, manufacturing, dealerships, service franchises and diagnostic engineering tools makers. Since they are all based upon the same standard, the same information in the same data representation format can be distributed. This takes away the waste effort of converting specifications to different formats, manual implementation efforts of processes that could otherwise be automated, handling of multiple description formats for the same information and the necessary error resolution efforts that come along with those extra working steps.

ASAM MCD-2 D opened new markets for automotive suppliers and engineering providers. Diagnostics development turned from projects to products. Instead of writing software and developing tools for each OEM, suppliers could now develop products that can be used by all OEMs. This significantly reduced development and tooling costs for OEM and increased the profit margins for suppliers through scalable product sales.

Application Areas

The standard ODX can be employed throughout the complete vehicle-diagnostics life-cycle, which is development, manufacturing and service.

During development, the OEM works together with an ECU system supplier and potentially with other engineering service suppliers. OEMs typically have a cross-platform strategy for the diagnostics capabilities of their vehicle platforms, which represent an OEM-wide standard to be implemented in each vehicle. The goal of such a strategy is to have as little variation in the diagnostic capabilities of vehicles as possible, to promote reuse and to reduce the overall costs of diagnostics development. To meet this goal, OEMs use ODX during the authoring process of diagnostics data. It is common practice on the OEM side to set up an ODX-compliant toolchain with an diagnostics database at its core in each engineering center. This database contains the complete diagnostics data for ECUs and vehicles. Data is exchanged between the databases via the standardized ODX or PDX files. This allows to synchronize cross-vehicle platform ECU diagnostics development at different engineering centers and to ensure that company standards are followed. ODX/PDX-files are also used in a manufacturer-supplier relationship to exchange diagnostics specifications and to ensure that the supplied ECU software meets the specification.

ECU system suppliers and engineering service providers are typically at the receiving end of exchanged diagnostics data. They use ODX to extract requirements to be implemented in the diagnostic software and to configure development and test tools. For example, ODX is used for configuring the automatic generation of source code for diagnostic kernel software components, which automatically setup the protocol stack and diagnostic parameters. Test tools can be automatically configured with test cases to verify the diagnostic behavior of ECUs. The diagnostics documentation may be automatically generated based upon ODX data. Since this process can be highly automated via machine-readable ODX-files, there is little room for errors, misunderstandings or inconsistencies. The likelihood is high that suppliers deliver software that is compliant to the OEM's specification.

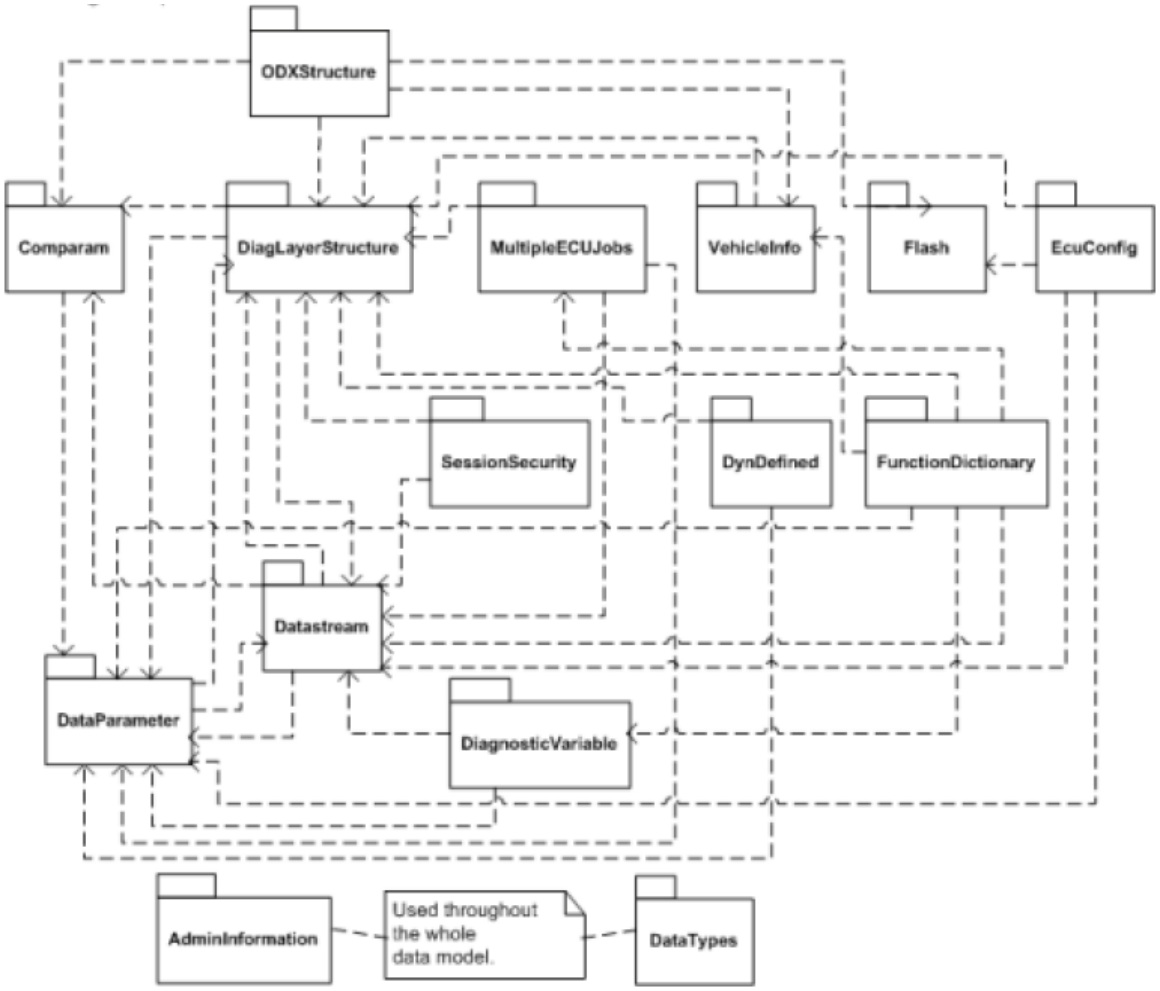
Once diagnostic development is near its end, the diagnostic data is released for use for manufacturing and service. Manufacturing uses the data to setup end-of-line test equipment. The service department generates the service documentation and the ODX runtime format for workshop testers. The latter is used by dealerships and service workshops to program existing diagnostic devices for access to the diagnostic system of the vehicle. One single file format is used for various diagnostic service systems.

