

**ODX-LINK V1.5**  
**ODX-FLASH V1.5**  
User's Guide



## Copyright

---

The data in this document may not be altered or amended without special notification from ETAS GmbH. ETAS GmbH undertakes no further obligation in relation to this document. The software described in it can only be used if the customer is in possession of a general license agreement or single license. Using and copying is only allowed in concurrence with the specifications stipulated in the contract.

Under no circumstances may any part of this document be copied, reproduced, transmitted, stored in a retrieval system or translated into another language without the express written permission of ETAS GmbH.

© **Copyright 2005 - 2013** ETAS GmbH, Stuttgart

The names and designations used in this document are trademarks or brands belonging to the respective owners.

Document DD010101 V1.5.0 R01 EN - 03.2013

---

## Contents

<b>1</b>	Introduction	7
<b>1.1</b>	About this Manual	7
<b>1.1.1</b>	Target Group	8
<b>1.1.2</b>	Using This Manual	8
<b>1.1.3</b>	Labeling of Safety Instructions	9
<b>1.2</b>	Safety Instructions	10
<b>2</b>	Installation	11
<b>2.1</b>	System Requirements	11
<b>2.1.1</b>	Hardware	11
<b>2.1.2</b>	Software	11
<b>2.2</b>	Installation	11
<b>2.3</b>	Licensing of the Software	13
<b>2.3.1</b>	ETAS License Models	13
<b>2.3.2</b>	How to Get a License	14
<b>2.3.3</b>	The License File	15
<b>2.3.4</b>	Grace Mode	16
<b>2.3.5</b>	The "Expiration Warning" Window	17
<b>2.3.6</b>	Borrowing a License	18
<b>3</b>	Introduction to ODX-FLASH and ODX-LINK	21
<b>3.1</b>	Tasks of an ECU	21
<b>3.1.1</b>	Diagnostics	21
<b>3.1.2</b>	Programming	22
<b>3.2</b>	The ODX Standard	22
<b>3.3</b>	ODX-LINK V1.5	23
<b>3.4</b>	ODX-FLASH V1.5	25
<b>3.5</b>	Working with ODX Projects	25
<b>3.5.1</b>	Procedure	26

<b>3.5.2</b>	Diagnostics without an A2L File . . . . .	27
<b>3.5.3</b>	Automatic Search for and Configuration of OBDonCAN Devices . . . . .	28
<b>4</b>	Working with ODX-FLASH . . . . .	29
<b>4.1</b>	Creating an ODX-FLASH Job . . . . .	31
<b>4.2</b>	Creating an ODX-FLASH Project with an Authoring Tool . . . . .	34
<b>4.3</b>	Working with ODX-FLASH in INCA . . . . .	35
<b>4.3.1</b>	Database Manager . . . . .	35
<b>4.3.2</b>	Hardware Configuration Editor . . . . .	35
<b>4.3.3</b>	Memory Page Management . . . . .	36
<b>4.3.4</b>	ODX-FLASH . . . . .	36
<b>5</b>	ODX-FLASH Tutorial . . . . .	39
<b>5.1</b>	Lesson 1: Creating an INCA Workspace and Configuring the Devices . . . . .	40
<b>5.2</b>	Lesson 2: Defining ODX Parameters . . . . .	45
<b>5.3</b>	Lesson 3: Making Settings for the Flash Job in Memory Page Management . . . . .	48
<b>5.4</b>	Lesson 4: Executing the Flash Job from an Experiment . . . . .	51
<b>5.5</b>	Lesson 5: Flashing with External Data . . . . .	54
<b>5.6</b>	Summary . . . . .	56
<b>6</b>	ODX-FLASH Troubleshooting . . . . .	57
<b>6.1</b>	General Errors . . . . .	57
<b>6.2</b>	Errors when Executing the Flash Job . . . . .	58
<b>6.3</b>	Errors When Adding an ODX Project to the Database . . . . .	62
<b>7</b>	ODX Link Menus and Functions . . . . .	67
<b>7.1</b>	User Views . . . . .	68
<b>7.1.1</b>	Diagnostic Services . . . . .	72
<b>7.1.2</b>	ECU Identification . . . . .	76
<b>7.1.3</b>	Hex Service . . . . .	79
<b>7.1.4</b>	Diagnostic Trouble Code . . . . .	80
<b>7.1.5</b>	Memory Dump . . . . .	87
<b>7.1.6</b>	Sequence . . . . .	96
<b>7.1.7</b>	OBD . . . . .	101
<b>7.2</b>	Data Logging Configuration . . . . .	117
<b>7.3</b>	Snapshots . . . . .	122
<b>7.4</b>	Diagnostic Signals in the INCA Variable Selection . . . . .	125
<b>7.4.1</b>	Definitions in the Variable Selection Dialog Box . . . . .	125
<b>7.4.2</b>	Adding Diagnostic Signals . . . . .	126
<b>7.4.3</b>	Storage Location of the DSL File . . . . .	132
<b>8</b>	ODX-LINK Tutorial . . . . .	133
<b>8.1</b>	Creating an INCA Workspace . . . . .	134
<b>8.2</b>	Preparing an INCA Experiment for ODX without Hardware Connection . . . . .	139
<b>8.3</b>	Working with ODX User Views . . . . .	142
<b>8.4</b>	Preparing an INCA Experiment for ODX with Real Hardware . . . . .	146
<b>8.5</b>	Configuration of ODX User Views . . . . .	150
<b>8.6</b>	Using the OBD Protocol with ODX-LINK . . . . .	154
<b>8.7</b>	Working with the "OBD" User View . . . . .	159
<b>8.8</b>	Using Diagnostic Signal in the Experiment . . . . .	163
<b>8.9</b>	Measuring OBD Data on the Vehicle . . . . .	166

<b>9</b>	ODX-LINK Troubleshooting . . . . .	175
<b>9.1</b>	Errors When Adding an ODX Project to the Database . . . . .	175
<b>9.2</b>	Error when opening the ODX Project . . . . .	180
<b>9.3</b>	Error while starting a measurement . . . . .	182
<b>9.4</b>	Error during the measurement . . . . .	183
<b>10</b>	ODX Communication Parameters . . . . .	185
<b>10.1</b>	A2L Structure: TP_BLOP . . . . .	186
<b>10.2</b>	A2L Structure: K-Line . . . . .	187
<b>10.3</b>	A2L Structure: CAN . . . . .	188
<b>10.4</b>	A2L Structure: CAN Address . . . . .	189
<b>10.5</b>	A2L Structure: CAN TesterPresentOptions . . . . .	190
<b>10.6</b>	A2L Structure: SESSION TesterPresentOptions . . . . .	191
<b>10.7</b>	A2L Structure: CAN_NETWORK_LIMITS . . . . .	192
<b>10.8</b>	A2L Structure: DIAG_BAUD . . . . .	193
<b>10.9</b>	A2L Structure: TIME_DEF KWP_TIMING . . . . .	194
<b>10.10</b>	A2L Structure: TIME_DEF USDTP_TIMING . . . . .	195
<b>10.11</b>	A2L Structure: USDTP_TIMING_DEFAULTS . . . . .	196
<b>10.12</b>	A2L Structure: SESSION . . . . .	197
<b>10.13</b>	A2L Structure: ADDRESS_AND_LENGTH_FORMAT_IDENTIFIER . . . . .	198
<b>10.14</b>	A2L Structure: SESSION SessionOpeningOrder . . . . .	199
<b>10.15</b>	A2L Structure: CAN Transport Protocol Version . . . . .	199
<b>10.16</b>	Parameters that can be changed by the flash job . . . . .	200
<b>11</b>	Glossary . . . . .	201
<b>12</b>	ETAS Contact Addresses . . . . .	203
	Figures . . . . .	205
	Index . . . . .	207



# 1 Introduction

---

This manual contains a description of the INCA add-ons ODX-FLASH V1.5 for ODX-based ECU reprogramming and ODX-LINK V1.5 for ODX-based ECU diagnostics.

## 1.1 About this Manual

---

This manual consists of the following chapters:

- "Introduction" on page 7  
This chapter
- "Installation" on page 11  
This chapter contains tips on installing the two add-ons ODX-FLASH V1.5 and ODX-LINK V1.5.
- "Introduction to ODX-FLASH and ODX-LINK" on page 21  
This chapter contains a general introduction to ECUs and the ODX standard. The characteristics of ODX-FLASH V1.5 and ODX-LINK V1.5 are also described.
- "Working with ODX-FLASH" on page 29  
This chapter describes how to work with ODX-FLASH.
- "ODX-FLASH Tutorial" on page 39  
In this tutorial you will learn the major operational procedures in ODX-FLASH V1.5.
- "ODX-FLASH Troubleshooting" on page 57  
This chapter describes possible problems and tips on how to solve them.
- "ODX Link Menus and Functions" on page 67  
This chapter describes how to work with ODX configurations and the ODX-LINK user views as well as their configuration.
- "ODX-LINK Tutorial" on page 133  
In this tutorial, you will learn the major operational procedures for ODX-LINK V1.5.
- "ODX-LINK Troubleshooting" on page 175  
This chapter describes possible problems and tips on how to solve them.
- "ODX Communication Parameters" on page 185  
This chapter describes the communication parameters used to initialize the devices used in diagnosis or flash programming via ODX.

### 1.1.1 Target Group

---

This manual is intended for specialist personnel trained in the development and calibration of automotive ECUs. Specialist knowledge of measurement and ECU technology is assumed.

Basic knowledge of how to operate a PC and work with WINDOWS® are also assumed. All users should be able to run menu functions, activate buttons etc. Users should also be familiar with the WINDOWS file storage system, particularly with the connections between files and folders. The user must be familiar and conversant with the basic functions of WINDOWS Explorer.

### 1.1.2 Using This Manual

---

#### *Representation of Information*

---

All activities to be carried out by the user are shown in what we call a "Use-Case" format, i.e. the target to be achieved is defined briefly in the title and the individual steps necessary to achieve this target are then listed. The information is displayed as follows:

#### **Target definition**

---

Any introductory information...

- Step 1  
Possibly an explanation of step 1...
- Step 2  
Possibly an explanation of step 2...
- Step 3  
Possibly an explanation of step 3...

Any concluding remarks...

#### **Specific example:**

##### **To create a new file**

---

If you want to create a new file, no other file may be open.

- Select **File** → **New**.  
The "Create File" dialog box appears.
- Enter a name for the file in the "File name" box.  
The file name must not be more than 8 characters long.
- Click **OK**.

The new file is created and saved under the name specified. You can now work with the file.



### *Typographic Conventions*

---

The following typographic conventions are used:

Select <b>File</b> → <b>Open</b> .	Menu functions are shown in boldface/blue.
Click <b>OK</b> .	Buttons are shown in boldface/blue.
Press <ENTER>.	Keyboard commands are shown in angled brackets in block capitals.
The "Open File" dialog box appears.	Names of program windows, dialog boxes, fields etc. are shown in quotation marks.
Select the file <code>setup.exe</code> .	Text in drop-down lists, program code, as well as path and file names are shown in the <code>Courier</code> font.
A conversion between the file types logical and arithmetic is <i>not</i> possible.	Content markings and newly introduced terms are shown in <i>italics</i>

Important notes for the user are shown as follows:

#### **Note**

*Important note for the user.*

### 1.1.3 Labeling of Safety Instructions

---

The safety instructions contained in this manual are shown with the standard danger symbol shown below:



The following safety instructions are used. They provide extremely important information. Please read this information carefully.



#### **CAUTION!**

*indicates a low-risk danger which could result in minor or less serious injury or damage if not avoided.*



#### **WARNING!**

*indicates a possible medium-risk danger which could lead to serious or even fatal injuries if not avoided.*



#### **DANGER!**

*indicates a high-risk, immediate danger which could lead to serious or even fatal injuries if not avoided.*

## 1.2 Safety Instructions

---

Please refer to the general safety instructions for working with INCA and, in particular, notes on using ODX-FLASH V1.5 (see "Safety Instructions" on page 29).

**DANGER!**

*Calibration activities influence the behavior of the ECU and the systems controlled by the ECU. This may result in unexpected behavior of the vehicle and thus can lead to safety critical situations. Only well trained personnel should be allowed to perform calibration activities.*

**DANGER!**

*Sending out CAN messages influences the behavior of the CAN bus network and the systems connected to it. This may result in unexpected behavior of the vehicle and thus can lead to safety critical situations. Only well trained personnel should be allowed to perform CAN message sending activities.*

## 2 Installation

---

This chapter contains tips on installing the two add-ons ODX-FLASH V1.5 and ODX-LINK V1.5.

### 2.1 System Requirements

---

#### 2.1.1 Hardware

---

The hardware requirements for working with INCA V7.1 (or higher) are also sufficient for the add-ons ODX-FLASH V1.5 and ODX-LINK V1.5 – they are described in the manual "INCA V7.1 - Getting Started".

#### 2.1.2 Software

---

The same is true of the software requirements for working with ODX-FLASH V1.5 and ODX-LINK V1.5 as for the hardware requirements described above.

### 2.2 Installation

---

This section describes the installation of INCA-ODX.

Certain system requirements must be met to install the product. Make sure that these system requirements are met before starting the installation. The system requirements are in the section "System Requirements" on page 11.

#### To install INCA-ODX

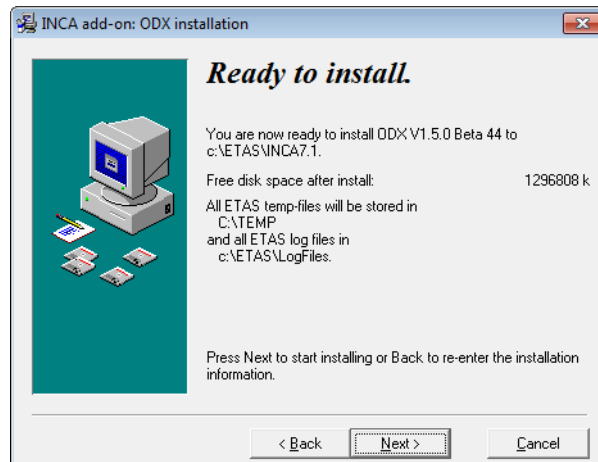
---

- Quit all programs before starting the installation.
- Insert the CD labeled "INCA-ODX" into the CD drive.  
If Autostart is enabled, a start window will appear almost right away.
- If Autostart is not enabled, run the `autostart.exe` file from the CD.
- Select the link **Main**.
- Select the INCA-ODX add-on you want to install.  
The welcome window opens.



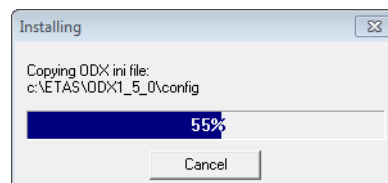
- Click **Next**.

Information on the installation directories is displayed. ODX-LINK is installed in the existing INCA directory.

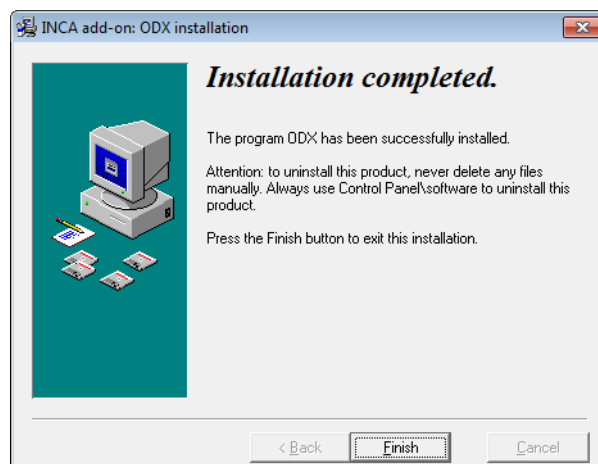


- Click **Next**.

Installation starts.



You are informed that installation has been completed in the following window.



- Click **Finish**.

This concludes the installation.

## 2.3 Licensing of the Software

---

To be able to work with an ETAS software product, you require a license. This section contains basic details on this subject.

- "ETAS License Models" on page 13
- "How to Get a License" on page 14
- "The License File" on page 15
- "Grace Mode" on page 16
- "The "Expiration Warning" Window" on page 17
- "Borrowing a License" on page 18

Details concerning the scope of the licenses and other legal aspects can be found in "Terms and Conditions".

### 2.3.1 ETAS License Models

---

There are three different license models available for licensing your ETAS software:

#### *Machine-named license, local*

---

- A license of this type is managed by the user him/herself.
- As it is linked to a particular PC (better: to the MAC address of the Ethernet adapter), it is valid wherever the PC is used.
- When you change your PC, you require a new license.

#### *User-named license, server-based*

---

- The licenses (of a department or company) are managed centrally on a server by a designated person.
- The license is linked to the user name with which the user is registered in the network and is available on every PC in the network.
- If the relevant PC is disconnected from the network, the license can be "borrowed."

#### *Concurrent (or floating) license, server-based*

---

Most of what is true of the user-named license applies to this type of license. The difference is that here several users share a limited number of licenses.

### 2.3.2 How to Get a License

---

If your company has a tool coordinator and server-based license management for ETAS software, contact this person. Otherwise (in the case of a machine-named license) you obtain your license from the ETAS license portal (the URL is shown on your Entitlement Certificate).

There are three ways of logging in on the welcome page:

- **Activation ID**

Once you have logged in, a specific activation<sup>1</sup> is visible and can be managed – the activation ID is shown on your Entitlement Certificate.

- **Entitlement ID**

All activations of the entitlement<sup>2</sup> are visible and can be managed (e.g. for a company with just one entitlement).

- **E-mail and password**

All activations of the entitlements assigned to the user account are visible and can be managed (e.g. for a tool coordinator responsible for several entitlements).

If you need help in the portal, click the [Help](#) link.

*What information is required?*

---

Information on the hosts must be entered to activate licenses:

- Machine-named license  
The MAC address of the Ethernet adapter to which the license is to be bound is required here
- User-named license  
Here, you need a server host or a server triad as well as a user name
- Concurrent (floating) license  
Here, you need a server host or a server triad.

**Note**

*If this data changes (e.g. due to changes in the hardware or a change of user), the license must be given a "rehost". This procedure is also described in the portal help file.*

*License file*

---

The result of your activities is the provision of a file `<name>.lic` with which you can license your software in the ETAS License Manager.

<sup>1</sup>. The activations refer to a specific product, its license conditions, the available number of licenses and other details required for generating a license. Activations are identified uniquely with activation IDs.

<sup>2</sup>. An entitlement shows the authorizations you have as a user; it stands for the right to own one or more licenses for a product. It is a kind of account of rights of use for software from which you can take licenses as you need to.

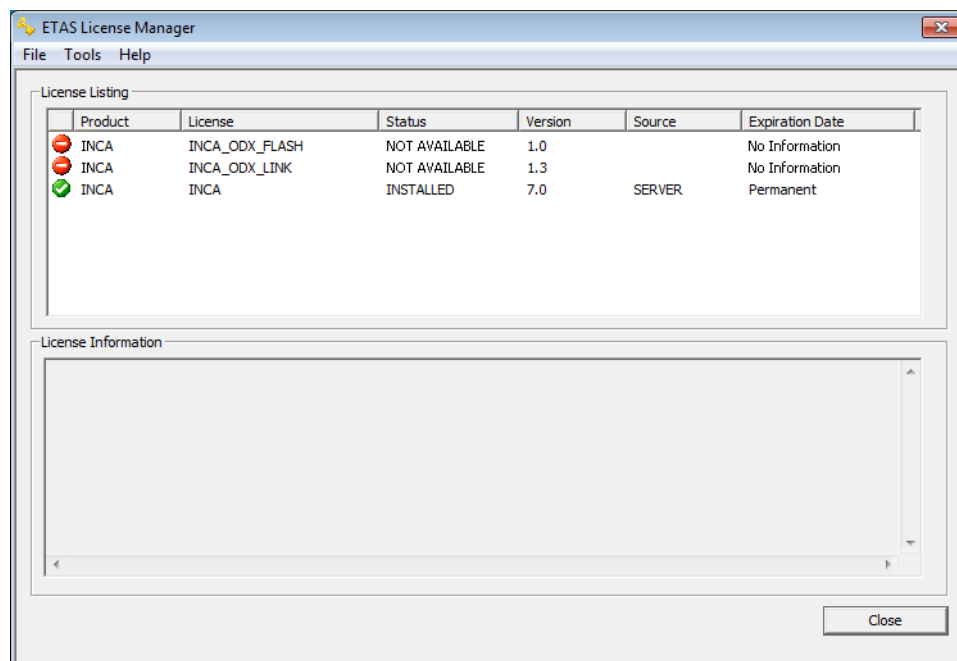
### 2.3.3 The License File

#### To check to license status

- In the Windows Start menu, select **Programs** → **ETAS** → **License Management** → **ETAS License Manager**.

The ETAS License Manager is opened. The ETAS License Manager contains one entry for each installed product.

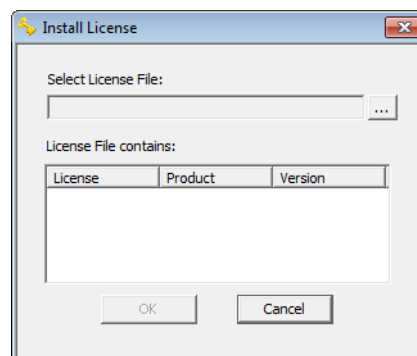
The symbol at the beginning of the entry and the "Status" column entry indicate whether a valid license has already been obtained or not.



