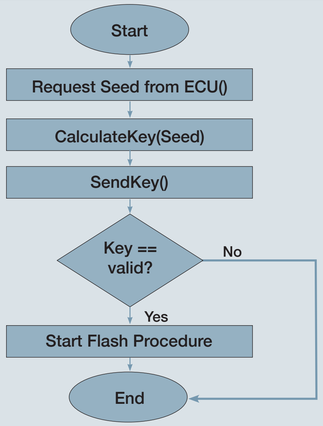
OTX Standardized Language for Diagnostic and Test Sequences

softing

Why OTX

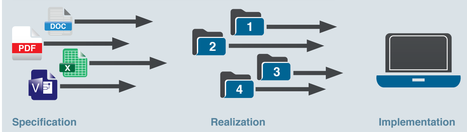
Typical Diagnostic Test Sequence



Problem / Situation

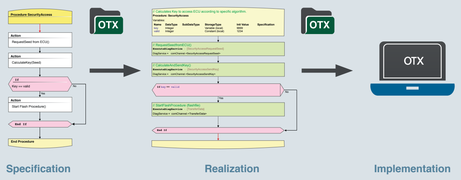
Process without OTX

Proprietary data - various data formats - media disruption



Process with OTX

Consistent workflow based on ISO standard

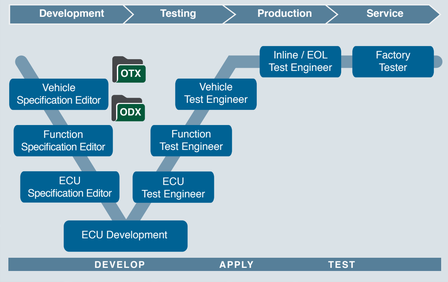


What is OTX?

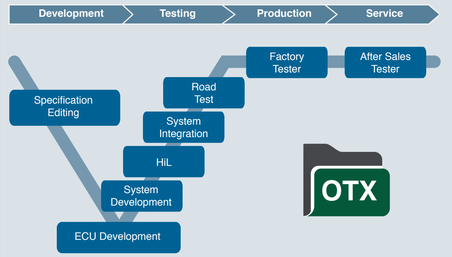
Standard format:
The OTX standard ISO 13209 proposes an open and standardized format for the human- and machine-readable description of diagnostic test sequences.

Diagnostic test sequences:
Sequences for diagnostic testing are utilized whenever automotive components or functions with diagnostic abilities are being diagnosed, tested, reprogrammed or initialized by off-board test equipment.

Who needs OTX?



Where is OTX applied?

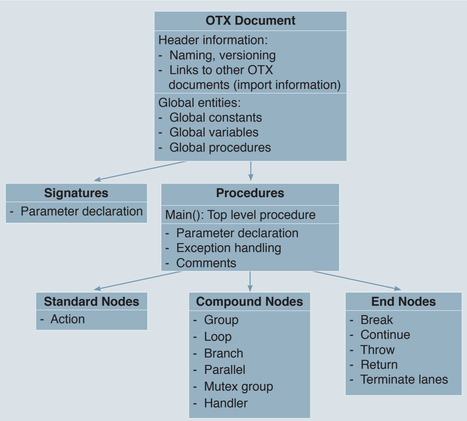


OTX Standard Overview

OTX Standard ISO 13209

OTX Overview	**Part 1** provides a general overview of the individual parts of the OTX standard. It also documents use cases that were considered during standardization.
OTX Core	**Part 2** lists the requirements and technical specifications for the basis of the OTX format, namely the "OTX Core". To achieve extensibility the core also contains well-defined extension points that allow a separate definition of additional OTX features – without the need to change the core data model.
OTX Extensions	**Part 3** extends the OTX Core with a set of additional standard libraries. Automotive-specific features are contained solely in Part 3 of ISO 13209. Furthermore, the list of extensions comprises controlling measurement equipment, internationalization, working with physical units, accessing the environment via external DLLs and APIs, human machine interface (HMI) elements and other utilities.

Overview OTX Core



Context Concept in OTX

Diagnostic test sequences require access to contextual information e.g. related to

- vehicle (model, engine type etc.)
- user (login name, access rights etc.)
- application (manufacturing, engineering etc.)

During runtime, context information is treated like static, unchangeable information.

Context information is declared as a context variable. Its value is set outside the OTX file. During program execution, it can be queried and evaluated.

```
ContextVariable_EngineType:String
ContextVariable_UserLogin:String
```

```
If ContextVariable_EngineType == "Diesel"
    ExecuteDiagService - {DiagnosticSessionControl}
```

Validity Concept in OTX

The *validity* concept is based on the *Context* concept.

A validity is always *true* or *false*.

It can be

- a context variable of type *Boolean*
- a composed logical expression resulting in *true* or *false* of e.g. several context variables.