ODX is most commonly used in the automotive industry, primarily by passenger car manufacturers, commercial truck makers, but also by construction- and off-road vehicle makers and in the shipbuilding industry. Some non-automotive use of ODX is known, such as for the development of heating systems for buildings.

Technical Content

Structure of the ODX Data Model

The foundation of ODX is a data model. The standard uses UML class diagrams to formally and unambiguously specify the data model. The data model is then mapped to XML, which is used for file-based transportation of ODX data between tools and databases. The standard also provides an XML-schema for validation of ODX files. Several ODX files and files in other formats can be packaged to a PDX-file (packaged ODX data) for transportation and archiving.



ODX package structure

The ODX data model is partitioned into four main areas, which describe the diagnostic capabilities of individual ECUs and a network of ECUs:

- Diagnostics Communication: Communication between an external device and an ECU or a group of ECUs in order to enquire diagnostic information and/or to modify the ECU's behavior for diagnostic purposes.
- ECU Memory Programming: Read or write access to the ECU's flash memory.
- ECU Configuration: Data records and their properties for ECU variant coding.
- Function Dictionary: Mapping of individual ECU diagnostic messages and data for the purpose to carry out diagnostics on a functional level.

The main areas of the ODX data model are furthermore subdivided into packages, which mostly reflect the logical structure of the data model and improves readability. The packages are described in tables in the following chapters.

Diagnostics Communication

The main part of the ODX standard contains the definition of diagnostics communication between one or several ECUs of a vehicle and an external device, such as a diagnostic tester or D-server. The standard covers several use-cases.

The most frequent functionality in the field of diagnostics is service-oriented communication. The external device sends a message with a service ID (SID) and related parameters to the ECU. The parameters might be a local ID (LID), common ID (CID), parameter ID (PID) or a memory address and number of bytes. The parameters define what is being requested, e.g. engine speed or a readout of a specific memory address section. The ECU responds with a message to the external device.

It contains a SID and typically repeats the requested parameters, which allows the external device to link responses to requests. The response message then contains more parameters, which contain the requested data, e.g. the current value of the engine speed or the content of the requested memory section. This kind of communication is specified via the *DataStrea m* package of the diagnostic communication data model.

A special use-case within the service-oriented communication is to read out diagnostic trouble codes (DTC) and related environmental data from the ECU's memory. This allows DTC-handling as defined in KWP 2000 (ISO 14230) or UDS (ISO 14229).