#### To add a license file

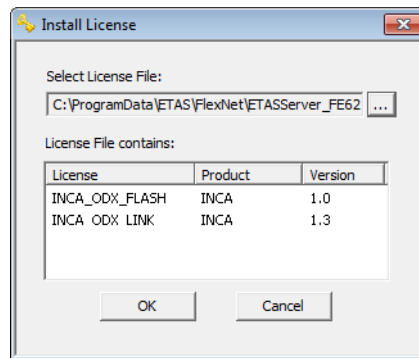
- Open the ETAS License Manager (cf. page 15) and select **File** → **Add Licensing File**.

The "Install License" dialog window opens.



- Next to the "Select License File" field click the ... button.
- In the file selection window, select the license file and click **Open**.

The "Install License" dialog window shows information on the selected license.



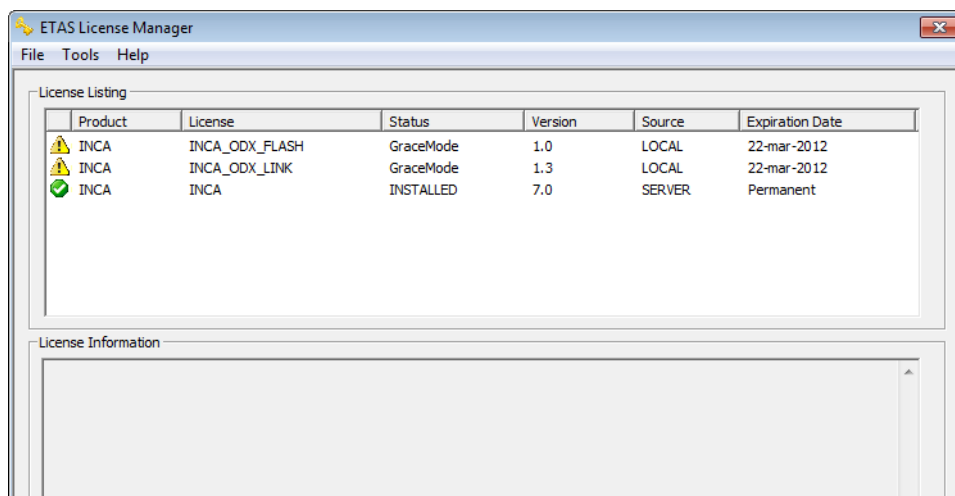
#### Note

The "Version" column shows the version number of the license, not the version number of the software.

- Confirm with **OK**.  
The license just added is now listed in the ETAS License Manager. A green symbol before the entry shows that the license is valid.
- Close the ETAS License Manager.

### 2.3.4 Grace Mode

If you have not yet installed a license, you can still operate the software for a limited amount of time – it then runs in what is referred to as grace mode. The "Expiration Date" column shows how long you can continue to operate the software in this mode.



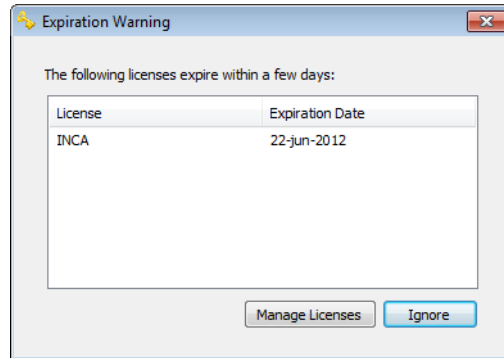


### 2.3.5 The "Expiration Warning" Window

---

If your installed license runs out in the next 30 days, a warning is shown when you open the ETAS software.

The "Expiration Warning" window contains a list of licenses that expire in the next 30 days. The expiration date is shown for each license; in the case of borrowed licenses (see "Borrowing a License" on page 18) it also shows when the borrowing period expires (i.e. the date when borrow mode runs out).



Click **Manage Licenses** to open the ETAS License Manager and install a valid license file. As soon as you have installed a valid license, you can continue to operate the ETAS software in normal operating mode.

Click **Ignore** to close the dialog box and start the ETAS software. This is only possible during the grace period; as soon as the grace period has expired, you can only continue to use the ETAS software once you have installed a valid license file.

As soon as the expiration date has been reached, you can continue to use the ETAS software for a further 14 days in what is referred to as limited mode (see "Grace Mode" on page 16). Once this phase is over, the ETAS software can only be used when a new or updated license file has been installed.

### 2.3.6 Borrowing a License

The borrowing mechanism makes it possible to work offline even when using a server-based license (i.e. without being connected to the license server).

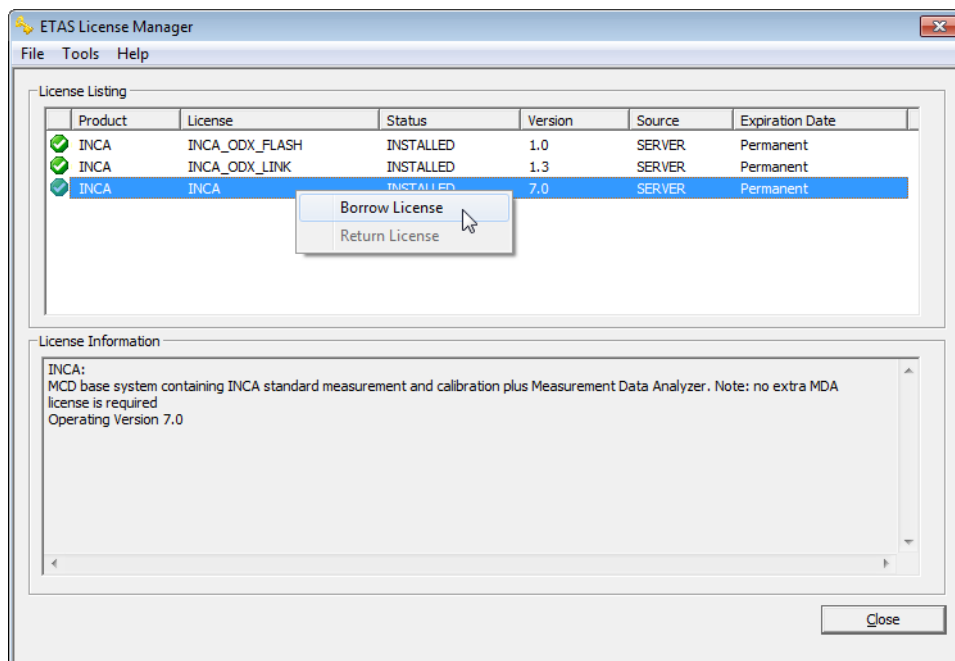
#### **Note**

*You can only borrow a license if a server-based license is being used!*

To borrow a license, proceed as follows:

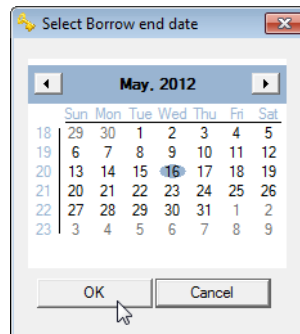
#### **To borrow a license**

- Make sure that the ETAS software the license of which you want to borrow is not open.
- Select the license you want to borrow in the "License Listing" table of the ETAS License Manager.

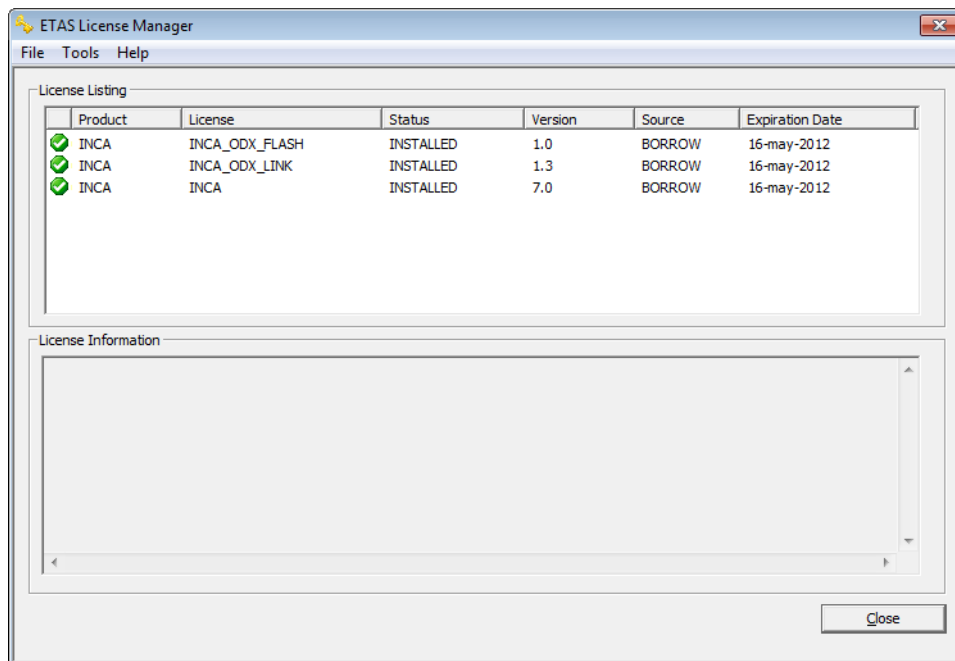


- Right-click and select **Borrow license**.  
The "Select Borrow end date" dialog box opens.

- Select the date until which you want to borrow the license from the calendar displayed and click **OK**.



The text in the "Source" column of the ETAS License Manager changes from "SERVER" to "BORROW", and the expiration date of the borrowed license is displayed.



You can now use the relevant ETAS software offline until the expiration date of the borrowed license has been reached.

If you want to use the ETAS software longer than you had originally planned, you can borrow the license again. If you stop using the ETAS software earlier than planned, you can return the license to the license server early (**Return License**). A borrowed license can only be returned by the person that borrowed it; it cannot be returned by another person.



### 3 Introduction to ODX-FLASH and ODX-LINK

---

This chapter contains a general introduction to ECUs and the ODX standard. The characteristics of ODX-FLASH V1.5 and ODX-LINK V1.5 are also described.

#### 3.1 Tasks of an ECU

---

Nowadays a modern car without electronic components is unthinkable. Every unit of a motor vehicle contains electronics: from the lights through electrically propelled starters to completely modern components like ABS and ESP.

Today's vehicles have a large number of ECUs – virtually every electronic function has its own ECU or uses an ECU. Every ECU executes different tasks. These functions can be divided into three categories:

- Controlling tasks
- Communication with other ECUs
- Diagnostics

##### *Controlling Tasks*

---

Every ECU has its own specific function area, for example gearbox, engine, or door ECU. The data the ECUs need to fulfil their tasks is received from a large number of analog and digital sensors. The ECUs receive additional data via the vehicle's bus systems (onboard communication). For example, the gearbox sends information about the gear currently selected to the engine ECU so that the latter can calculate the correct ignition time.

In addition to the actual acquisition of data (sensor system), actuating elements play an extremely important role. Target values calculated by the ECU are converted to physical values (voltage, pressure etc.) using various actuators.

##### *Communication Tasks*

---

Normally an ECU requires information from other ECUs to carry out its controlling tasks. The resulting communication with other ECUs is referred to as onboard communication. This involves an ECU sending information on the bus system and all other ECUs checking to see whether they need this information.

##### *Diagnostic Tasks*

---

The third category of task of an ECU, after the controlling tasks and communication, is diagnostics. The increased volume of electronic components inside the vehicle is providing more and more diagnostic options. Communication between ECUs and diagnostic devices is referred to as offboard as the diagnostic system is not part of the vehicle. The diagnostic system provides developers and technicians with a lot of failure-specific and vehicle information which helps them to solve vehicle problems and optimize vehicle performance.

##### 3.1.1 Diagnostics

---

Development engineers and technicians use the diagnostic functions to exchange data with the ECU. This makes it easier to detect faults and to optimize vehicle performance. When a fault occurs, for example wrong ignition timing, all relevant parameters are stored in the fault memory of the ECU. The technician reads this information using the diagnostic functions to find the cause of the fault.

Developers can also use the diagnostic functions to acquire up-to-date information on the behavior of the ECU under normal operating conditions.

Diagnostic functions can also be used during production. Internal test results can be checked or the serial number of the ECU is read to guarantee a faultless device.

### 3.1.2 Programming

---

Development ECUs are usually equipped with flash memory so that the software for the program and data version can be updated via flash programming.

The use of flash as a memory technology for the program and data version is also increasing in production ECUs. This makes software updates for ECUs possible “in the field” thanks to reprogramming of the flash memory, for example using the central offboard diagnostic interface of the vehicle. This makes it possible to update the software without having to remove the ECU from the vehicle, which means a considerable saving in terms of cost in comparison to having to exchange an ECU.

This also means an increase in flexibility in the areas:

- Software update due to improvements or changes of regulations
- Servicing new ECU series
- Activating or deactivating vehicle functions

The possibility of flashing also results in a cost reduction as only software variants are used instead of different ECU variants (platform ECUs). In addition, the exchange of hardware is avoided (e.g. changing of EEPROM chips).

The increased speed of function development and optimization means products are ready for the markets faster and also enables quality improvements in the service sector at short notice.

## 3.2 The ODX Standard

---

The ODX standard (ASAM MCD 2D) standardizes the formal description of information in the vehicle and ECU diagnostic sector – ODX stands for “Open Diagnostic Data Exchange”.

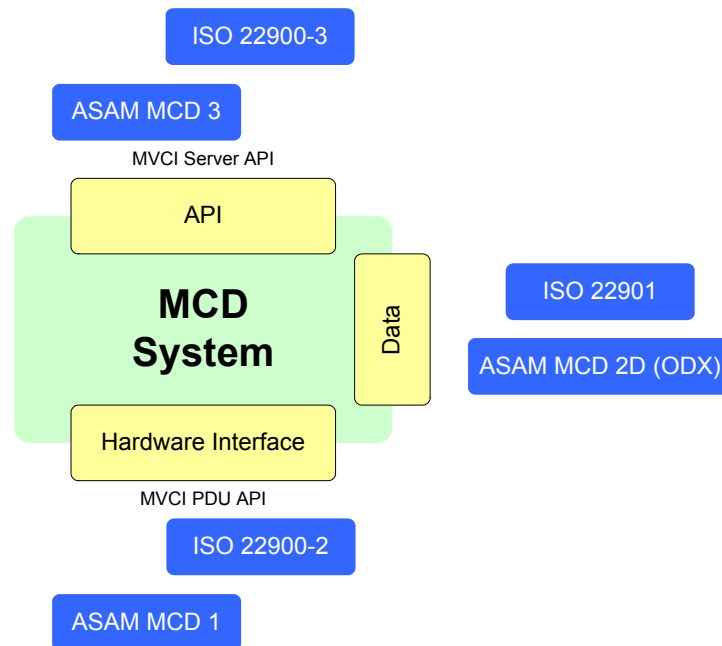
A vehicle-external repair shop tester is connected to the network of these ECUs via a special data interface, called the diagnostic interface. The tester exchanges information with the ECUs and uses message-oriented protocols for this purpose. These protocols are usually standardized (KWP2000 in acc. with ISO14230, UDS in acc. with ISO14229 etc.).

The ODX specification contains the data model for describing all diagnostic data of a vehicle/ECU. ODX is described with UML diagrams (Unified Modelling Language) – the format for data exchange is XML (eXtensible Markup Language).

The ODX specification enables the following:

- The transfer of diagnostic data and program data between system suppliers, vehicle manufacturers and repair shops in accordance with the single-source principle.
- The communication of an offboard tester with the ECUs and the subsequent interpretation of the data contained in the messages without test equipment having to be specifically programmed for this purpose.

The following figure shows an overview of the ISO standards and ASAM specifications for MCD systems.



**Fig. 3-1** ISO Standards and ASAM Specifications

Using the standardization

- of the API to the vehicle hardware interface (ISO 22900-2: D-PDU API),
- the diagnostic data model (ISO 22901-1 ODX) and
- the interface between the runtime system and the test application (ISO/ CD 22900-3: D-Server API),

it is possible, depending on the application, to combine the best hardware with the desired runtime system and the most suitable application.

### 3.3 ODX-LINK V1.5

ODX-LINK V1.5 adds diagnostic functionality to INCA – once ODX-LINK V1.5 has been installed, there are additional dialog boxes available for ECU diagnosis in the INCA experiment environment. ODX-LINK V1.5 makes it possible for you to access the fault memory, for example, during calibration and display the diagnostic information in plain text.

ODX-LINK V1.5 processes diagnostic data using the ODX description – the user can select and run every diagnostic service described in ODX.

ODX-LINK V1.5 determines all information necessary for sending the service, sends this to the ECU using the connected hardware, receives the response of the ECU with the same hardware and then decodes the response for the user.

#### *ODX Standard*

ODX-LINK V1.5 supports ODX files (including PDX = Packaged ODX) and the binary databases created with DTS-Venice which correspond to the “ASAM MCD 2D (ODX) V2.0.1” specification. This thus guarantees the use of single-

source databases throughout the entire development cycle – new implementation of diagnostic information, which is both expensive and prone to errors, is thus no longer necessary.

#### *Protocols*

---

ODX-LINK V1.5 is available as an add-on to INCA – together with INCA ODX-LINK supports the following protocols for the serial ECU interfaces:

- KWP2000 (ISO 14230-3)
- UDS (ISO 14229)
- KWPOncAN (ISO 15765)
- OBDonCAN (SAE J1979 / ISO 15031-5 on ISO 15765-4)

#### *Hardware Support*

---

Amongst others<sup>1</sup>, ODX-LINK V1.5 supports the following ETAS measure and calibration hardware:

- ES690 Compact System, ES59x Interface Modules
- ES58x ES59x Interface Modules
- ES6510 / ES520 Vehicle Interface Modules
- ES910 Rapid Prototyping Module / ES921 CAN Module
- ES511 / ES520 Interface Modules

By using ETAS hardware, measure, calibration and diagnostic access to the ECU are possible both from a user interface and via a hardware interface, e.g. UDS on CAN. This means software and hardware costs can be saved and valuable time gained to spend on development.

#### *ODX-LINK and INCA*

---

But ODX-LINK offers a great deal more than the special dialog boxes for querying diagnostic data: The diagnostic data that can be queried by the ECU (via the diagnostic interface) can be used as normal measurement signals in INCA.

This means that all INCA functions that are available for measurement signals can also be used for diagnostic data:

- configuration of measure windows with diagnostic signals in the experiment via the variable selection
- definition of trigger conditions based on diagnostic signals
- definition of calculated signals
- recording of diagnostic data in INCA measure files

Diagnostic data can thus be measured together with standard INCA measure data acquired address-based via measure interfaces such as, for example, ETK, CCP and XCP, and recorded and analyzed in a common measure file. Among other things, this makes it possible to validate and evaluate diagnostic data more precisely and efficiently than before.

<sup>1</sup>. Generally, ODX-LINK and ODX-FLASH support all INCA hardware with a CAN or K-Line port – the following is simply a list of a few examples.



### 3.4 ODX-FLASH V1.5

---

ODX-FLASH V1.5 uses ODX as the standard format for describing flash tasks and is integrated seamlessly into the INCA user interface. This includes INCA Standard-Flash-Use-Cases such as flashing from the Memory Page Manager, Quick-start- or InCircuit2 flashing etc.. ODX writes the data necessary for an upload or download.

ODX flash containers guarantee the safe and smooth exchange of information between those involved in the process. The use of the programming language Java further increases the flexibility of the flash jobs.

#### *ODX Standard*

---

ODX-FLASH V1.5 supports ODX files (including PDX = Packaged ODX) and binary databases created with DTS-Venice which correspond to the "ASAM MCD 2D (ODX) V2.0.1" specification. This thus guarantees the use of single-source databases throughout the entire development cycle – new implementation of flash information, which is both expensive and prone to errors, is thus no longer necessary.

#### *Protocols*

---

ODX-FLASH V1.5 is available as an add-on to INCA – together with INCA, ODX-FLASH supports the following protocols for the serial ECU interfaces:

- KWP2000 (ISO 14230)
- UDS (ISO 14229)
- KWPonCAN (ISO 15765)

#### *Hardware Support*

---

ODX-FLASH V1.5 supports the following ETAS measure and calibration hardware as ODX-LINK V1.5 (see "Hardware Support" on page 24).

Using these interfaces enables common access via CAN and K-Line for measuring and calibration, diagnostics and flash programming.

### 3.5 Working with ODX Projects

---

In earlier versions of INCA/ODX-LINK, the ODX project was opened and configured in the INCA experiment environment. The ODX configuration (i.e. the logical link mapping, the settings of the ODX-LINK dialog boxes, the configuration of the snapshot function etc.) was, however, stored in the ODX project in the INCA database (and not in the experiment). This meant that experiments using the same ODX project also had to use the same ODX configuration.

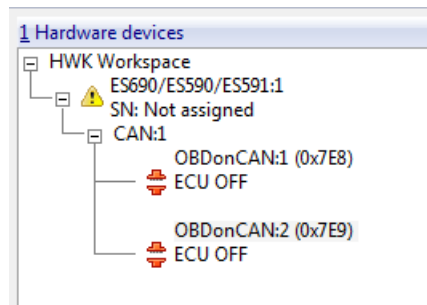
From Version V1.5 of ODX-LINK, the ODX project is part of the INCA workspace – the ODX project and the logical link are assigned in the Hardware Configuration Editor and saved with the workspace.

This standardizes the operation of ODX-LINK and ODX-FLASH and experiments of one workspace can use different ODX-LINK windows and settings.