Nodes in an OTX sequence can be bound to a validity and therefore contain several realizations.

```
Validity_UserLogin : ContextVariable_UserLogin == "Mueller"
Validity_EngineType : ContextVariable_EngineType == "Diesel"
```

Signature Concept in OTX

A signature describes an *interface to a procedure*. It equals a procedure without a realization.

Similar to nodes, procedures in an OTX sequence can be bound to a validity.

Instead of a procedure, the programmer calls the signature in an OTX sequence.

Depending on the validity, the corresponding procedure is executed.

In the OTX Core data model there is only one type of signature called *ProcedureSignature*.

```
My_Test_Procedure is executed when
- My_Signature is called
- and Validity_V1 is TRUE
```

Overview: OTX Extensions based on Part 3 of the ISO Standard

Extension	Description	Dependency
DateTime (otxIFD_DateTime.xsd)	Provides access to system time	Core
DiagCom (otxIFD_DateTime.xsd)	Connecting to ECUs, creating and executing diagnostic services, analyzing communication data	EventHandling, Quantities, Core
DiagDataBrowsing (otxIFD_DiagDataBrowsing.xsd)	Browsing functionality for reading data from the diagnostic database	DiagCom, Core
EventHandling (otxIFD_Event.xsd)	Support for the OTX event handling mechanism	Core
Flash (otxIFD_Flash.xsd)	Downloading and uploading flash data to and from ECUs	DiagCom, Core
HMI (otxIFD_HMI.xsd)	Exchange of information via the graphical user interface (GUI) through dialogs and screens	EventHandling, Core
i18n (otxIFD_i18n.xsd)	Internationalization features, multi-language support and translation mechanism	Quantities, Core
Job (otxIFD_Job.xsd)	Emulation of ODX Java Jobs by OTX test sequences	DiagCom, Quantities, Core
Logging (otxIFD_Logging.xsd)	Support for (Log4J-style) logging	Core
Math otxIFD_Math.xsd)	Extended mathematical functions	Core
Measure (otxIFD_Measure.xsd)	Executing measurement devices service, measuring physical values, analyzing measurements	EventHandling, Quantities, Core
Quantities (otxIFD_Quantities.xsd)	Handling of quantity data, with regard to SI unit system, transformations between units, etc.	Core
StringUtil (otxIFD_StringUtil.xsd)	Extended functionality for string handling	Core

OTX in Application

How do I use OTX?

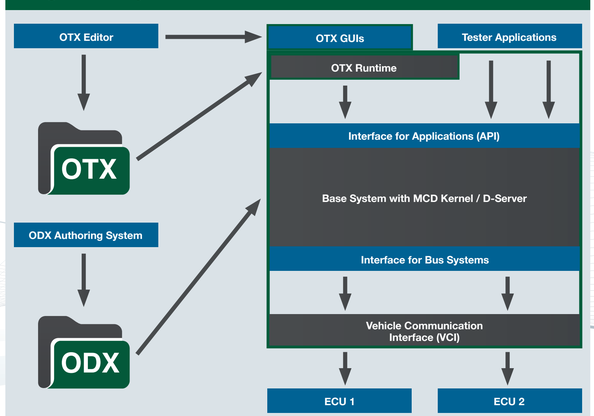
The OTX format offers explicit support for a three-stage development of test sequences.

Specification stage:
The overall sequence logic is developed at a descriptive, linguistic level.
This is helpful at early stages in the test sequence development process, when most details for creating an executable test sequence are not known.

Intermediate stage:
The test sequence contains parts that are already executable;
other parts are still at the specification stage.
At any step in this process, the sequence can be saved and exchanged in a valid format.
A runtime interpreter can execute the implemented parts.

Realization stage:
There are no more specification-only parts in the sequence.
The OTX sequence is now fully executable.
It represents an *"executable specification"*.

Diagnostic System Overview with OTX



Advantages of OTX

- The specification and implementation of test sequences are combined in one OTX document
- Easy maintainability of OTX sequences thanks to the separation of user interface, sequence model and data
- Simple integration of extensive OTX libraries for reusable sequences
- Debugging possibilities support error search when creating OTX sequences
- Human- and machine-readable filing format for test sequences (XML)
- OTX can also be used outside of vehicle diagnostics; other add-ons can be incorporated (invoking DLLs or APIs)
- Simpler in-house sequence creation as the focus is on application know-how, not on programming expertise
- The high degree of flexibility in the OTX creation process enables the use of even minor improvement potential which can nevertheless represent considerable cost savings
- The use of an ISO standard ensures a greater selection of available software tools (manufacturer independence)
- Long-term availability of validated diagnostic sequences and thus the securing of diagnostic know-how

Diagnostics - Use Cases for OTX

- ECU standard tests
- Protocol verification
- HiL - Hardware in the Loop testing
- Gateway testing
- Onboard diagnostics (OBD)
- ECU variant coding
- ECU flash programming
- ECU error memory
- Guided fault diagnosis
- ECU calibration
- Assembly-aiding production processes
- Test sequences for assembled systems
- EoL - End of Line tests
- Error analysis
- Measurement data monitoring
- Ensuring sequence order during production