Closely related to service-oriented messages are dynamically defined messages. They allow to configure the requested data during runtime. Some protocols such as KWP 2000 (ISO 14230) and UDS (ISO 14229) allow dynamically defined massages. This kind of communication is specified via the *DynDefined* package of the diagnostic communication data model.

Besides service-oriented messages, ODX also allows to specify access to ECU data based upon variable identifiers (aka labels). This access method is similar to how calibration systems access measurement variables and calibration parameters in the ECU memory. This kind of communication is specified via the *DiagnosticVariable* package of the diagnostic communication data model.

ODX does not specify communication protocols, but the data model of ODX contains parameters of communication protocols. They are typically used to configure the protocol stack of external devices and define the behavior or real-time performance of a protocol. The communication parameters are specified via the *Comparam* package of the diagnostic communication data model.

ODX further supports dynamic diagnostic responses and variable length parameter. Diagnostic states, state transitions and authenticated access can be defined. The data model also contains a description of the network topology and vehicle identification information.

The packages of the data model for diagnostics communication are briefly described in the following table.

Package	Description

DiagLayerStructure	The DiagLayerStructure package allows the specification of a hierarchical diagnostic data model in several layers that allow different levels of company-internal standardization of diagnostics functions, communication and variants. The structure consist of four layers: • Protocol: Selection of the diagnostics protocol (e.g. KWP 2000 or UDS) with OEM-specific variants • Functional group: Diagnostics for functional groups such as powertrain, chassis, body, etc. • Base variant: Diagnostics for one type of ECU, e.g. engine controller or automatic transmission controller. • ECU variant: Variants of one type of ECU, e.g. engine controller for the US, engine controller for the EU, etc. Each lower layer inherits the data description from the higher layer and may override or eliminate descriptions. The hierarchy also includes a library of data shared among all layers. This structure allows to select a protocol and to systematically derive from this further specializations for the diagnostics communication and functionality. Hence, the *DiagLayerStructure* package facilitates internal standardization, reuse of specifications, definition of ECU variants and avoidance of data redundancy.
Comparam	The *Comparam* package contains parameters that specify the timing and logical behavior of the diagnostic communication. The parameters are protocol-specific. They are used to configure the protocol stacks of tools such as D-servers. ASAM MCD-2 D provides the communication parameter definitions for KWP 2000 on CAN, KWP 2000 on K-Line and UDS on CAN.
MultipleECUJobSpec	The *MultipleECUJobSpec* package allows the specification of services and jobs that are sent to multiple ECUs. This allows a tester device to open and close logical links in parallel to different ECUs and communicate simultaneously. This method of ODX is not used in practice. Instead, it is recommended to use ISO 13209 (OTX) to define sequences of services and jobs to be sent to multiple ECUs.
VehicleInfo	The *VehicleInfo* package allows the specification of data that describes the vehicle and access to its diagnostic network. In the first case, the data contains information about the make, model and type of the car, etc. In the second case, diagnostic access is described by the pin-out of the diagnostic connector, the logical link to gateway controllers or to individual ECUs and a description of the protocol with its characteristic communication parameters.

DataStream	The *DataStream* package allows the specification of the communication between the diagnostic tester and the ECU based upon services and jobs. The diagnostic tester requests a specific service via a message. The message contains a SID (Service Identifier) and dependent parameter values. The ECU responds with a message. The response contains a related SID and dependent parameter values. A job is used for more complex tasks and con-sist of one or more services. Messages in the DataStream package are defined on a bit-accurate level. Data requested from the ECU is packed in a message and in a coded format. The data can be interpreted and converted to a physical format via the DOP de-scription (data object properties) of the message's definition in the ODX data model.
DiagnosticVariable	The *DiagnosticVariable* package allows the specification of the communication between the diagnostic tester and the ECU based upon variable identifiers (aka labels). This communication is limited to read and write measurement variables and calibration parameters of the ECU. The diagnostic tester requests specific variables and/or parameters. They are bundled to one diagnostic variable, then put into a message and sent via a service request from the diagnostic tester. The ECU responds with a message and the requested data. This variable identifier-based access is an alternative access method to ECU data, similar to what is known from calibration systems. The method is internally mapped to the service-oriented access method as described in the package *DataStream*. This communication method is rarely supported by diagnostics tools and not very common in practice.
DataParameter	The *DataParameter* package allows the specification of simple and complex data objects in the diagnostic data stream. Simple data objects can be scalar values or diagnostic trouble codes (DTC). Complex data objects bundle simple data objects as structures, multiplexed data, environment data, fields or tables. Simple data is described via data object properties (DOP), which includes data types, computation methods (COMPU-METHOD, to convert values between a coded format and a physical format), units, limits, and other properties. The DataParameter package furthermore allows the specification of packaging of data into a protocol data unit (PDU) and extraction of data from a PDU.
DynDefined	The *DynDefined* package allows the specification of dynamically define data records at run time. Such records contain values from other existing records. This communication method is frequently used, when the configuration of a vehicle can change during or after production, as it is typical with US commercial trucks. Furthermore, dynamically defined data records ensure that the requested data is time-correlated, i.e. all data is produced at the same time and send via one message. It also reduces bus load, since the data is exchange via one instead of multiple request and response messages. This package contains information used by services to define and to read the dynamically defined data records. They occur in some protocols such as KWP 2000 (ISO 14230) and UDS (ISO 14229).