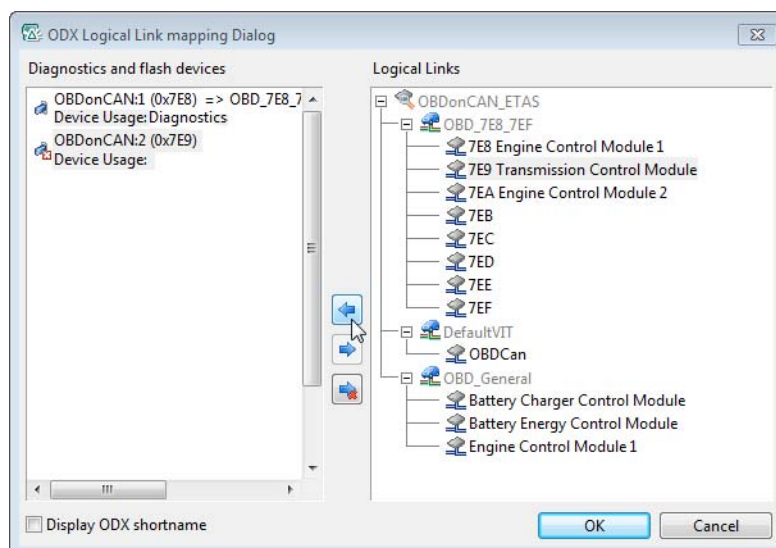
### 3.5.1 Procedure

This section takes a brief look at the procedure in INCA – for more details, refer to the two tutorials "ODX-FLASH Tutorial" on page 39 and "ODX-LINK Tutorial" on page 133.

- Create a **main directory** and **workspace**
- Add an **ECU project** if necessary  
This step can be skipped (see "Diagnostics without an A2L File" on page 27).
- Read in the **ODX project**
- Add a **hardware configuration** (diagnostic or flash devices) (see "Automatic Search for and Configuration of OBDonCAN Devices" on page 28)
  - In the Hardware Configuration Editor: **Device → Add**



- **ODX configuration**
  - In the Hardware Configuration Editor: **Hardware → Configure ODX**. Select the ODX project.
  - Logical link mapping (assignment of a logical link of the ODX project to the INCA device)



- Add **experiment** and assign **workspace**

### 3.5.2 Diagnostics without an A2L File

Communication with the connected ECUs is established during hardware initialization. The communication parameters necessary for this can be defined in A2L, CAN-DB or ODX files and must correspond to the ECUs.

A different set of parameters is also required for each protocol (KWP2000, UDS, CCP, XCP, etc.). A2L files normally contain parameters for a specific protocol and a specific bus system and cannot be used for other protocols and buses.

In earlier versions of ODX-LINK, KWP2000 and UDS devices could only be used if they were assigned an A2L file with the corresponding communication parameters. If there was no such file, a dummy file had to be created containing these KWP2000 or UDS parameters. The ODX communication parameters for hardware initialization could be used in ODX-FLASH, but not in ODX-LINK.

ODX-LINK V1.5, however, can also be used without an ECU project (in the form of an A2L file). The communication parameters for UDS or KWP2000 devices are then determined from the assigned logical link of the ODX project.

#### **Note**

*The ODX communication parameters must comply with the ISO 22900-1 specification for ODX V2.0.1 (as far as parameter names, values, units, etc. are concerned)!*

An ECU project can still be assigned to UDS or KWP2000 devices – in this case, the A2L file is the “master” and ODX parameters are ignored.

If you use a UDS or KWP2000 device in the hardware configuration, you can simply skip assigning an A2L file – after assigning an ODX project and logical link for the device, the communication parameters are read out of ODX and used during hardware initialization.

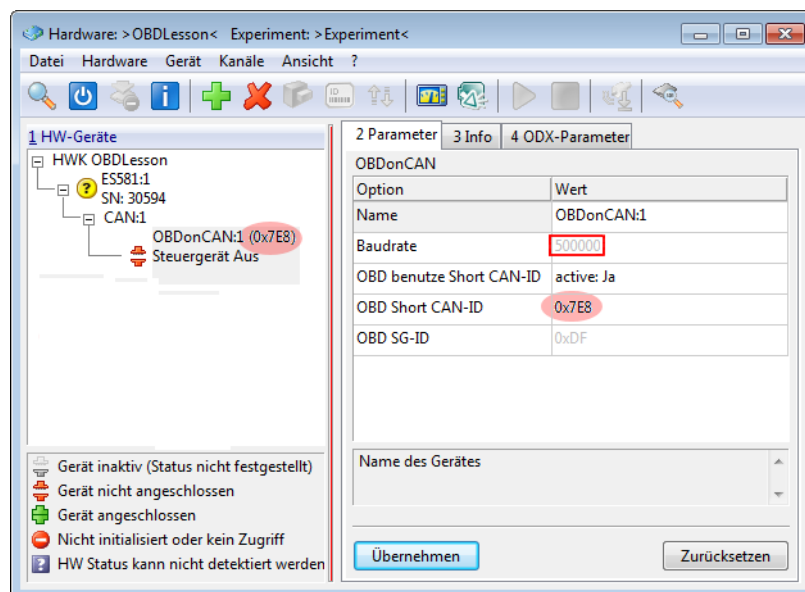
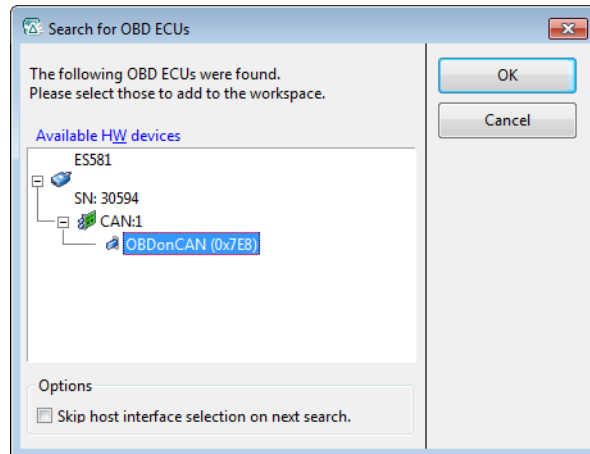
#### **Note**

*If hardware initialization via ODX communication parameters does not work, you have either assigned the wrong logical link or the ODX data does not correspond to the ECU.*

### 3.5.3 Automatic Search for and Configuration of OBDonCAN Devices

In ODX-LINK V1.5 it is possible to search for all connected devices that support OBDonCAN. To do so, select **Hardware** → **Search For OBD ECUs** in the Hardware Configuration Editor.

An OBDonCAN device with the correct OBD parameters (baud rate and CAN-ID) is automatically added to the hardware configuration for every ECU found.



For more details, refer to "Using the OBD Protocol with ODX-LINK" on page 154 in the tutorial.

## 4 Working with ODX-FLASH

---

With ODX-FLASH you can read, edit and overwrite individual data and/or code areas of an ECU.

### *Safety Instructions*

---

**DANGER!**

*Changing the ECU ROM influences the behavior of the ECU and the systems influenced by that ECU. The result of this activity can lead to undesired behavior of the vehicle and thus to safety-relevant situations.*

*Only technically experienced personnel are allowed to change the ECU memory.*

**CAUTION!**

*Once the content of the ECU memory has been overwritten, this action cannot be undone. To restore the original state, the original data has to be reprogrammed to the ECU using a flash job.*

An ODX-FLASH project consists of the following components:

- Flash job
- ODX files
- Reference to a flash file
- Security DLLs

These components are either available as individual files or in the form of an ODX-compliant PDX archive.

A PDX archive (Packaged ODX) is a compressed file which contains all components of an ODX project.

### *Flash Job*

---

The flash job contains the sequence control of ECU programming. This has to be available in the programming language Java, in accordance with the ODX standard. The sequence control can be created in any JAVA development tool, such as Eclipse, VisualJava etc.

The flash job uses the diagnostic services and security DLLs specified in the ODX files. References are made to the flash job from these ODX files. This is how flash job, ODX files and security DLLs are coordinated with one another and can be exchanged at any time.

The flash job can be in one of the following file formats:

- \*.java
- \*.class
- \*.jar

### *ODX Files*

---

The ODX files contain, for example, information on communication parameters, diagnostic services, memory area structure, vehicle information etc and are available in XML format.

Any standard ODX authoring tool which generates ODX 2.0.1-compliant ODX files (ASAM-MCD-2D -Standard Version 2.0.1) can be used to create the ODX files (e.g. DTS-Venice from Softing AG).

The following ODX files exist:

- \*.odx-c: contains the communication parameters for the connection to the ECU.
- \*.odx-d: contains hierarchically structured diagnostic levels in which the diagnostic services of the diagnostic protocol they are based on are defined.
- \*.odx-f: contains the flash containers which in turn contain the memory areas, flash sessions, the reference to the flash file and the reference to the flash job.
- \*.odx-v: contains the vehicle-specific information to connect a flash/diagnostic system to the ECU of the corresponding vehicle.

### *Flash File*

---

The flash file contains the data to be programmed.

The following file formats are possible:

- Intel HEX (\*.hex)
- Motorola S-Record (e.g. \*.s19)

### *Security DLLs*

---

The security DLLs enable access to the ECU (e.g. via *seed & key*). Specifications on how authentication is to take place are defined by the ECU manufacturer.

## 4.1 Creating an ODX-FLASH Job

---

Java Job Templates are available for the following standard protocols supported by INCA:

- UDSONCAN
- KWP2000
- KWPOncAN

The Java Job Templates are just examples of the sequence of a flash job. Adjustments have to be made to the templates when using them with a real ECU.

Any standard Java environment (e.g. Eclipse, Visual Java, etc.) can be used to create a Java job.

The ODX files and Java Job Templates provided with ODX-FLASH can be used as a template for proprietary ODX projects. These are in the folder: ETAS-Data\ODX1\_5\_0\ETASFlashLib\etas\odx\Flash.

The HTML documentation on the individual classes and methods is in the `index.html` file in the following data folder:  
ETASData\ODX1\_5\_0\ETASFlashLib\doc.

### *Structure of the Java Job Templates*

---

The structure of the main components and subroutines of the Java Job Templates is as follows for all supported diagnostic protocols:

- Initialization
- Authentication / preparation
  - Ending the diagnostic session
  - Security check
  - Starting the new programming job
  - Communication timing for the flash job
  - Information on the person responsible for the flash job started (e.g. user name from the operating system)
- Programming
  - Deleting the flash memory
  - Segment-by-segment programming
  - Verification of the newly programmed segments (using checksum algorithms)

- Post programming
  - Resetting the time parameters
  - Resetting the ECU
  - Restarting the diagnostic session

Depending on the requirements, the java jobs may also have a different structure.

#### *Files of the Java Job Templates*

---

The Java Job Templates include the following files:

- `GlobalVars.java`

This file contains the global variants which are required both by the flash job and by other classes. The initialization which is necessary at the beginning of the flash job (`executeFlashSession`) also results in the opening of the log file (see `LogChannel.java`).
- `DiagService.java`

The links between the flash job, the ODX files and the classes used are managed in this class. In addition to the subclasses for the UDS and KWP protocols, they also contain information and parameters which are used in other classes when sending diagnostic services.

  - `LogChannel.java`

This class and the methods contained in it are used to create a log file, write output and error messages to the log file and to close it again.
- `SecurityAccess.java`

This class is required for authentication and enables the release of the ECU via Seed & Key.

The following steps are executed:

  - Querying of the ECU seed
  - Loading of the security DLLs (Wrapper-DLL and Seed&Key-DLL)
  - Calculation of the key
  - Release of the ECU using the key calculated
- `Initialization.java`

This class is required for the initialization and preparation of the ECU. The following steps are executed:

  - Setting the ECU to programming status.
  - Defining the time parameters (e.g. time out, etc.).
  - Setting the user information (fingerprint).
- `Programming.java`

This class is required for programming and takes care of deleting and programming the ECU. Depending on the ECU, different flash methods (e.g. synchronous/asynchronous) are available.
- `Finalization.java`

This class is required for post programming and closes the flash job. The following steps are executed:



- Closing the programming session
- Resetting the time parameters
- Resetting the ECU to diagnostic mode
- `Utility.java`

This class contains help methods, such as methods for displaying the progress indicator and intermediate and end results in the user interface of ODX-FLASH.

#### *Messages during the Flash Job*

---

All messages which are created after the start of the flash job in the message window of the user interface of ODX-FLASH, first have to have been defined in the flash job.

The following messages are issued during the flash job using methods from the `Utility` class in the example provided with ODX-FLASH:

- Progress of the flash job (`updateProgressBar` method)
- Intermediate status (`sendIntermediateResult` method)
- Final status of the flash job (`sendFinalResult` method)

The messages issued by the flash job are also written to the `ETAS\Logfiles\ODX` directory.

## 4.2 Creating an ODX-FLASH Project with an Authoring Tool

Any standard ODX authoring tool (e.g. DTS-Venice from Softing AG) which generates ODX 2.0.1-compliant ODX files (ASAM MCD 2D-Standard Version 2.0.1) or a corresponding PDX archive can be used to create an ODX-FLASH project.

The structure of the individual ODX files is determined in the ODX 2.0.1 standard.

The ODX-C, ODX-D and ODX-F files always have to be available for a flash job.

### *ODX-C*

---

This file contains the communication parameters for the connection to the ECU.

### *ODX-D*

---

This file contains the hierarchically structured diagnostic levels in which the diagnostic services of the diagnostic protocol they are based on are defined. These services are used by the flash job to program the ECU.

### *ODX-F*

---

This file contains the flash containers. In turn, these contain the memory areas, flash sessions, the reference to the flash file and the reference to the flash job.

### *ODX-V*

---

This file contains the vehicle-specific information which is needed to connect a flash/diagnostic system to the ECU of the corresponding vehicle.

## 4.3 Working with ODX-FLASH in INCA

---

Prerequisite for working with ODX-FLASH in INCA is that INCA V7.1 and the add-on ODX-FLASH V1.5 have been installed and that there is a valid license.

The following sections contain brief information on the INCA components which are relevant when working with ODX-FLASH.

### 4.3.1 Database Manager

---

The Database Manager (DBM) creates and manages the individual database objects which are required when working with INCA. As in Windows Explorer, you can create, move and copy folders and objects in the Database Manager, as well as import and export projects, experiments and configurations. In addition, you can also create completely new databases.

For more details on the Database Manager refer to the “Working in the Database Manager” chapter of the INCA manual.

The following database objects, required for working with ODX-FLASH, are managed here:

- Workspace (incl. hardware configuration)
- Experiments
- ECU projects (A2L)
- ODX configurations

If a PDX project is selected in the DBM, then, as the case may be, project information will be shown in the window area **3 ODX Project Info**. This project information is the check history of the ODX project and is edited by the author of the ODX project.

If a user needs further information regarding a project, he can find out in this way who edited the project last.

### 4.3.2 Hardware Configuration Editor

---

In the Hardware Configuration Editor (HWC Editor), it is possible to manage and configure the hardware for the active workspace. In addition to the possibility just described of making hardware which corresponds to the project description file known to the software in the Database Manager, you can also add additional components from a list of all the possible modules (e.g. measuring hardware) here.

For more details on the Hardware Configuration Editor refer to the “Working with the Hardware Configuration Editor” chapter of the INCA manual.

### 4.3.3 Memory Page Management

Memory page management is a component of the Hardware Configuration Editor and is used to manage different datasets (e.g. working and reference datasets). This is how, for example, datasets can be read into and out of the ECU, data can be copied from the working to the reference dataset and vice versa, working datasets stored or write-protected as "interim results" and reference data sets changed for the current project.

Memory page management can be activated either from the experiment environment or from the hardware configuration.

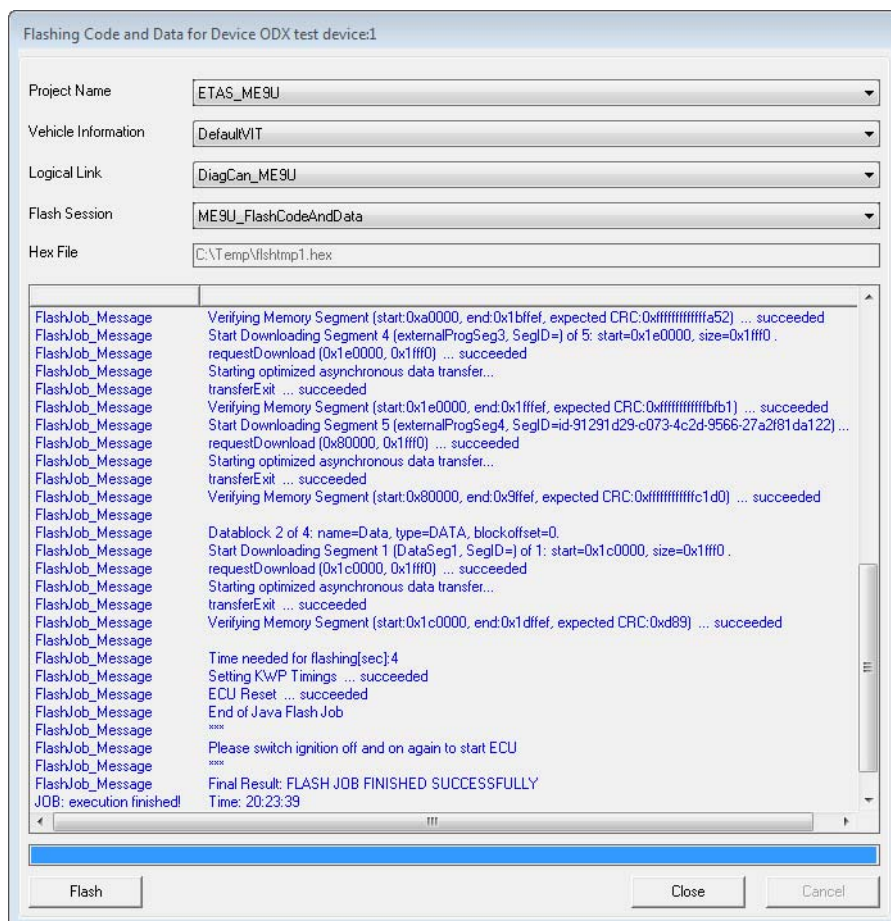
### 4.3.4 ODX-FLASH

In ODX-FLASH, you enter the final settings for the flash job and run it.

The selected settings are stored and used again as standard settings the next time the flash tool is launched.

#### *ODX-FLASH User Interface*

The following figure shows the ODX-FLASH user interface.



The meaning of the individual fields and buttons is described below.

- “Project Name” field  
This is where the ODX project to be used for the flash job is selected. The ODX project assigned to the device selected for the flash job in the hardware configuration is already predefined as default.
- “Vehicle Information” field  
The vehicle variant of the ECU for which the flash job is to be run is selected here. The possible vehicle variants are defined in the ODX project.
- “Logical Link” field  
The connection protocol for the connection between the device selected in the hardware configuration and the ECU is selected here. If there are several connection protocols available, they are displayed as a list.
- “Flash Session” field  
This is where the actions to be run during the flash job are selected. For example, you can select whether just the data area, just the code area or the code and data area are to be reprogrammed. The actions available are defined in the ODX project.
- “Hex File” field  
This is where the flash file which contains the information to be flashed is displayed. If you flash with data from an external file, the name of this file is displayed here.  
  
If no external file is used for flashing, the file is generated by INCA as a temporary hex file. This file is based on the hex file of the A2L project (in the INCA DB) and added calibration data.
- “ASAP2 Flash ID” field  
This is where the flash ID of the ECU project is displayed.
- “ODX Flash ID” field  
This is where the flash ID of the ODX project assigned to the ECU project is displayed.

**Note**

*Flash IDs are only displayed if the relevant projects have actually been assigned a flash ID (see "The ODX Flash ID" on page 38).*

- **Flash** button  
This button is used to start the flash job.
- **Close** button  
This button saves the changes made and closes the flash tool without starting the flash job.
- **Cancel** button  
This button closes the flash tool without saving the changes and without starting the flash job.

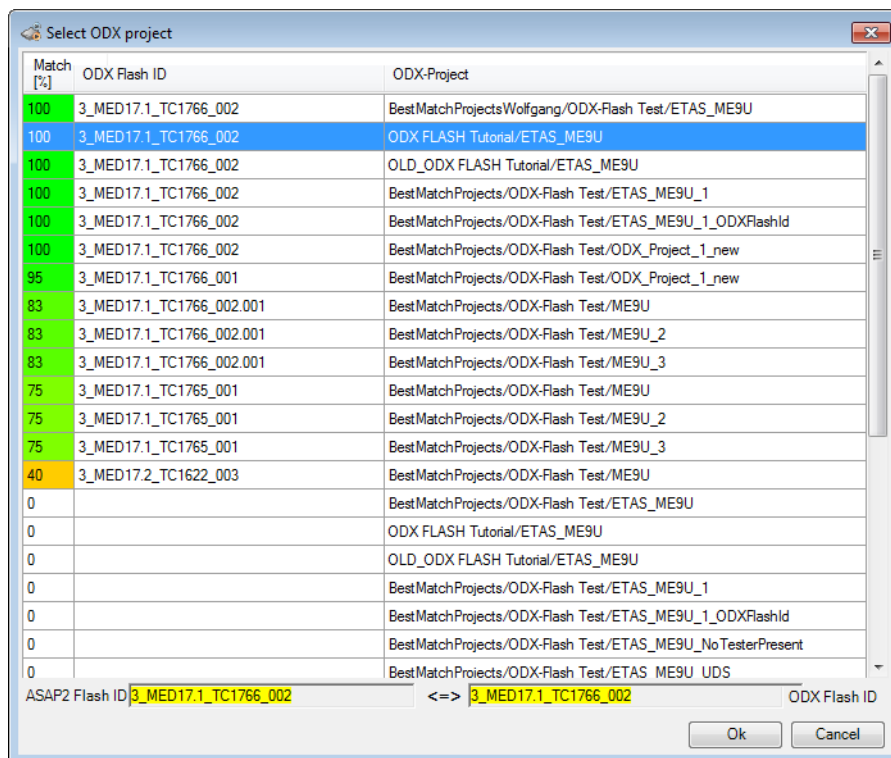
### The ODX Flash ID

The ODX flash ID is intended to simplify the assignment of an ODX project (for flashing an ECU) to an ECU project (in the form of an A2L file).

#### Note

*ECU projects and ODX projects do not have to be assigned flash IDs – in this case, assignment takes place without the following dialog box.*

If the ECU project has a flash ID, the following window opens during the assignment of an ODX project (**Hardware** → **Configure ODX** in the Hardware Configuration Editor) which is also assigned one (or even more) flash IDs.



This is where the flash IDs of all ODX projects in the database are compared to the ID of the ECU project and the degree to which the names of these IDs match is shown in the “Match [%]” column.

The longest ID string is decisive here, so “Match” is somewhere between 0 and 100%. If an ODX project does not have a flash ID, it is shown in the list as “Match = 0%”.

A project which has been selected once before is shown on a colored background. An ODX project can be listed several times because it can contain several flash IDs which correspond in varying ways to the flash ID of the ECU project.

To select the ODX project, choose the line you require and click **OK**.

## 5 **ODX-FLASH Tutorial**

---

In this tutorial you will learn the major operational procedures in ODX-FLASH V1.5.

You do not need any additional devices for this exercise. The ECU and the device via which the flash job is executed are simulated in this exercise by the "ODX test device". The measuring/calibration device is simulated by the "ETK test device".

For this tutorial, you need to have installed INCA V7.1 and ODX-LINK V1.5/ODX-FLASH V1.5. For details on the installation of ODX-LINK V1.5/ODX-FLASH V1.5, refer to the section "Installation" on page 11. For this tutorial, you should be familiar with basic INCA operations.

The tutorial contains the following lessons:

- **Lesson 1: Creating an INCA Workspace and Configuring the Devices**

Here you prepare the INCA system for the "ODX Flash Tutorial" application by setting up the database required for the task, setting up a workspace, adding and then configuring the required devices.

- **Lesson 2: Defining ODX Parameters**

In this lesson, you determine which device is to be used as the measuring/calibration device, which device is to be used as interface for programming the ECU and which ODX project is to be used for this.

- **Lesson 3: Making Settings for the Flash Job in Memory Page Management**

In this lesson, you define the tasks to be executed during the flash job in memory page management.

- **Lesson 4: Executing the Flash Job from an Experiment**

In this lesson, you execute the flash job directly from an experiment. This means you can transfer the settings you have made in the experiment directly to the flash memory of the ECU.

- **Lesson 5: Flashing with External Data**

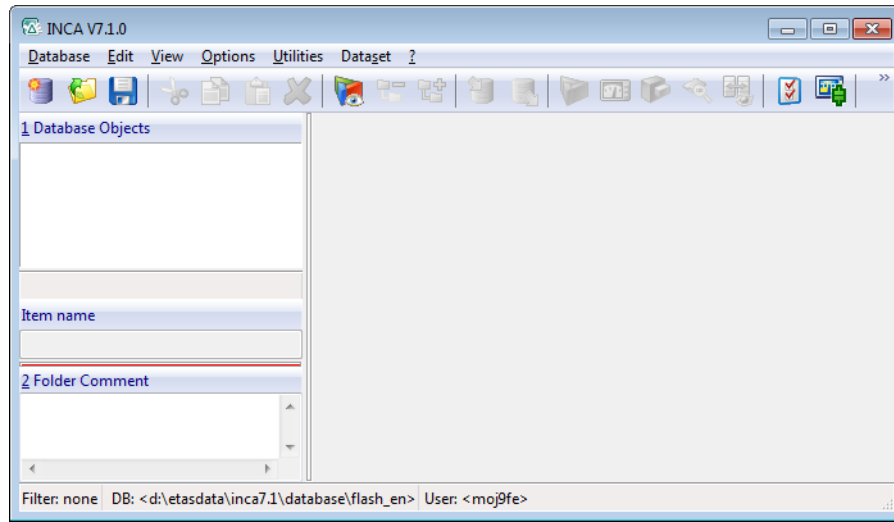
In this lesson, you write information directly from an external hex file to the flash memory of the ECU during the flash job.

## 5.1 Lesson 1: Creating an INCA Workspace and Configuring the Devices

In this lesson, you will create a new INCA workspace for working with ODX-FLASH and add, and then configure, the necessary devices.

### To create a top folder and INCA workspace

- Start INCA.  
The Database Manager (DBM) opens.



- Create a new top folder. Select **Edit** → **Add** → **Add Top Folder**.
- Enter `ODX-FLASH Tutorial` and press <ENTER>.
- Add a new workspace. Select **Edit** → **Add** → **Workspace**.
- Enter `Workspace` and press <ENTER>.

or



- Click the **Add Workspace** icon. Enter `Workspace` and press <ENTER>.

### To add an ECU project

- Select **Edit** → **Add** → **ECU Project**.  
A window opens in which you can select the ECU project.

or



- Click the **Add Project** icon.  
A window opens in which you can select the project description file.
- In the INCA data folder `ETAS-Data\INCA7.1\Data\Demo` select the `ODX-FLASH Tutorial-SimETK.a21` file.



- Click **Open**.  
A window opens in which you can select the project program file.
- In the INCA data folder  
ETASData\INCA7.1\Data\Demo select the  
ODX-FLASH Tutorial-SimETK.s19 file.
- Click **Open**.  
The ODX-FLASH Tutorial-SimETK project is added.
- Add a second ECU project, ODXTestDevice.a21, as described above from the same folder.
- As no ECU program file is required for this project, select **Cancel** in the next file selector window.  
The ODXTestDevice project is also added.

### To add an ODX project

---

- Select **Edit** → **Add** → **ODX Project**.

or



- Click the icon.  
A window opens in which you can select the project or the ODX files.
- In the INCA data folder ETAS-Data\ODX1\_5\_0\Projects\  
ETAS\_ME9U select the ETAS\_ME9U.prj file.
- Click **Open**.  
The ODX project "ETAS\_ME9U" is added.

### To define a flash device

---

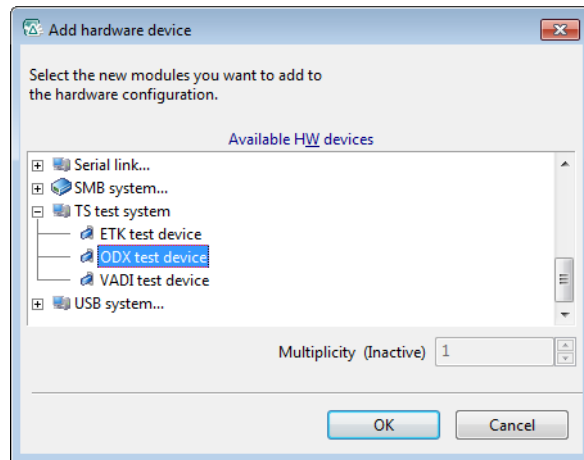
- Select the "Workspace" workspace.
- To add a new device, select **Device** → **Configure hardware...**  
The hardware configuration window opens.
- To add the "ODX test device" hardware component, select **Device** → **Insert...**

or

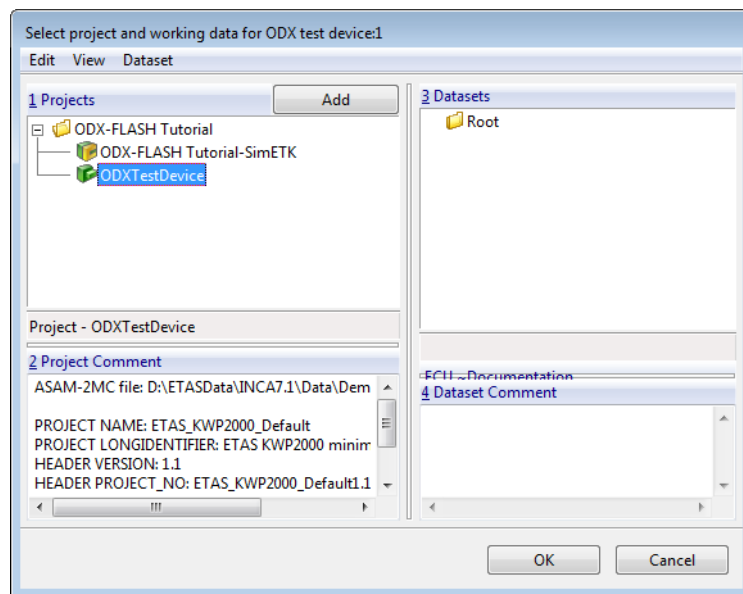


- Click the **Insert...** icon.  
The "Add hardware device" window is displayed.

- In the TS-Testsystem folder select "ODX test device".

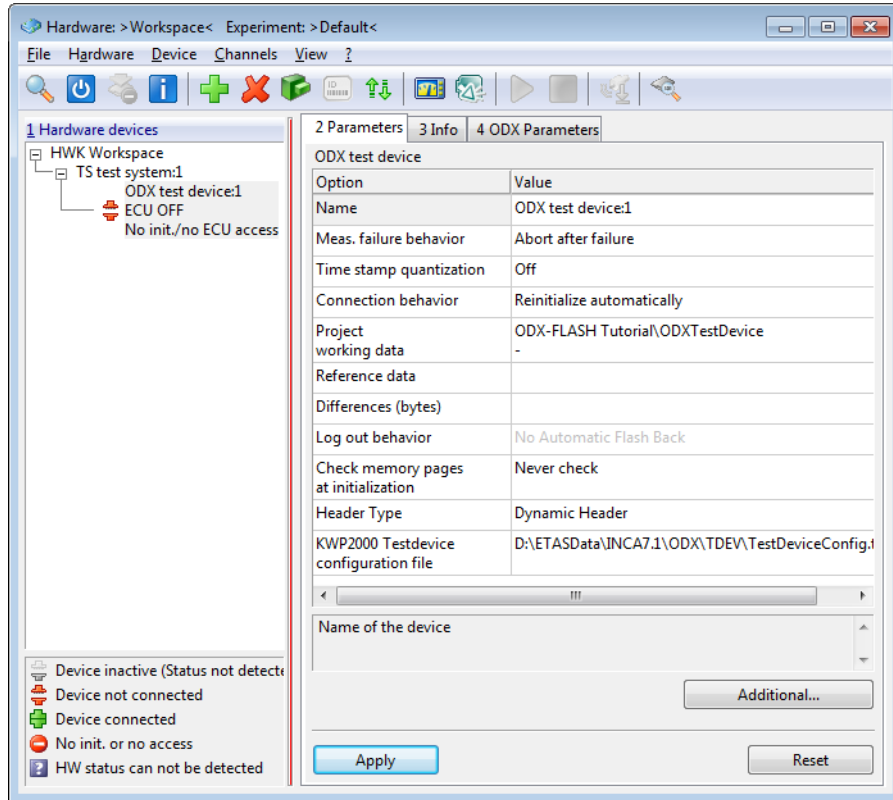


- Click **OK**.  
A window opens in which you can select the project.



- Select the „ODXTestDevice“ project and click **OK**.  
A window opens in which you can select a TDEV file.
- Select the configuration for the test device. Navigate in the INCA data folder `ETASData\INCA7.1\ODX\TDEV` and select the `TestDeviceConfig.tdev` file.

- Click **Open**.  
The Hardware Configuration Editor opens and the added device is displayed in the “Hardware devices” window.



The request/response behavior of the simulated ECU is stored in the TDEV file. When you are working with real devices, you do not have to select a TDEV file and this dialog box will not be displayed. If you select a different device or ODX project to be simulated, you may require a different TDEV file.

### To define the measuring/calibration device

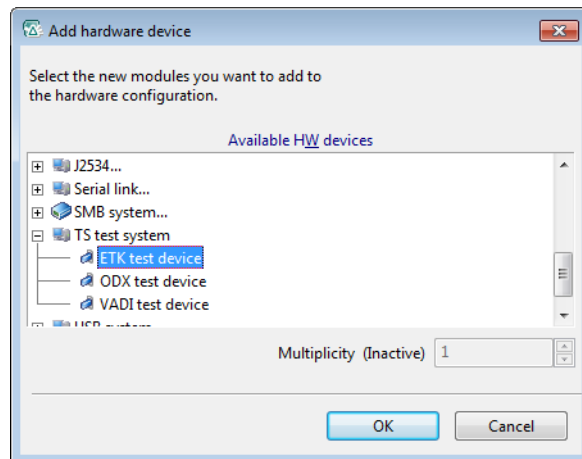
- In the Hardware Configuration Editor select **Device** → **Insert...**

or



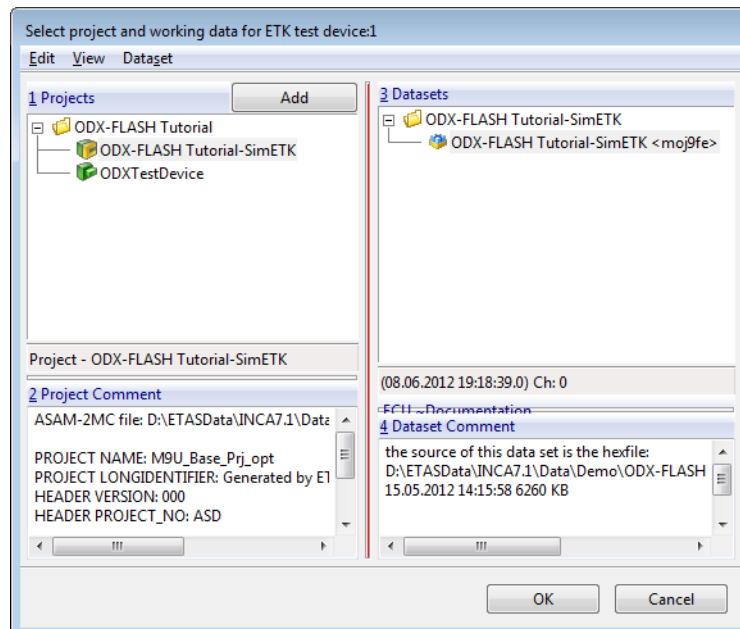
- Click the **Insert..** icon.

The “ Add hardware device ” window is displayed.



- In the **TS test system** folder select the **ETK test device** device.
- Click **OK**.

A window opens in which you can select the project.



- Select the **ODX-FLASH Tutorial-SimETK** project.

- Click **OK**.
- Leave all the windows open for the next exercise.

In this exercise, you have created a new INCA workspace and configured a simulated flash device connected via CAN and a simulated measuring/calibration device connected via ETK in the hardware environment.

## 5.2 Lesson 2: Defining ODX Parameters

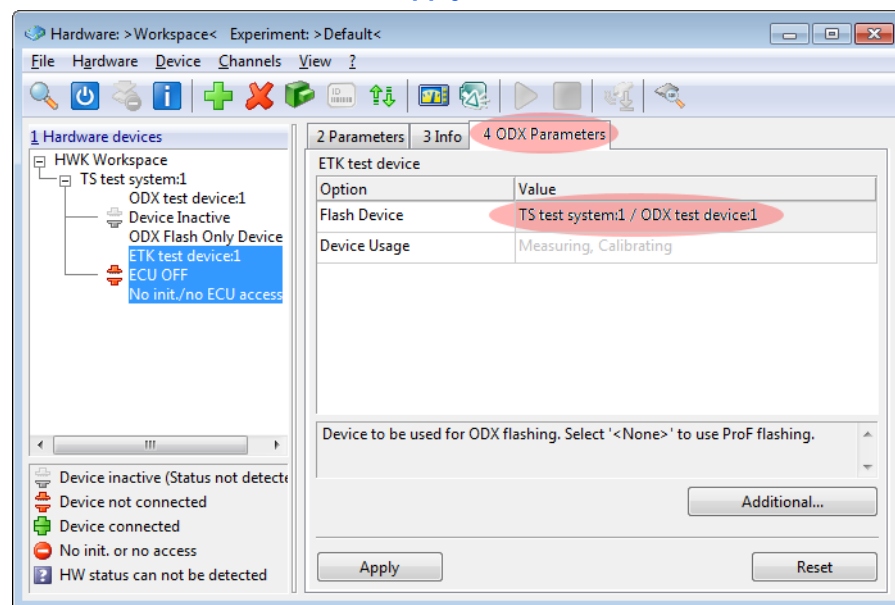
In this exercise, you specify which of the selected devices is to be used as the flash device and assign it the ODX project to be used.

This lesson follows on directly from the previous one.

### To specify a flash device

You are in the Hardware Configuration Editor.

- In the “Hardware devices” box, select the measuring/calibration device **ETK test device**.
- Select the “ODX Parameters” tab.
- In the “Flash Device” box, select **ODX test device** which is to be used to program the ECU.
- Click **Apply**.



The “Device Usage” box now displays what the device is to be used for – these values cannot be changed in this box.

### To assign an ODX project to the flash device

You are in the Hardware Configuration Editor.

- In the “Hardware devices” window, select the flash device **ODX test device**.
- In the “Flash Device” box, select the option **<None>**.

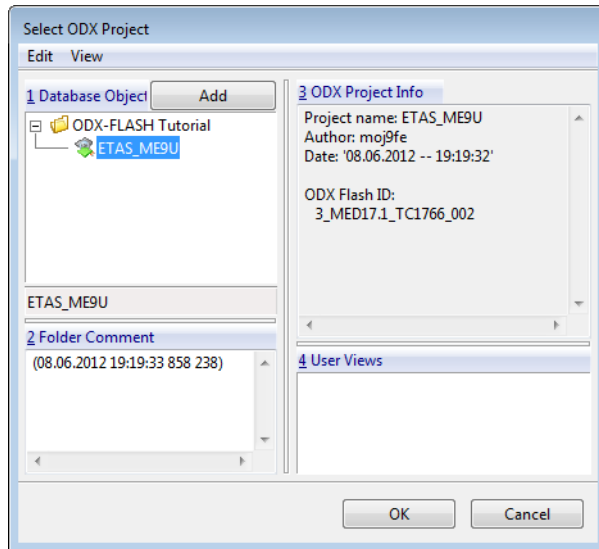
- Select **Hardware** → **Configure ODX**

or

- Click the **Configure ODX** icon.



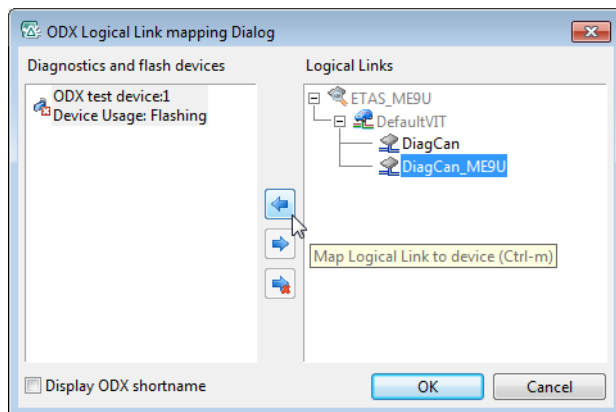
The "Select ODX Project" dialog box is displayed.



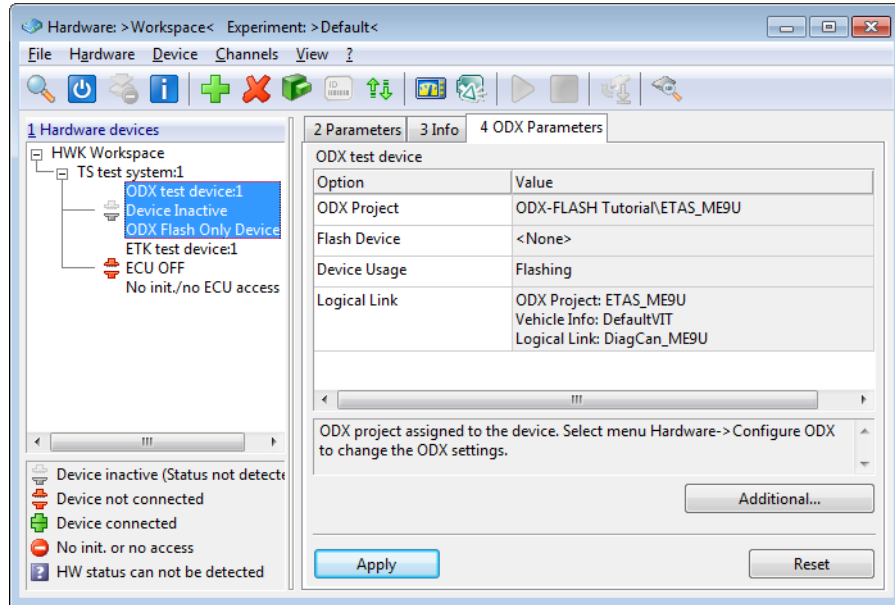
- Select the database object "ETAS\_ME9U" as ODX project and click **OK**.

The "ODX Logical Link mapping Dialog" window opens.