DataTypes	The *DataTypes* package contains the data types used to characterize UML data model elements. This package does not specify data types of parameters. See the definition of BASE-DATA-TYPE in the package *DataParameter* for this purpose.
SessionSecurity	The SessionSecurity package allows the specification of a state machine model, which handles two aspects of diagnostics communication: • preconditions for the execution of a diagnostic service or job • states and state transitions resulting from the execution of a diagnostic service or job This state model allows to specify diagnostic sessions, security states and correspondi ng methods to switch diagnostic sessions and to perform access authentication.
AdminInformation	The *AdminInformation* package allows to provide information used to support the diagnostics development process. Administrative information can be attached to each data model element. It may contains the name of the responsible person for the element, his company information, the revision history of the element, labels from a version management system and a special data group (SDG) for company-specific data that has otherwise no standardized location in the data model.

ECU Memory Programming

One use case of data transfer between an external device and an ECU is the programming or (partial) reprogramming of ECUs, called ECU memory programming. ODX allows to describe the information needed for the programming procedure. The standard is aligned to ISO 14229-1 and 14230, which define flash jobs in detail. A diagnostic application can use the ODX information to initiate such a flash job. The data model contains all needed information for this purpose.

The main description element of the ODX model for ECU memory programming is the ECU memory object (ECU-MEM). It contains the logical description of data blocks (DATABLOCK) with references to the actual data that is to be written into flash memory (FLASHDATA). The data model also may contain a description of the memory layout (PHYS-MEM), which enables the diagnostic tester to check, whether the data blocks actually fit into available memory. Further security means are available, e.g. identification labels and signatures that can be used to make sure that the data to be flashed is compatible with the data in the ECU.

The ECU memory object furthermore contain sessions (SESSION). They are the objects that can be chosen for ECU memory programming. Each session references one or more data blocks to be written to the ECU memory. Identification labels ensure that sessions are compatible to the target device. The session also holds information about how to perform checksum calculations to check the integrity of the flashed data. Sessions may also define an upload of data blocks from the ECU memory to the external diagnostic device. The data is then written to a file.

The data definition for ECU memory programming is contained in one package of the ODX data model, which is described in the following table.

Package	Description

Flash	The *Flash* package allows the specification of ECU memory programming, which enables an external device to write data into the ECU's flash-memory. The programming capabilities are organized in sessions, which contain a description of the data blocks to be programmed, the actual data to be flashed, and further information to ensure integrity, validity and authenticity of the flashed data. This package also allows to upload data blocks from the ECU to an external device.

ECU Configuration

One specific use-case of ECU memory programming is the configuration of an ECU, which is also commonly called "variant coding" in the automotive industry. Here, the diagnostics bus is used to set specific variants, e.g. to handle different car configurations or to configure the car for export to a specific country. ECUs are typically configured end-of-line during the production process of the car. The configuration may also be changed via a diagnostic tester after production in a workshop. ECU configurations are specified via the ECUConfig package of the ODX data model. The package contains variants in configuration records (CONFIG-RECORD), which are programmed into the ECU's flash memory.

The data definition for ECU configuration is contained in one package of the ODX data model, which is described in the following table.

Package	Description
ECUConfig	The *ECUConfig* package allows the specification of data to be written into the ECU's flash memory to parameterize different variants of functionality. A configuration record (CONFIG-RECORD) contains one or several data records (DATA-RECORD). Each data record represents one alternative content of the configuration. The data record is a contiguous block of binary data that is written to ECU memory. The meaning of the binary data and conversion to human-readable values can be described via configuration item objects (CONFIG-ITEM) and data object properties (DOP) on a bit-accurate level.