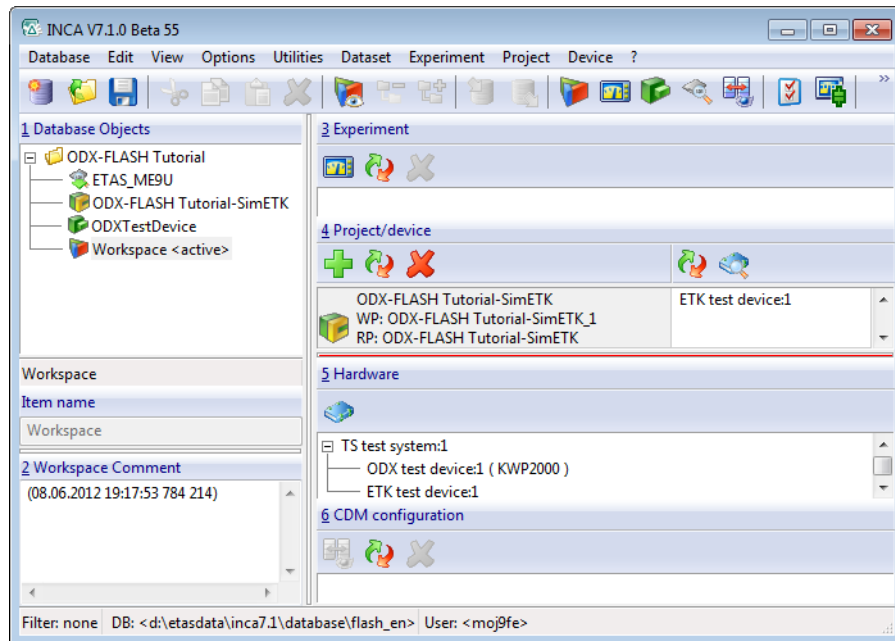
- Select the "DiagCan\_ME9U" logical link and assign it to the flash device using the arrow pointing left.



- Click **OK**.  
The ODX project is now assigned to the flash device.



- Close the Hardware Configuration Editor.



ODX projects with this icon can be used to program the ECU memory.

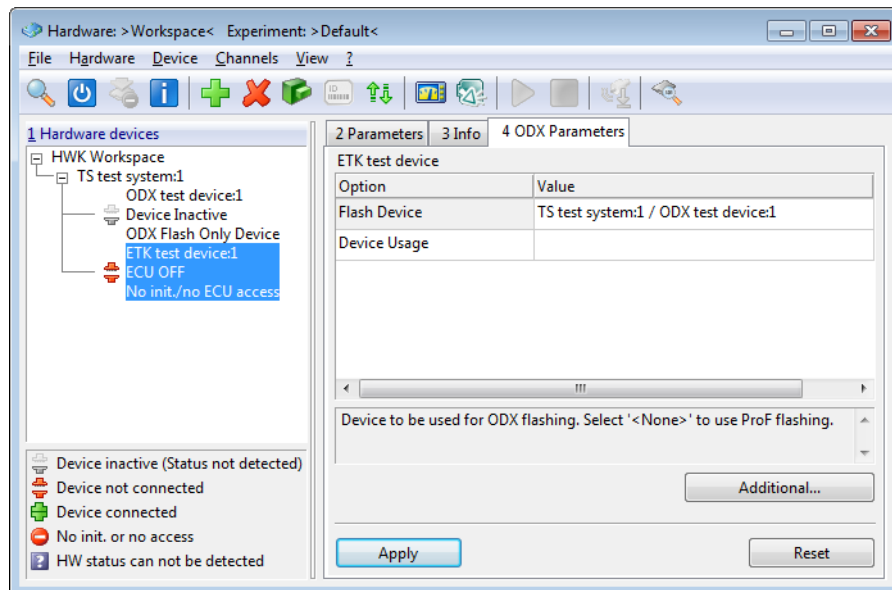


ODX projects with this icon can only be used with ODX-LINK. Programming is not possible with these ODX projects as not all ODX components or references which are required for programming are contained.

### 5.3 Lesson 3: Making Settings for the Flash Job in Memory Page Management

In this lesson, you activate memory page management and specify the settings for the flash job.

- Open the Hardware Configuration Editor.
- In the “Hardware devices” box, select the measuring/calibration device **ETK test device**.



- To open the memory page management, select **Device** → **Manage memory pages**.

or



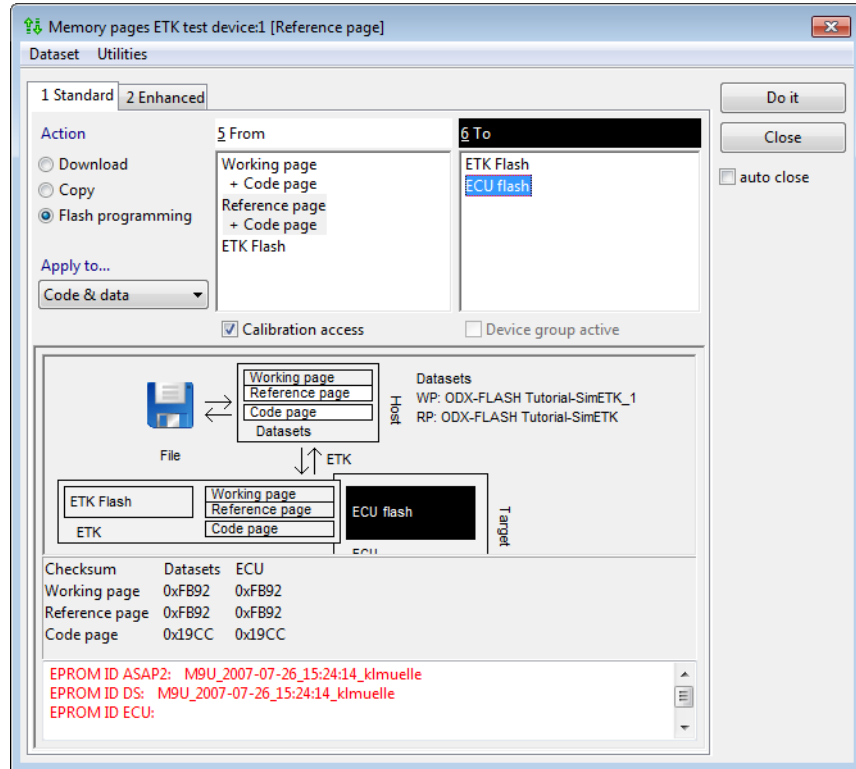
- Click the **Manage memory pages** icon.
- Memory page management opens.



## To make settings for memory page management

You are in Memory Page Management.

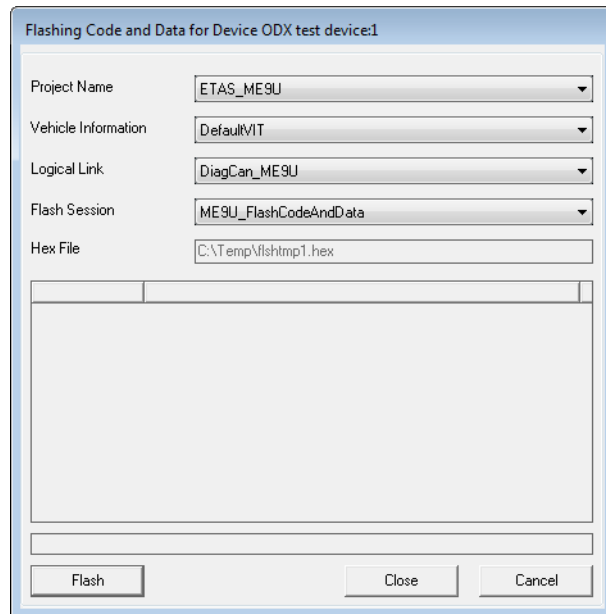
- Select the "Standard" tab.



- In the "Action" box, select the option "Flash programming".
- In the "Apply to" drop-down list, select the option "Code & data".
- In the "From" box, select the option "Reference page + Code page".
- In the "To" box, select the option "ECU flash".
- Click **Do it**.  
ODX-FLASH opens.

### To make settings in ODX-FLASH and execute the flash job

The title bar of the window shows the job to be executed and the flash device. In this example "Flashing Code and Data for Device ODX-Testdevice:1"



- In the "Project Name" box, enter the file name of the ODX project to be used. In this example, the ODX project "ETAS\_ME9U" is the only choice.
- In the "Vehicle Information" box, enter the vehicle for which the ECU is to be programmed. In this example, "DefaultVIT" is the only option available. Information about the vehicles the ECU can occur in is specified in the ODX project.
- In the "Logical Link" box, enter the connection protocol to be used for the connection between the flash device and the ECU. In this example, select the option "DiagCan\_ME9U".
- In the "Flash Session" box, enter the session to be executed during the flash job. In this example, select the option "ME9U\_FlashCodeAndData".
- Click **Flash**.  
The ECU is now programmed.

Information about which vehicles, which connection protocols and which flash sessions are available are specified in the ODX project.

## 5.4 Lesson 4: Executing the Flash Job from an Experiment

---

In this lesson, you execute the flash job directly from an experiment.

The prerequisite for this is that you have completed lessons 1 and 2. You should also be familiar with the basics of working with experiments in the INCA experiment environment.

Start by creating a new experiment in the existing top folder. This experiment is only an example object to explain how to call a flash job from an experiment.

For details on how to configure experiments refer to the INCA manual.

### To create an experiment

---

- Change to the “ODX FLASH Tutorial” top folder.
- Create a new experiment. Select **Edit** → **Add** → **Experiment**.
- Enter `ODX-Tutorial Experiment` and press `<ENTER>`.

or

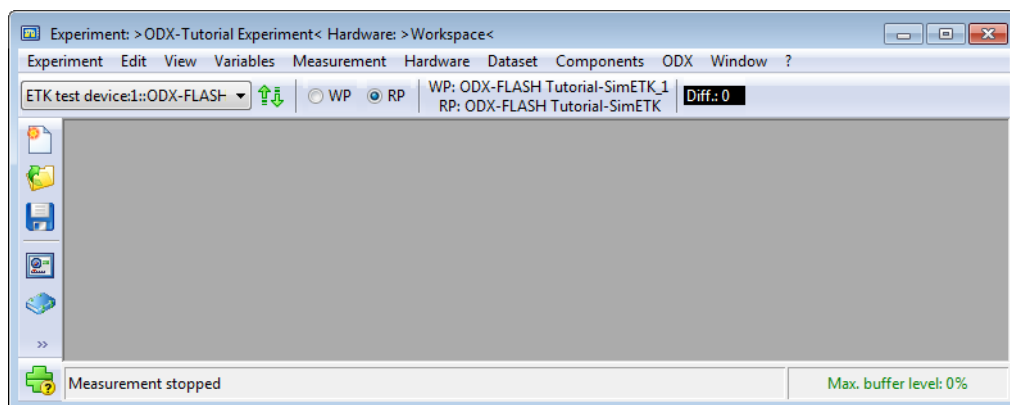


- Click the **Add Experiment** icon.
- Enter `ODX-Tutorial Experiment` and press `<ENTER>`.

### To assign an experiment to the workspace and open it

---

- Double-click the experiment `ODX-Tutorial Experiment`.  
The “Select workspace” window opens.
- Select “Workspace” and click **OK**.  
The experiment opens in the experiment environment.



## To call the memory page management from the experiment

You are in the experiment environment.

- Open memory page management. For this purpose, select **Hardware** → **Manage memory pages...**

or

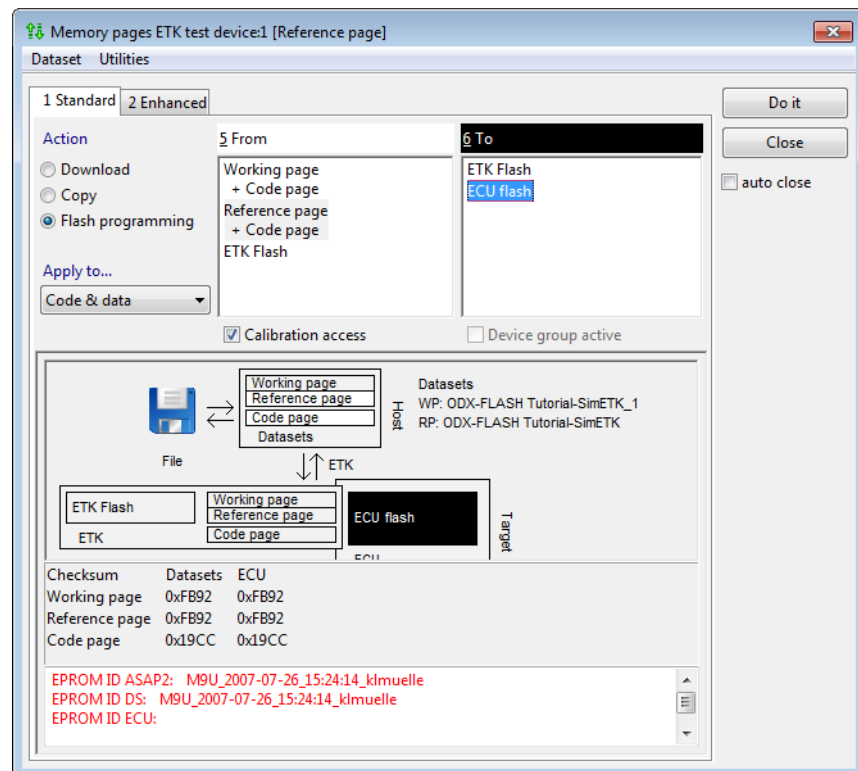


- Click the **Manage memory pages...** icon.  
The memory page management opens.

## To make settings for memory page management

You are in memory page management.

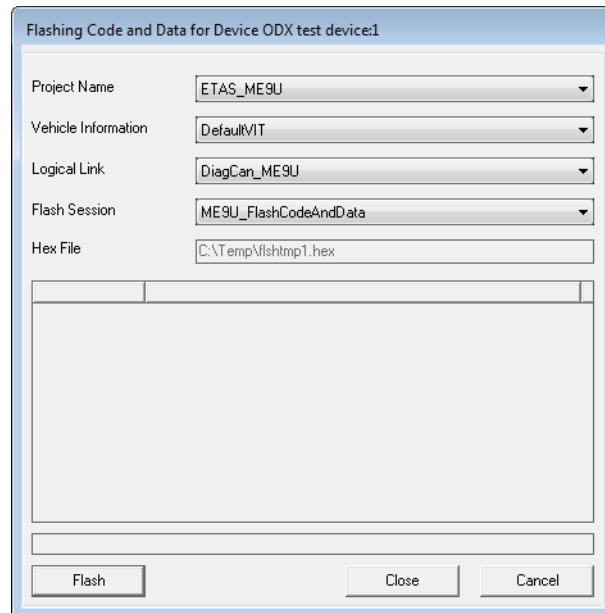
- Select the "Standard" tab.



- In the "Action" box, select the option "Flash programming".
- In the "Apply to" drop-down list, select the option "Code & data".
- In the "From" box, select the option "Reference page + Code page".
- In the "To" box, select the option "ECU flash".
- Click **Do it**.  
ODX-FLASH opens.

### To make settings in ODX-FLASH and execute the flash job

The title bar of the window shows the job to be executed and the flash device used. In this example: "Flashing Code and Data for Device ODX-Testdevice:1".



- In the "Project Name" box, enter the file name of the ODX project to be used. In this example, the ODX project **ETAS\_ME9U** is the only choice.
- In the "Vehicle Information" box, enter the vehicle for which the ECU is to be programmed. In this example, "DefaultVIT" is the only option available. Information about the vehicles the ECU can occur in is specified in the ODX project.
- In the "Logical Link" box, enter the connection protocol to be used for the connection between the flash device and the ECU. In this example, select the option "DiagCan\_ME9U".
- In the "Flash Session" box, enter the session to be executed during the flash job. In this example, select the option "ME9U\_FlashCodeAndData".
- Click **Flash**.  
The ECU is now programmed.

Information about which vehicles, which connection protocols and which flash sessions are available are specified in the ODX project.

## 5.5 Lesson 5: Flashing with External Data

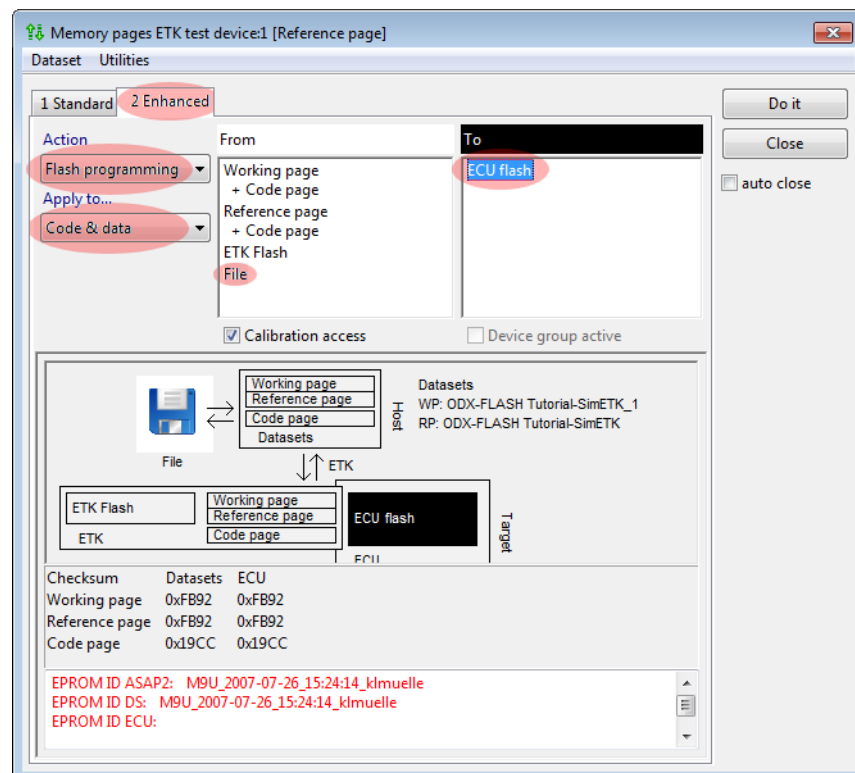
In this lesson, you learn how saved values and settings for a flash job can be used in a flash file. This can be useful, for example, if the same information is to be transferred to several ECUs or if the settings were not created on the flashing system (INCA) but on an external one.

The prerequisite for this is that you have completed lessons 1 and 2.

### To make settings for memory page management

You are in memory page management.

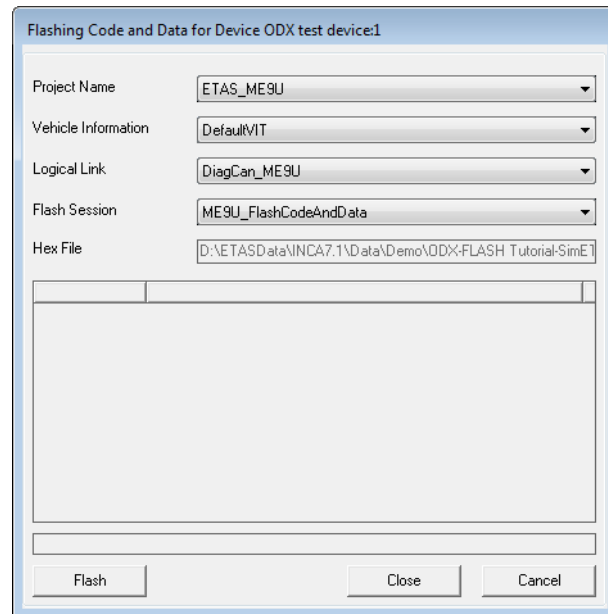
- Select the "Enhanced" tab.



- In the "Action" box, select the option "Flash programming".
- In the "Apply to" drop-down list, select the option "Code & data".
- In the "From" box, select the option "File".
- In the "To" box, select the option "ECU flash".
- Click **Do it**.  
The window for reading in the flash file opens.
- In the INCA data folder  
ETASData\INCA7.1\Data\Demo select the  
ODX-FLASH Tutorial-SimETK.s19 file.
- Click **Open**.  
ODX-FLASH opens.

## To make settings in ODX-FLASH and execute the flash job

You are in ODX-FLASH.



The title bar of the window shows the job to be executed and the flash device. In this example: "Flashing Code and Data for Device ODX test device:1".

- In the "Project Name" box, enter the file name of the ODX project to be used.  
In this example, the ODX project "ETAS\_ME9U" is the only choice.
- In the "Vehicle Information" box, enter the vehicle for which the ECU is to be programmed.  
In this example, "DefaultVIT" is the only option available. Information about the vehicles the ECU can occur in is specified in the ODX project.
- In the "Logical Link" box, enter the connection protocol to be used for the connection between the flash device and the ECU.
- In this example, select the option "DiagCan\_ME9U".
- In the "Flash Session" box, enter the session to be executed during the flash job.
- In this example, select the option "ME9U\_FlashCodeAndData".
- Click **Flash**.  
The ECU is now programmed.

Information about which vehicles, which connection protocols and which flash sessions are available are specified in the ODX project.

## 5.6 Summary

---

These are the tasks you have executed in the lessons in the tutorial:

- Created a top folder and INCA workspace
- Added an ECU project
- Added an ODX project
- Defined a flash device
- Defined a measuring/calibration device
- Assigned an ODX project to the flash device
- Made settings for memory page management
- Made settings in ODX-FLASH and executed the flash job

You have also completed the following tasks to enable the flash job to be executed from an experiment:

- Created an experiment
- Called an experiment
- Called memory page management from the experiment

All configurations you have executed in this tutorial have been supplied as a demo configuration in an export file. You can import this demo configuration containing all database objects.

### To import the demo configuration

---

- In the INCA main window select **Edit → Import**.  
A window opens in which you can select the input options.
- Click **OK**.  
A file selector window opens.
- In the INCA data folder  
ETASData\INCA7.1\Export\ODX select the  
ODX-FLASH-Tutorial.exp file.
- Click **OK**.  
The top folder ODX-FLASH Tutorial is  
imported and displayed in the “Database Objects”  
window.



## 6 ODX-FLASH Troubleshooting

---

### 6.1 General Errors

---

Error	Remedy
Hardware Configuration Editor cannot be opened from the DBM.	Check whether a database object of the type "Workspace" has been selected.
Memory page management cannot be opened from the Hardware Configuration Editor .	Make sure a measuring/calibration device has been selected.
The required vehicle type (Vehicle Information) is not displayed in ODX-FLASH.	Use a different ODX project which is designed for the required vehicle type.
The desired action (Flash Session) is not available in ODX-FLASH.	Check whether the selected connection protocol (Logical Link) is suitable for the desired action.
The desired project (Project Name) is not available in ODX Flash.	Make sure that you have specified the name of the basic file here. Project name and file name may differ.

## 6.2 Errors when Executing the Flash Job

Error Message	Error Description	Remedy
Error in TP_BLOP generation: Com-Param * is out of valid range	The ODX-COM-PARAM-Spec contains a parameter whose default value is outside the valid range.	Check the parameters in terms of their validity range (see "ODX Communication Parameters" on page 185).
Error in TP_BLOP generation: Com-Param * not found	A parameter is missing from the ODX-COM-PARAM-Spec which is required for initializing the flash device.	Check whether all parameters necessary for initializing the flash device are available (see "ODX Communication Parameters" on page 185).
Error in TP_BLOP generation: Sending a Tester Present Message without parameters is not allowed for *	The ODX-COM PARAMs define a "Tester present" message without parameters. UDS does not support this.	Set the byte size of the "CP_TesterPresentMessage" parameter to a value >1.
Error when start flashings	When an attempt was made to start the flash job, an error occurred in the MVCI Server.	Please read the notes in the protocol window of the DiagServer.
ODX-FLASH is not licensed	No valid license for ODX-FLASH could be found.	Check your license for ODX-FLASH. Contact your ETAS Support.
Within flash session '*' the hex file is not replaceable	The ODX-FLASH Container has no reference to an external flash file. INCA does not support this.	Assign an external flash file to the ODX-FLASH container.
Flash Job finished without final result	The flash job did not return a result.	Edit the flash job so that a result is always produced.

Error Message	Error Description	Remedy
Can't write '*' to file	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
Unable to connect device '*' to the ECU	No connection can be established between the flash device and the ECU.	Check all physical connections between the INCA-PC and the ECU. Check the power supply. Make sure the protocol used is supported by the ECU.
Version check of ODX Project failed	The ODX project to be used was created with a more recent INCA version.	Create the ODX project in the INCA version to be used to program the ECU.
Can't set project path to '*'	An error occurred when specifying the project path in the MVCI Server.	Shut down INCA, delete the contents of ETAS temp and reboot INCA. It may be necessary to reboot your PC.
Some referenced files or directory do not exist	The ODX project cannot be imported on the basis of the selected file(s).	Check the ODX project to be imported as well as the referenced components.
Some of the referenced files have an unsupported file extension		
Flash Job returned with error	Executing the flash job returned an error.	Check the flash user interface for further error information.
The settings are incomplete	The flash job could not be executed as not all necessary settings had been made in the ODX-FLASH user interface.	Check the settings in the ODX-FLASH user interface and update/complete these if necessary.

Error Message	Error Description	Remedy
The ODX project is inconsistent. It contains no project file (*.prj)	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
	The database object for the ODX project is damaged.	Generate a new database object using the ODX files.
Decompressing failed	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
Can't create temporary directory '*'.	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
Can't remove temporary directory	INCA cannot remove a subdirectory from the ETAS temp temporary directory.	
	INCA is still accessing this file.	Shut down INCA, delete the contents of ETAS temp and reboot INCA. It may be necessary to reboot your PC.
	Another process is still accessing this file.	End all other applications which could be accessing the ETAS temp directory. It may be necessary to reboot your PC.
	You do not have the necessary user rights.	Contact your system administrator.

Error Message	Error Description	Remedy
The ODX project is inconsistent. It contains no file '*'	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
	The database object for the ODX project is not valid.	Generate a new database object using the ODX files.
The ODX project is inconsistent. File '*' is not readable	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
	The database object for the ODX project is not valid.	Generate a new database object using the ODX files.
Extended addressing for tester and normal addressing for ECU is not supported	The "CP_CanRespUSDtFormat" parameter requires "extended addressing" for the tester and "normal addressing" for the ECU. INCA does not support this combination.	
	The "CP_CanPhysReqFormat" parameter requires "extended addressing" for the tester and "normal addressing" for the ECU. INCA does not support this combination.	

### 6.3 Errors When Adding an ODX Project to the Database

Error Message	Error Description	Remedy
Unable to create ODX project. The MVCI Server is in use.	The applications ODX-FLASH and ODX-LINK, and the adding of an ODX project to the database cannot take place simultaneously.	End ODX-FLASH or ODX-LINK respectively.
ODX file import failed due to one of the following reasons: some ODX files or files referenced by the selected ODX files (e.g. java code) are missing	An error was reported by the MVCI Server during the conversion or verification of the ODX project.	Check the reference files of the project to be imported.
ODX file import failed due to one of the following reasons: the ODX files are not ODX V2.0.1 compliant		Create the ODX project and the relevant ODX files in accordance with the "ODX V2.0.1" specification.
ODX file import failed due to one of the following reasons: the ODX files are inconsistent and violate ASAM ODX checker rules.		Check whether the ODX files conform to the ASAM-ODX rules.
Reinstall ODX AddOn Installation.	Necessary directories or files of the ODX Add-on installation are missing.	Install ODX Add-on again.
Can't save ODX project to file '*'	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.

Error Message	Error Description	Remedy
The ODX project is inconsistent. It contains no project file (*.prj)	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
	The database object for the ODX project is not valid.	Generate a new database object using the ODX files.
Decompressing failed	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
Can't create temporary directory '**'.	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
Can't remove temporary directory	INCA cannot remove a file from the ETAS temp temporary directory.	
	INCA is still accessing this file.	Shut down INCA, delete the contents of ETAS temp and reboot INCA.
	Another process is still accessing this file.	End all other applications which could be accessing the ETAS temp directory. It may be necessary to reboot your PC.
	You do not have the necessary user rights.	Contact your system administrator.

Error Message	Error Description	Remedy
Can't copy file '*' to '*'	There is too little storage space in the ETAS temp folder.	Shut down INCA, delete the contents of ETAS temp and reboot INCA. It may be necessary to reboot your PC.
	You do not have the necessary user rights.	Contact your system administrator.
Close ODX-LINK	The applications ODX-FLASH and ODX-LINK, and the adding of an ODX project to the database cannot take place simultaneously.	End ODX-FLASH or ODX-LINK respectively.
Error accessing MSCI Server	An error occurred when trying to access the MSCI Server.	Shut down INCA, delete the contents of ETAS temp and reboot INCA.
		End all other applications which could be accessing the ETAS temp directory. It may be necessary to reboot your PC.
Can't remove the file '*'	INCA cannot remove a file from the ETAS temp temporary directory.	Shut down INCA, delete the contents of ETAS temp and reboot INCA.
	INCA is still accessing this file.	
	Another process is still accessing this file.	End all other applications which could be accessing the ETAS temp directory. It may be necessary to reboot your PC.
	You do not have the necessary user rights.	Contact your system administrator.
Can't load ODX project '*'	An error occurred when adding an ODX project to the INCA database.	Shut down INCA, delete the contents of ETAS temp and reboot INCA. It may be necessary to reboot your PC.



Error Message	Error Description	Remedy
The file * is missing.	A necessary file of the ODX Add-on installation is missing.	Install ODX Add-on again.
The ODX project is inconsistent. It contains no file '**'	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
	The database object for the ODX project is not valid.	Generate a new database object using the ODX files.
'The directory * is missing..	A necessary directory of the ODX Add-on installation is missing.	Install ODX Add-on again.
The ODX project is inconsistent. File '**' is not readable	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
	The database object for the ODX project is not valid.	Generate a new database object using the ODX files.
Can't delete temporary ODX project in file '**'	INCA cannot remove a file from the ETAS temp temporary directory.	
	INCA is still accessing this file.	Shut down INCA, delete the contents of ETAS temp and reboot INCA. It may be necessary to reboot your PC.
	Another process is still accessing this file.	End all other applications which could be accessing the ETAS temp directory. It may be necessary to reboot your PC.
	You do not have the necessary user rights.	Contact your system administrator.

Error Message	Error Description	Remedy
Can't delete temporary file '*'	INCA cannot remove a file from the ETAS temp temporary directory.	
	INCA is still accessing this file.	Shut down INCA, delete the contents of ETAS temp and reboot INCA. It may be necessary to reboot your PC.
	Another process is still accessing this file.	End all other applications which could be accessing the ETAS temp directory. It may be necessary to reboot your PC.
	You do not have the necessary user rights.	Contact your system administrator.
Error in TP_BLOP generation: Sending a Tester Present Message is required for *	The ODX-COM PARAMs define a "tester present" message without parameters. KWPOncAN does not support this.	Set the value of the "CP_TesterPresentHandling" parameter to 1 or set the byte size of the "CP_TesterPresentMessage" parameter to a value > 0.

## 7 **ODX Link Menus and Functions**

---

If you have installed ODX-LINK V1.5 on your system, the **ODX** menu is displayed in the INCA menu bar of the experiment window.

You can use this menu to

- work with the GUIs to run ODX functions (see "User Views" on page 68)
- define how the snapshot data is saved (see "Data Logging Configuration" on page 117)
- configure and trigger snapshot functionality (see "Snapshots" on page 122)

This chapter also contains information on the subject "Diagnostic Signals in The INCA Variable Selection" (see section 7.4 on page 125).

## 7.1 User Views

---

Generally ODX-LINK can open all ODX projects which comply with the ODX standard in Version 2.0.1.

These projects describe the diagnostic services, with their parameters, which can be queried via the ODX-LINK user views. In addition, the ODX project contains decoding information for responses from the ECU, i.e. information about how the data from the ECU is to be interpreted.

The ODX-LINK user views output this decoded information (or parts thereof) in the windows.

With the help of several ODX-LINK user views (e.g. the "DiagnosticServices" window), the user can select and parameterize any service – once the service request has been sent, the complete, decoded response is shown in the user view. In this case, the user must have detailed knowledge of the different services and of the data transported.

Other ODX-LINK user views, on the other hand, have been developed for very special requirements, such as the display of data for ECU identification ("ECUIdentification" window) or reading the error memory ("DTC" window). In these cases, the user gets the required result at the simple push of a button.

The ODX-LINK user views do not depend on whether INCA measuring is taking place or not, i.e.:

- even if there is no current INCA measuring, the windows for querying diagnostic data can be used,
- the display of data in the ODX-LINK user views is not controlled via the starting or stopping of INCA measuring but exclusively via manual user operation (using the relevant **Read ...** buttons),
- the displayed diagnostic data of the ODX-LINK windows is not recorded in an INCA measure file. To record diagnostic data, corresponding diagnostic signals must be configured for INCA measuring in the INCA variable selection.

Use **ODX → User views** to open the dialog windows for running your diagnostic tasks. The menu contains the following items:

- **DiagnosticServices** ("Diagnostic Services" on page 72)
- **ECUIdentification** ("ECU Identification" on page 76)
- **HexService** ("Hex Service" on page 79)
- **DiagTroubleCode** ("Diagnostic Trouble Code" on page 80)
- **MemoryDump** ("Memory Dump" on page 87)
- **Sequence** ("Sequence" on page 96)
- **OBD** ("OBD" on page 101)

**Note**

The configuration of most ODX-LINK user views (apart from "DiagnosticServices" and "HexServices") must correspond to the ODX diagnostic database read in.

ODX-LINK makes ODX configurations (ODX databases and the corresponding default window configurations) available to all available standard diagnostic protocols (KWP2000, Diagnostics on CAN, UDS and OBDonCAN) in the form of INCA export files (to be found in `ETASData\INCA7.1\export\ODX`). Please note, however, that these ODX configurations may not actually suit all diagnostic services of the ECU used, as the diagnostic services of many ECUs deviate from the relevant standard diagnostic protocol. ECU-specific fault memory entries and environment data in particular are thus not part of these sample configurations.

**The "OBD" User View**

Unlike other diagnostic protocols, the OBD standard defines which services there are, how they are to be parameterized and which responses they supply.

This is why it is possible in the application case "OBD" to use a specific ODX project and preconfigure the "OBD" user view entirely for this project.

When the "OBD" user view is opened, a check is made to see whether all the necessary services and parameters etc. exist in the ODX project currently being used by ODX-LINK. If this is not the case, an error message is issued. In this case, the "OBD" user view is opened but cannot be used with the current database.

**Note**

The "OBD" User View can only be used with the ETAS OBD-ODX database `ETASData\ODX1_5_0\Projects\OBDonCAN_ETAS`. The database is also contained in the INCA export file: `ETASData\INCA7.1\export\ODX\OBDonCan.exp` provided.

**Default Configurations**

The configuration of a user view can be saved as a default setting by selecting the **Save as default** button in the relevant window.

When you have made a configuration for the first time, you are prompted to specify whether you want to save these changes as the default setting when you close the user view. These settings are saved as part of the ODX configuration and are valid for all GUIs of one type.

This guarantees that ODX project-specific settings in the relevant GUIs do not have to be started from scratch every time.

**Snapshots**

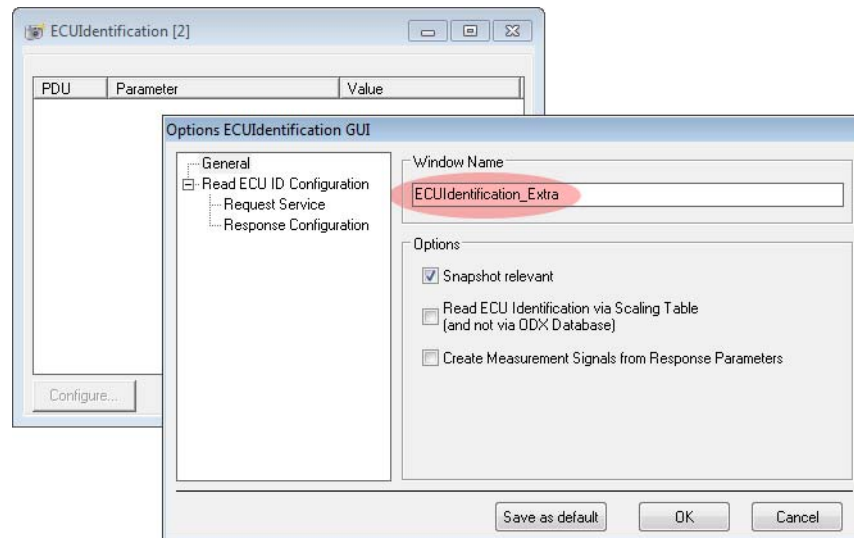
All current results of the service requests which you define with user views, can be saved via a Snapshot function. You decide for each individual user view whether its result should be included in the snapshot file. The relevant information can be found in the configuration settings of the individual user views.

The snapshot icon – a small camera – at the very left of the title bar of each user view window shows that the results of this user view are recorded in the snapshot. For more details on the snapshot function, refer to the section "Snapshots" on page 122.

#### *Names of User Views*

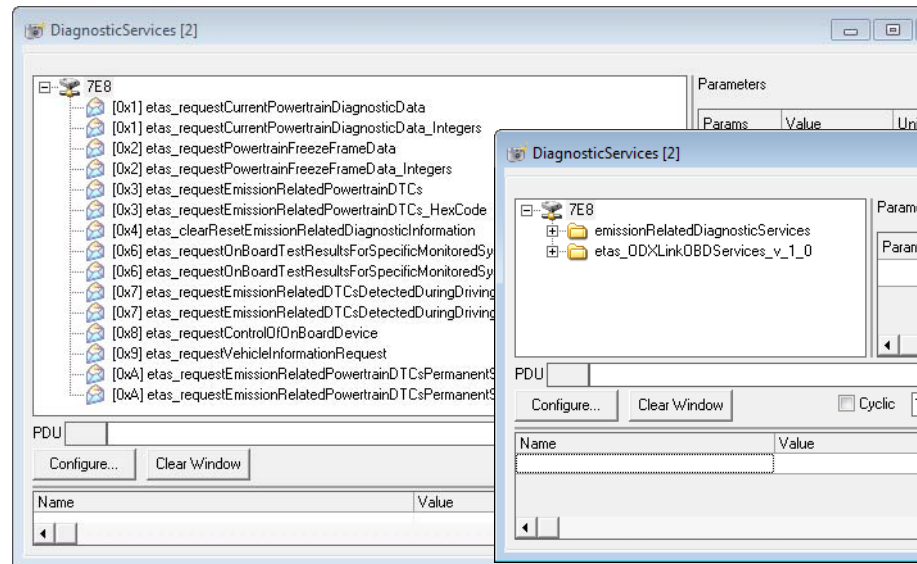
If several user views of the same type but with different configurations are used in an experiment, they can be given different names (with the exception of "Hex-Service").

To do this, select **Configure** and enter the desired name in the "Window Name" box.



### Showing the Services in Function Classes

The services can either be displayed as a list (see left of the figure) or in functional classes (shown by a folder) (see right of the figure) in a user view window (e.g. DiagnosticServices):



### To configure service display

- Select **Options** → **Users Options** → **Open** from the INCA main menu.  
The window for setting user options opens.
- Select the "ODX" tab.
- Select "Yes" or "No" with the option "Show Functional Classes".

A service is assigned to a functional group by the author of the ODX project data.

### 7.1.1 Diagnostic Services

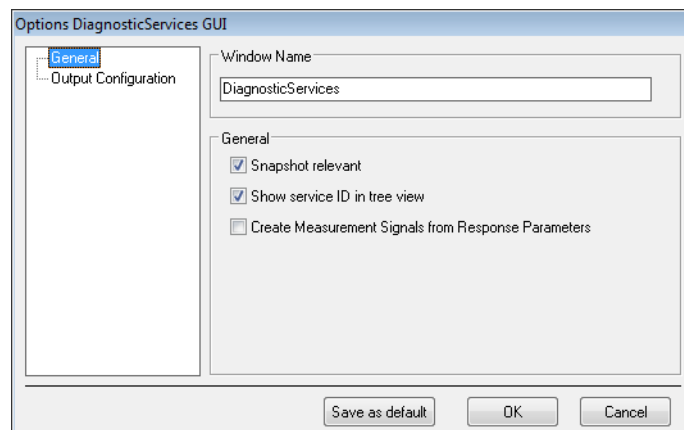
The **Diagnostic Services** function allows you to send a service request to the ECU. The service request must be defined in the diagnostic database. The service request and its result can be shown both in clear text and in hexadecimal notation. All clear text must be defined in the diagnostic database. The display of the service request and the results can be configured.

The request can be executed either just once or recur periodically. The recurrence rate can be configured.

If the cyclical send timeframe you defined cannot be adhered to, for example because the bandwidth of the interface is not sufficient, it is adjusted automatically. In this case, the adjusted cycle time is shown with a red background.

#### To configure a diagnostic service

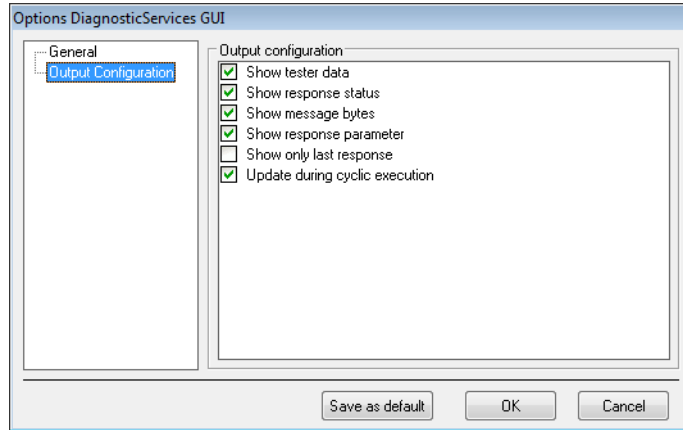
- Select **ODX** → **User views** → **DiagnosticServices**.  
The dialog box for selecting a service is displayed.
- Click **Configure**.
- Select "General" in the left-hand window.



- Activate the check box "Snapshot relevant" if the results of the service request are to be recorded in the snapshot (see the section "Snapshots" on page 122).
- Activate the check box "Show service ID in tree view" if you want the ID of the Diagnostic Service to be displayed next to the name.
- Activate the option "Create Measurement Signals from Response Parameters" if you want to add the signal measured subsequently to the signals available in the INCA variable selection (see "Diagnostic Signals in the INCA Variable Selection" on page 125).
- Pressing the **Save as default** button makes any changes you have made become the default setting.



- Select “Output Configuration” in the left-hand window.



- Activate the required options.
  - Click **OK**.
- Refer to the following table for details of the significance of the options.