Besides flashing ECU configurations into memory, it is important that diagnostic testers are able to identify the variants that are implemented in an ECU. ODX allows to specify ECU variants (ECU-VARIANT), pattern and matching parameters, which identify specific ECU variants. A diagnostic tester can read out the ECU configuration via regular diagnostic commands and compare the returned data with the variant identification contained in ECU-VARIANTs. Once a match is found, the diagnostic tester reports that variant to the user. ECU variants and how to identify them is specified in the *DiagLayerStructure* package of the ODX data model.

Function Dictionary

The basic features of the ODX data model, - diagnostics communication, ECU memory programming and ECU configuration -, mostly focus on the diagnostic functions of a single ECU. However, the diagnostic functions in modern vehicles are increasingly distributed across multiple ECUs. The diagnostic requests have to be send to several ECUs and their responses may need to be consolidated into one result to be displayed in the diagnostic tester. This is called "function-oriented diagnostics" or "functional diagnostics", as opposed to "communication-based diagnostics" described in the earlier chapters.

The *FunctionDictionary* package of the ODX data model allows to specify functional diagnostics. Diagnostic functions are organized in the data model as hierarchical function node groups (FUNCTION-NODE-GROUP) and function nodes (FUNCTION-NODE). A function node represents a diagnostic function. This node contains references to existing diagnostic objects of ECUs, such as diagnostic messages (DIAG-COMM), data structures (TABLE-ROW), diagnostic trouble codes

(DTC) and associated environment data (ENV-DATA). A function node is essentially a mapping of one diagnostic function to individual messages and data pertaining to different ECUs.

A function dictionary does not specify new diagnostic capabilities of ECUs. It just bundles existing diagnostic messages and data of ECUs and makes them selectable in an diagnostic tester as one diagnostic function. This way the user of a diagnostic tester can carry out diagnostics on a functional level and does not have to understand or handle the diagnostic communication with each individual ECU that is involved in this function.

Package	Description
FunctionDictionary	The *FunctionDictionary* package allows the specification of function-oriented diagnostic routines that require communication to multiple ECUs. The package maps diagnostic messages (DIAG-COMM), data structures (TABLE-ROW), diagnostic trouble codes (DTC) and associated environment data (ENV-DATA) of individual ECUs to one diagnostic function. This enables diagnostic testers to carry out diagnostic requests on a functional level. Users do not have to deal with messages and data from individual ECUs and do not have to have knowledge of the underlying ECU network.

Relation to Other Standards

ASAM MCD-2 D defines together with the other standards ASAM MCD-3 D, ISO 22900-2 (D-PDU API) and ISO 22900-3 (D-Server API) an architecture for fully automated access to vehicle diagnostics in all phases of the ECU's life-cycle, i.e. during development, testing, production and service.

Furthermore, ODX is independent of particular vehicle diagnostic protocols. It is compatible with most diagnostic protocols in the automotive industry and specifically used with KWP 2000 (ISO 14230), UDS (ISO 14229), SAE J1939.

Industry Adoption

The first versions of ODX were just used as an exchange format between proprietary systems. OEMs initially kept their own description formats and diagnostics data bases. Gradually, German OEM adopted the ODX data model, ODX methods and related ISO standards for their ECUs. ASAM MCD-2 D is today fully accepted by all major German OEMs and Tier-1s and a standard component in the development of diagnostics software and systems. Other European OEMs followed gradually with lesser adoption rates. Asian OEMs just start to use ODX whereas American OEMs have not yet shown any significant use of the standard in their development or production processes.

OEMs employing ASAM MCD-2 D have reported considerably reduced development times of diagnostic functions, reduced setup times in the production of new models, highly reduced vehicle communication problems with diagnostic testers and much better support for vehicle variants.

List of Deliverables

The standard includes the following deliverables:

- Standard documents
- Schema files
-

Communication parameter specifications for KWP 2000 on CAN, KWP 2000 on K-Line and UDS on CAN
- Authoring guidelines for ODX files

Further Reading

- Datenkommunikation im Automobil: Grundlagen, Bussysteme, Protokolle und Anwendungen; Christoph Marscholik, Peter Subke; Hüthig-Verlag; 2007; ISBN: 3778529692
- Road Vehicles - Diagnostic Communication; Christoph Marscholik, Peter Subke; University Science Press; 2009; ISBN: 8131807347
- ASAM ODX; Stefan Bienk; IEEE/ICSE; 2008; ISBN: 9781605580791