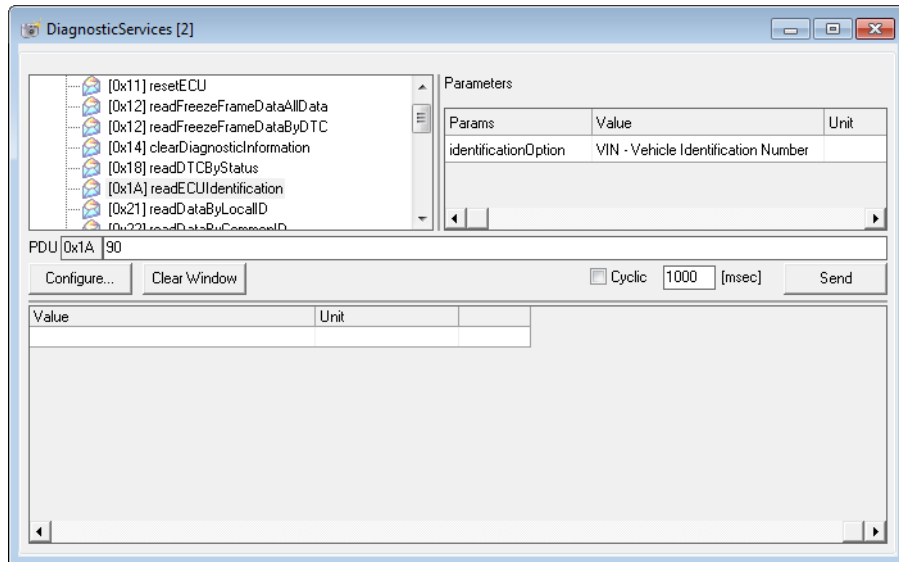
Option	Meaning
Show tester data	Service request and relevant parameters which were sent to the ECU
Show response status	Type of response of the ECU, response status
Show message bytes	Response of the ECU in hexadecimal notation
Show response parameter	ASAM MCD2D-interpreted response of the ECU
Show only last response	Shows only the data of the last service request. The data of previous service requests is deleted.
Update during cyclic execution	If the cyclic repetition of the service request is activated, the display is updated in every cycle.

**Tab. 7-1** Options for Configuring the Service Request Display

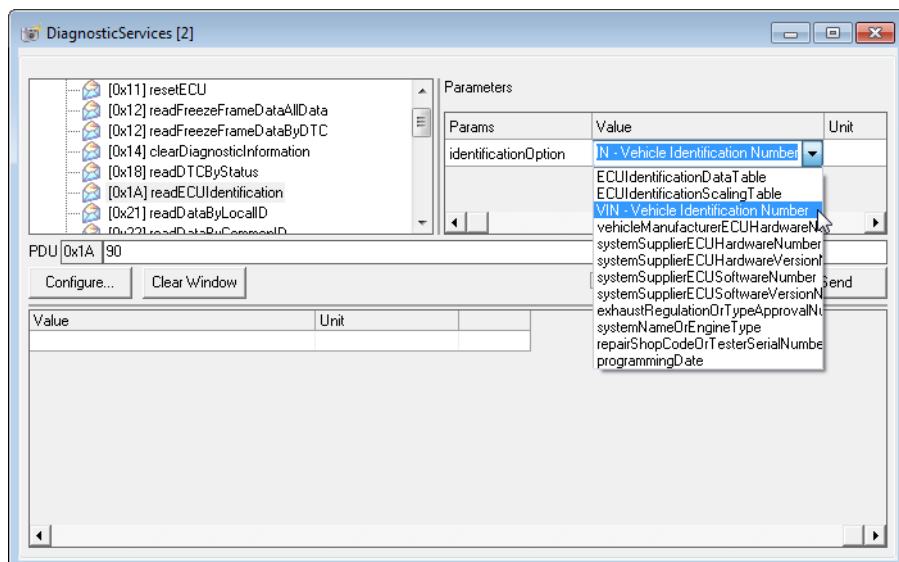
**To execute a diagnostic service**

- Choose **ODX → User views → DiagnosticServices**.

The dialog window for selecting the service will be displayed.



- Select the service you want to execute from the top left section of the window.  
The top right section of the window will show the parameters for the service you selected.
- In the “Values” column, select the parameter values for the service.



- If the service is to be repeated on a regular basis, select the “Cyclic” check box and enter the cycle intervals.

- Click **Send**.  
The service request is sent to the control unit and the response from the ECU will show up in the bottom section of the window.

The data (PDU, Protocol Data Units) sent to the ECU after the service ID has been entered, can be modified manually. To do this, use hexadecimal notation to enter the data in the "PDU" field. Please note that afterward, you can no longer select the parameters for this service.

#### *Java Jobs*

---

The ODX data model makes it possible to run Java code. These Java jobs are handled like diagnostic services - in particular, intermediate results can be issued while they are run.

Java jobs are indicated with a Java icon in the list of services - when selected in the list, both job parameters and diagnostic service parameters are shown on the right in the window and can also be edited there.

Java jobs are started using the **Send** button - depending on how complex they are these may take several seconds or even minutes to run.

Although not permissible in the ODX data model, working with Java GUIs is technically possible (see the example "JobDemo\_JavaGUIs").

#### **Note**

*The first time they are run, the Java windows may remain in the background!  
The windows are only in the foreground when they are run again.*

### 7.1.2 ECU Identification

This function allows you read the whole ECU identification. The result for the service request will be displayed in clear text. The service requests and all clear text must be defined in the diagnostic database.

This function also allows you to configure the service request and its parameters, as well as the display of the results.

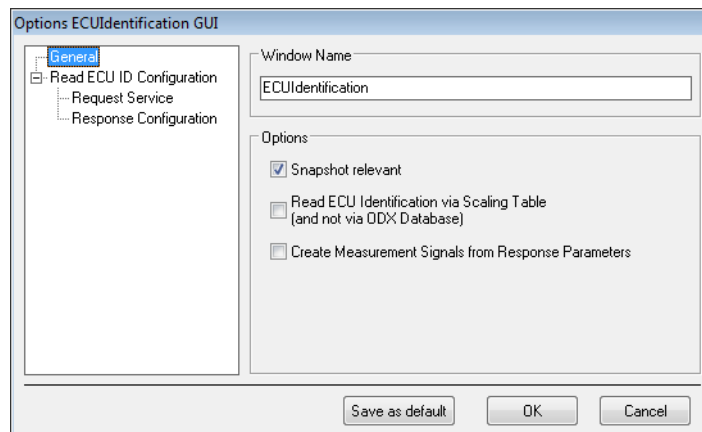
Alternatively, the ECU identification and the Scaling Table can be determined. Then no entries are required in the diagnostic database.

The window can also be used to generate diagnostic signals for the INCA variable selection (see "Diagnostic Signals in the INCA Variable Selection" on page 125).

#### To configure the ECU Identification

- Select **ODX** → **User views** → **ECUIdentification**.
- Click **Configure**.

The service request configuration dialog box is displayed.

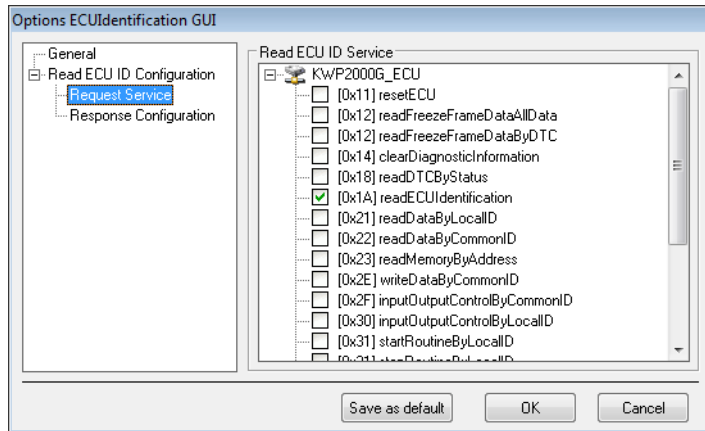


- Click "General" in the left-hand window.
- Activate the "Snapshot relevant" check box if the result of this service request is to be recorded in the snapshot (see the section "Snapshots" on page 122).
- Activate the "Read ECU Identification via Scaling Table" check box if the ECU is to be identified via the Scaling Table.
- Activate the option "Create Measurement Signals from Response Parameters" if you want to add the signal measured subsequently to the diagnostic signal list (see "Diagnostic Signals in the INCA Variable Selection" on page 125).

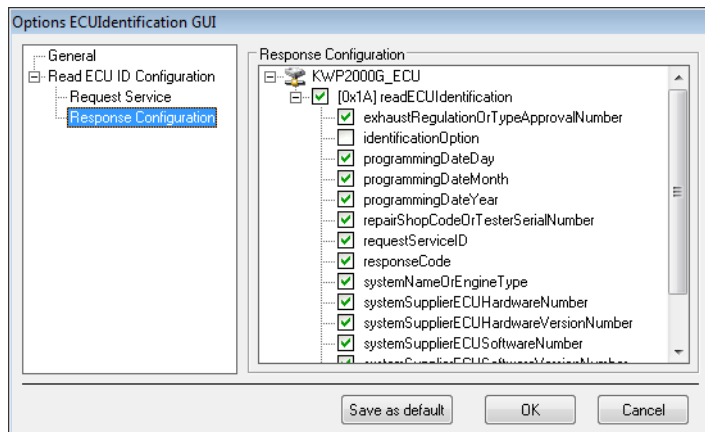
The configuration is complete.

or

- Click "Request Service" in the left-hand window.



- Select the service you want to assign to this function.
- Click "Response Configuration" in the left-hand window.

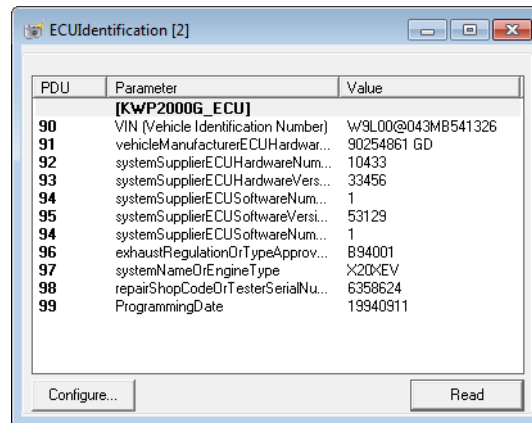


- Select those response parameters which are to be recorded in the display of results of the service request.
- Click **OK**.

### To execute an ECU identification

- Select **ODX** → **User views** → **ECUIdentification**.
- Click **Read**.

The service is executed for all service parameters defined in ODX and the ECU responses will be displayed.



PDU	Parameter	Value
	<b>[KWP2000G_ECU]</b>	
90	VIN (Vehicle Identification Number)	W9L00@043MB541326
91	vehicleManufacturerECUHardwar...	90254861 GD
92	systemSupplierECUHardwareNum...	10433
93	systemSupplierECUHardwareVers...	33456
94	systemSupplierECUSoftwareNum...	1
95	systemSupplierECUSoftwareVersi...	53129
94	systemSupplierECUSoftwareNum...	1
96	exhaustRegulationOrTypeApprov...	B94001
97	systemNameOfEngineType	X20XEV
98	repairShopCodeOrTesterSerialNu...	6358624
99	ProgrammingDate	19940911

- Click **Read** to execute the service again.

#### **Note**

After installation, this function is assigned the "[1A] readECUIdentification" service from the KWP2000 protocol (if this service is contained in the diagnostic database). However, you can assign any ODX service to this function.

### 7.1.3 Hex Service

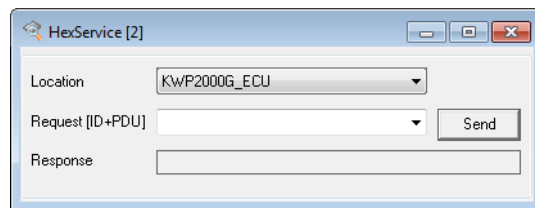
The **Hex Service** view allows you to send any data to the ECU through one of the defined interfaces. You select the diagnostic database for each ECU defined.

The response from the ECU is shown in hexadecimal notation.

The data you enter is stored in a history. You can use data entered within a session as many times as required.

#### To execute a hex service

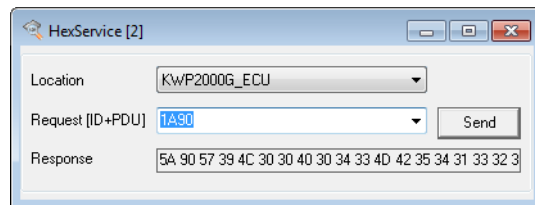
- Choose **ODX** → **User views** → **HexService**.  
The dialog window for specifying the request will be displayed.



- Select a logical link in the "Location" field.
- In the "Request" field, enter the data (service ID and PDU) for the request that is to be sent to the ECU in hexadecimal notation.

or

- Select an existing entry from the selection list of the "Request" field.
- Click **Send**.



The response from the ECU will be displayed in the "Response" field in hexadecimal notation.

#### 7.1.4 Diagnostic Trouble Code

The **DiagTroubleCode** view allows you to execute the following operations:

- Read the fault memory
- Read the environment data on a trouble-log entry
- Clear the fault memory
- Configure the diagnostic services for reading and clearing the fault memory, and the display of the trouble-log entries
- Configure the diagnostic services for reading the environment data on a trouble-log entry.

After reading the fault memory and the environment data, the contents are shown in clear text. The ECU-specific ODX database will map the trouble-log entries to the corresponding clear text.

To clear the fault memory, a configurable service identification is sent to the ECU. The response from the ECU will be displayed.

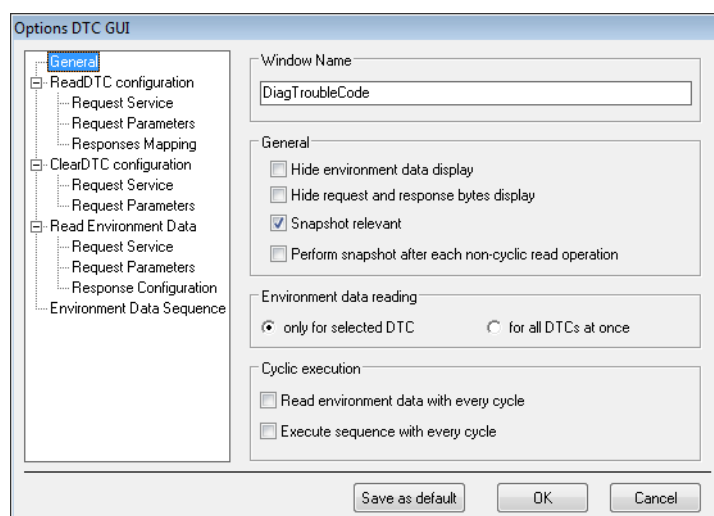
You can configure how the fault memory data and the environment data is displayed. After installation, the default values for the KWP2000 protocol are set.

You can determine whether the ECU data read is to be recorded in the snapshot. For more details on the snapshot function, refer to the section "Snapshots" on page 122.

#### General configuration

The response of the Diagnostic Trouble Code function can be configured. This is where you determine how the results of the service requests are displayed, whether the results are recorded in a snapshot and how the environment data is read.

- Select **ODX** → **User views** → **DiagTroubleCode**.
- Click **Configure**.  
The configuration dialog box is displayed.
- Click **General** in the left-hand window.





- Activate the required options. The following table explains the significance of the individual options. Click **OK**.

Option	Meaning
Hide environment data display	The display of the environment data on the trouble-log entry is suppressed.
Hide request and response bytes display	The display of the service request and the response of the ECU in hexadecimal form is suppressed.
Snapshot relevant	The results of the service request are recorded in the snapshot. For more details on the snapshot function, refer to the section "Snapshots" on page 122.
Perform snapshot after each non-cyclic read operation	Each time a DTC is read, a snapshot is taken. This is not the case when the "Cyclic" option is active.
Environment data recording only for selected DTC	The environment data is only read for the selected trouble-log entry.
Environment data recording for all DTCs at once	The environment data is read for all trouble-log entries.
Read environment data with every cycle	In the cyclical querying of trouble-log entries, all relevant environment data is automatically read.
Execute sequence with every cycle	The sequence specified under "Environment Data Sequence" is executed every cycle.

#### **Note**

After installation, the following diagnostic services are assigned to this function:

Read fault memory: "ReadDTCByStatus"

Clear fault memory: "ClearDiagnosticInformation"

Read environment data: "ReadFreezeFrameDataByDTC"

If these diagnostic services are not contained in the diagnostic database you use or you would like to use other services, you have to adapt the configuration accordingly.

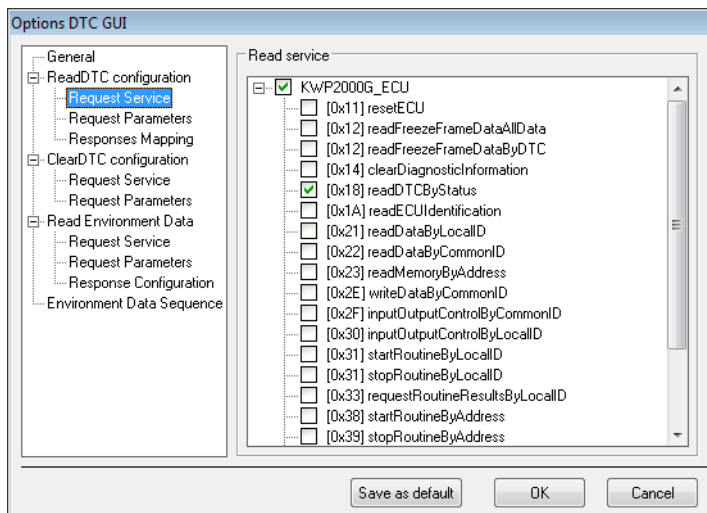
The relevant diagnostic service from the diagnostic database has to be configured for reading and clearing the fault memory as well as reading the environment data.

#### **To configure the diagnostic service**

You have to specify the relevant service identification for each of the operations.

- Select **ODX** → **User views** → **DiagTroubleCode**.
- Click **Configure**.  
The configuration dialog box is displayed. The operations to be configured are displayed in the left-hand window.
- Open the operation you want to configure.

- Select “Request Service” for the operation you want to configure.
- Highlight the diagnostic service in the “Services” group you want to assign to the operation.



- Repeat these steps for all operations you want to use.
- Click **OK**.

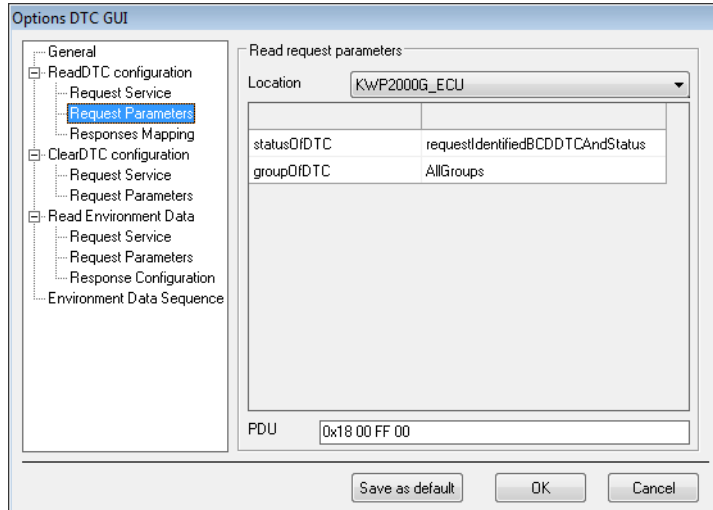
### To configure the parameters of the diagnostic service

You have to specify the parameters for the service identification for each of the operations.

- Select **ODX** → **User views** → **DiagTroubleCode**.
- Click **Configure**.

The operations to be configured are displayed in the left-hand window.

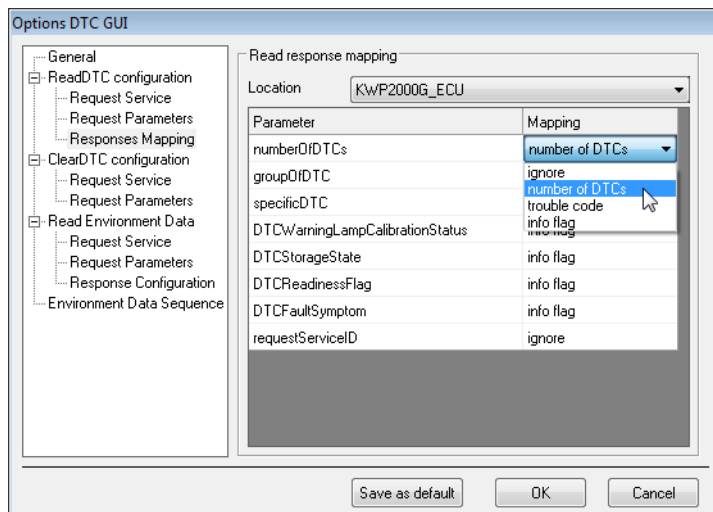
- Select “Request Parameters”
- In the right-hand window, a table of assignments between the parameters of the service identification and their assigned values is displayed.



- In the selection list, in the “Value” column, select the value you want to assign the parameter of the relevant row. Assign each parameter a value.
- Click **OK**.

**To configure the display for the trouble-log entries**

- Select **ODX → User views → DiagTroubleCode**.
- Click **Configure**.
- The configuration dialog box is displayed.
- Select “Responses Mapping”.
- In the “Mapping” column, select the display mapping for each parameter.

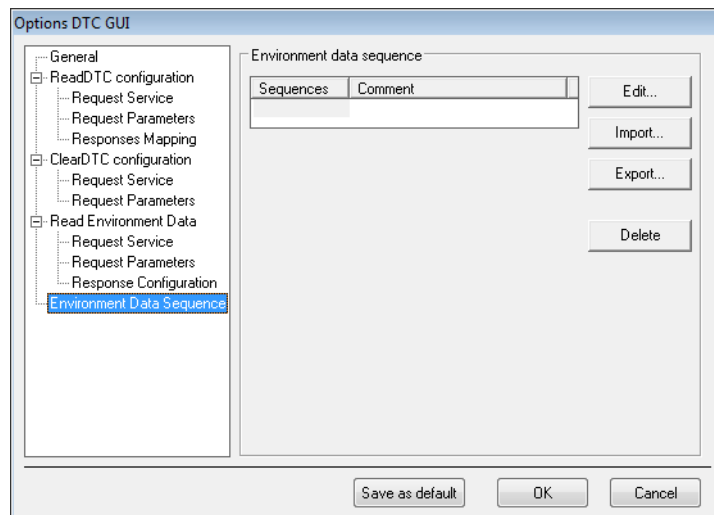


- Click **OK**.

### To define a sequence for reading environment data

You can define an additional sequence to be run when running DTCs.

- Select "Environment Data Sequence" from Options DTC GUI.

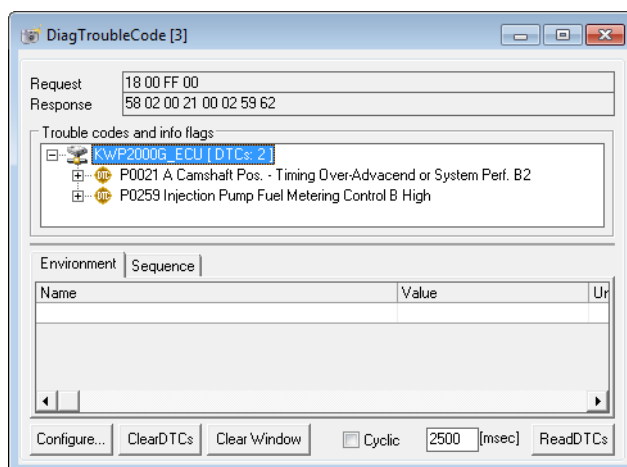


- Click **Import...** and select a sequence created previously.
- Click **Edit...** to edit this sequence if necessary.

For more details on how to work with sequences and the Sequence Editor, refer to the section "Sequence" on page 96.

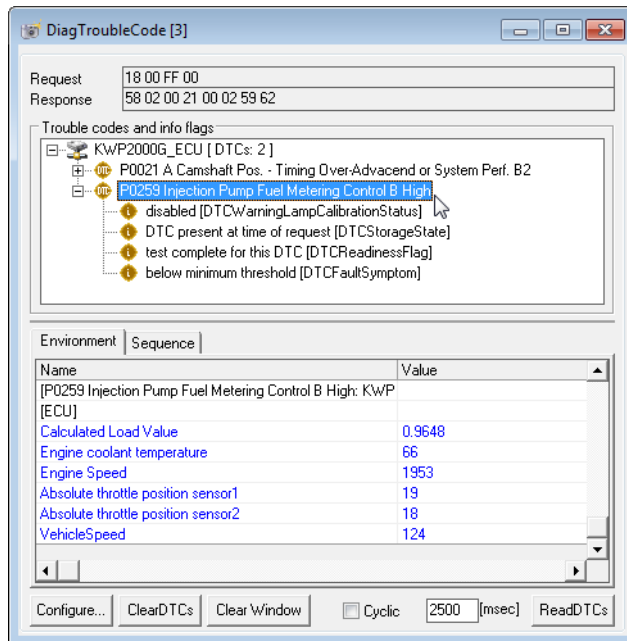
### To read the fault memory and the environment data

- Select **ODX → User views → DiagTroubleCode**.
- Click **ReadDTCs**.



The contents of the fault memory will be displayed.

- Click the trouble-log entry whose environment data you want to display. Cyclical reading does not allow for toggling between the trouble-log entry and environment data.



#### Note

*If an error message is displayed, the service request for reading the environment data has possibly been defined incorrectly.*

The environment data is shown at the bottom of the window in accordance with the option set.

If a sequence for reading the environment data has been defined, it is also run when a fault memory entry is clicked and the data is displayed in the lower part of the window in the "Sequence" tab.

- Click **Clear Window** to clear the content of the lower part of the window.

#### To read the fault memory periodically

- Select **ODX → User views → DiagTroubleCode**.
- Select the "Cyclic" check box and enter the cycle interval.
- Click **ReadDTCs**.
- The contents of the fault memory will be periodically read and displayed.
- Click **STOP** to stop reading the fault memory on a regular basis.

### To clear the fault memory

---

- Click **ClearDTCs**.

The service for clearing the fault memory will be sent to the ECU. A positive or negative response from the ECU will be displayed. A positive response indicates that the fault memory has been cleared.

### 7.1.5 Memory Dump

---

The **Memory Dump** function is used to read the contents of memory areas from the ECU memory. The prerequisite for this is that the diagnostic interface of the ECU supports this functionality.

You define up to five memory areas which are then read by clicking the relevant button. The memory areas can be defined both as hexadecimal addresses and ASAP labels. You can also define unconnected address spaces as a memory area.

The values are issued as hexadecimal numbers or - if the conversion functions are defined - as physical quantities.

The control elements of the **Memory Dump** function are spread across two dialog boxes. The first one displays the memory contents read. In the second one, you define the memory areas and the service to be used to read the memory areas.

Before you can use the **Memory Dump** function, you have to define a few settings. Execute the following steps:

- Change the general settings
- Define the service request for querying the memory area
- Define the memory area
- Read the defined memory area

Alternatively, you can export the definition for the memory areas for later use and reimport definitions which were exported previously.

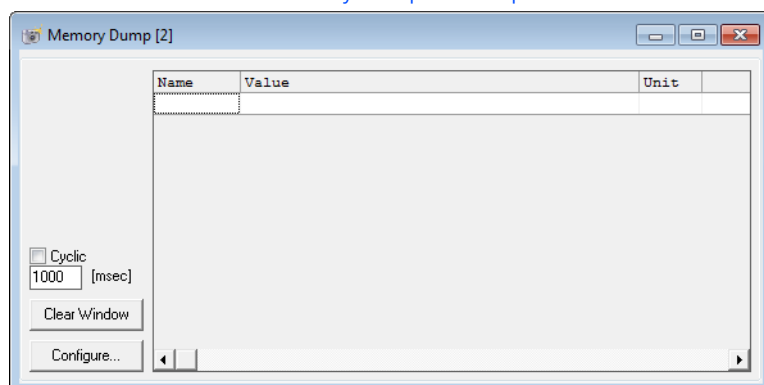
#### To change the general settings

---

In the general settings you can change the window title and the display of the output field. You can also determine whether the results of the memory area query should be recorded in a snapshot (see the section "Snapshots" on page 122).

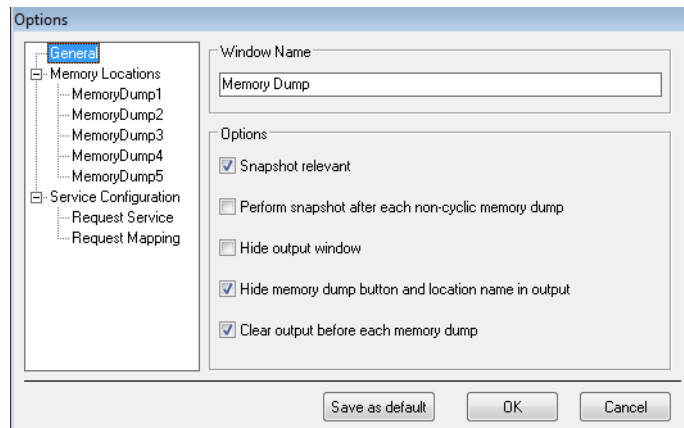
- Select **ODX** → **User Views** → **MemoryDump**.

The "MemoryDump" GUI opens.



- Click **Configure**.

- Select “General” in the left-hand window.



- Enter the window title you require in the “Window Name” field.
- Activate the “Snapshot relevant” check box if the results of the memory area query are to be recorded in the snapshots.
- Activate the “Perform snapshot after each non-cyclic memory dump” check box if you want to perform a snapshot after every query.
- Activate the “Hide output window” check box if you do not want the result list (the right-hand side of the “Memory Dump” window) to be displayed.
- Activate the “Hide memory dump button and location name in output” check box if you want to suppress the display of the button for reading the memory area.
- Activate the “Clear output before each memory dump” check box if the display of read data is to be cleared before every cyclic reading.

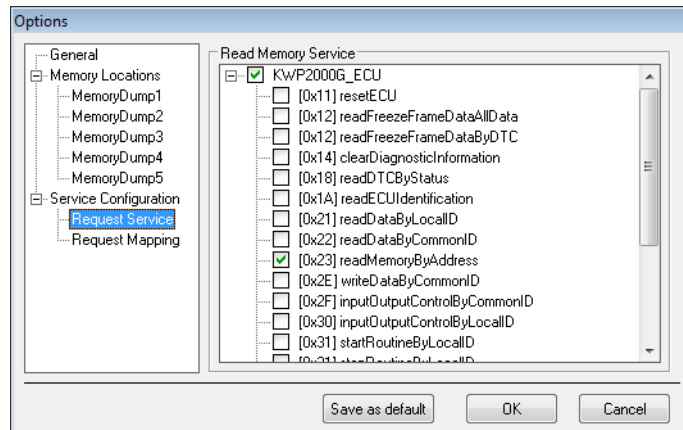
After installation, the service request “ReadMemoryByAddress” is assigned to the query of a memory area in the ECU. If you would like to use a different service request for this function or this service request is not available in the diagnostic database you are using, you have to define the service request for querying the memory area.



### To define the service request for querying the memory area

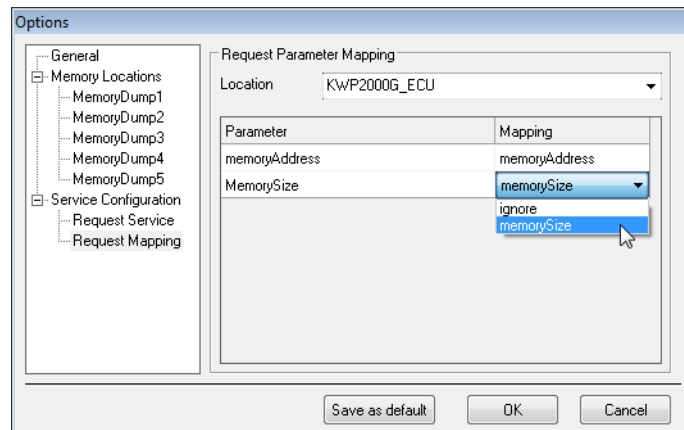
Before you can query the contents of memory areas via the diagnostic interface, you have to define which service request should be used for this purpose.

- Select **ODX** → **User Views** → **Memory Dump**.
- Click **Configure**.
- Select "Request Service".



- Select the service with which a memory area is to be read from the ECU in the "Read Memory Service" field.
- Select "Request Mapping" in the left-hand window.

- In the “Mapping” column, select the parameters of the service which correspond to the parameters “memoryAddress” and “MemorySize” (i.e., the service parameters of the previously selected service containing memory address and size).



#### Note

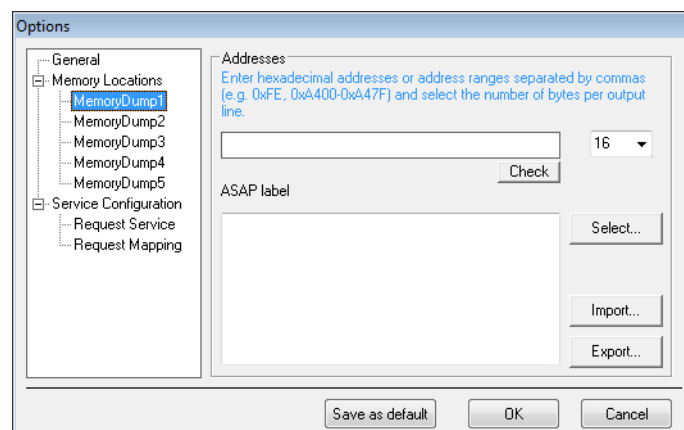
Each parameter of the service can only be assigned once. If the service parameter has already been assigned, it is no longer displayed in the parameter selection list. Rerelease the parameter by selecting “ignore” in the selection list.

- Click **OK**.

### To define memory areas

Before you can read a memory area from the ECU, the memory area has to be defined. The query button is only displayed for defined memory areas.

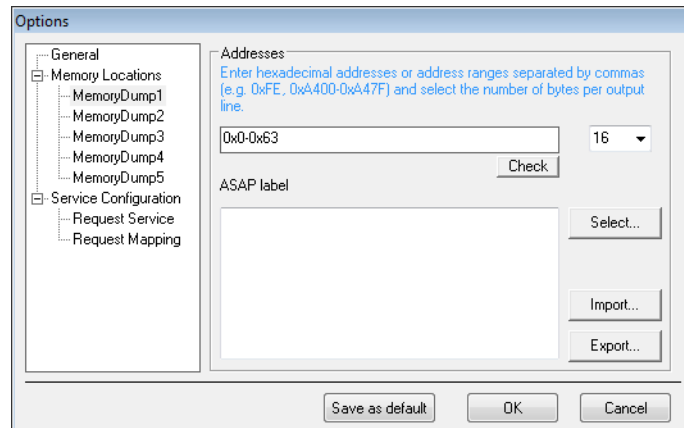
- Select **ODX** → **User Views** → **Memory Dump**.
- Click **Configure**.
- Select one of the entries under “Memory Locations”.



- Enter the address range you want to read in the “Addresses” field.

The address ranges do not have to be connected. Separate the addresses using commas. Enter the addresses as decimal numbers without a prefix or as hexadecimal numbers with the prefix “0x”.

- Use the selection field to determine how many bytes should be displayed in one output line.

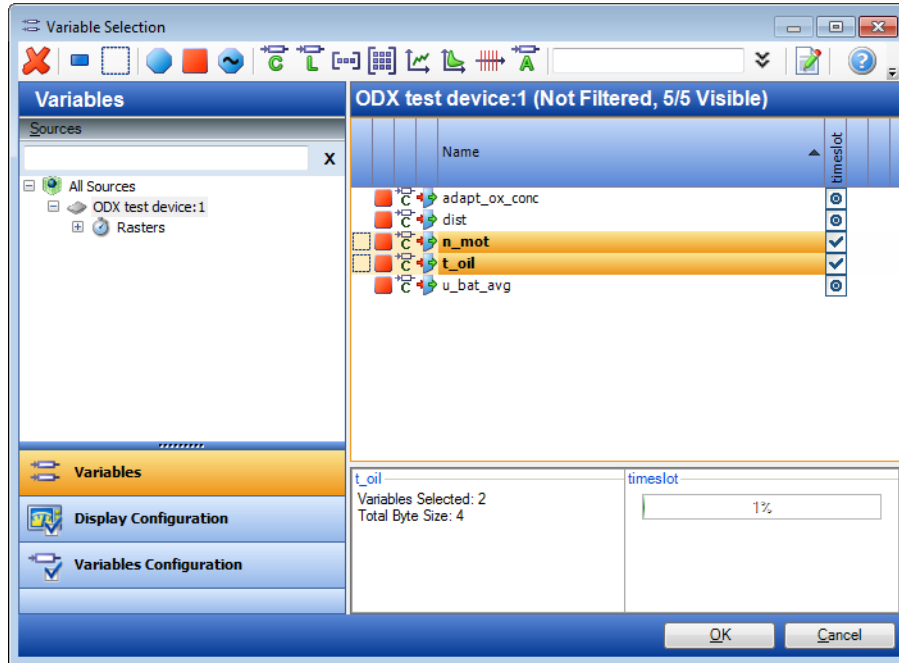


#### **Note**

*The names “MemoryDumpn” can be edited when they are double-clicked.*

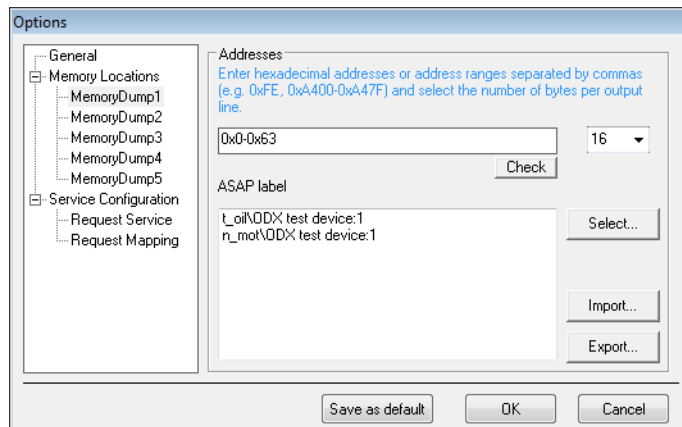
- Click **Select** to display a selection list of ASAP labels.

The "Variable Selection" window opens.



- Select the variables whose contents you would like to display and click **OK**.

The selected variables are shown in the "ASAP label" field.



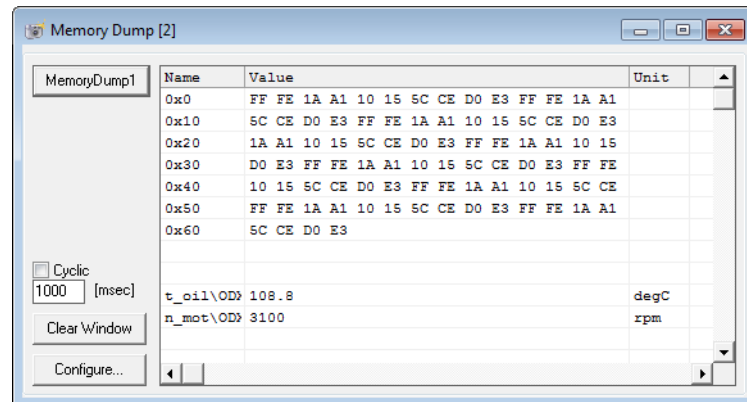
- Click **OK**.

### To read a defined memory area

Before you can read a memory area from the ECU, the memory area has to be defined. The query button is only displayed for defined memory areas. If you have not yet defined a memory area proceed as follows.

- Select **ODX → User Views → Memory Dump**.
- Click one of the **MemoryDumpn** buttons.

The contents of the memory area defined previously is displayed. If a conversion function is defined for the ASAP labels read in the A2L file used, the values are displayed as physical quantities.



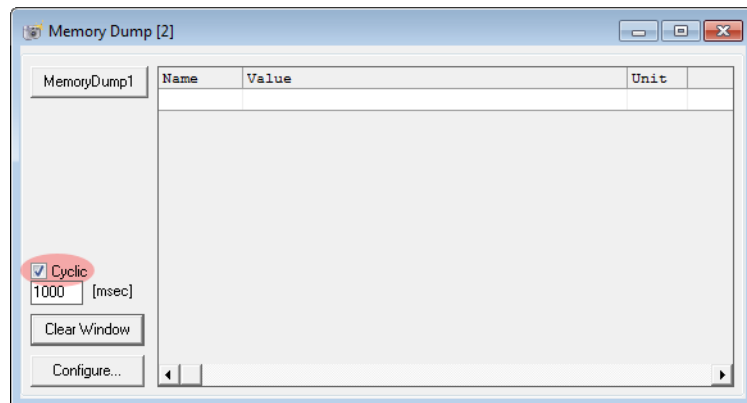
- Click **Clear Window** to clear the contents of the display.

### To repeat memory dumps periodically

If the service is to be repeated periodically, proceed as follows:

- Select **ODX → User Views → Memory Dump**.
- Activate the "Cyclic" check box.

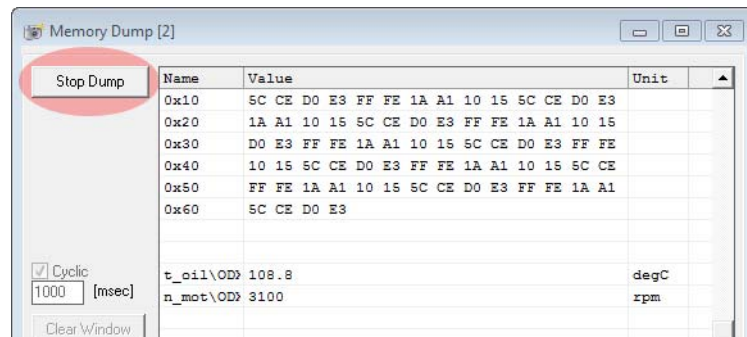
- Add the period duration in the field below (in ms).



### Note

As reading may take some time, depending on the number of ASAP labels selected, the "Cycle Time" can take up to 99 sec.

- Click **MemoryDumpn**.  
The memory dump is run cyclically in accordance with the specification you made in the period duration box.
- To stop the memory dump being run, click **Stop Dump**.



### To export settings

You can export the settings you have defined for a memory area so they can be used later.

- Click **Configure**.
- Select one of the entries under "Memory Locations".
- Click **Export**.
- Enter a file name and click **OK**.

### To import settings

---

If you have exported the settings for a memory area, you can reimport these settings.

- Click **Configure**.
- Select one of the entries under "Memory Locations".
- Click **Import**.
- Select a file name and click **OK**.

The current settings are overwritten by the settings exported previously.

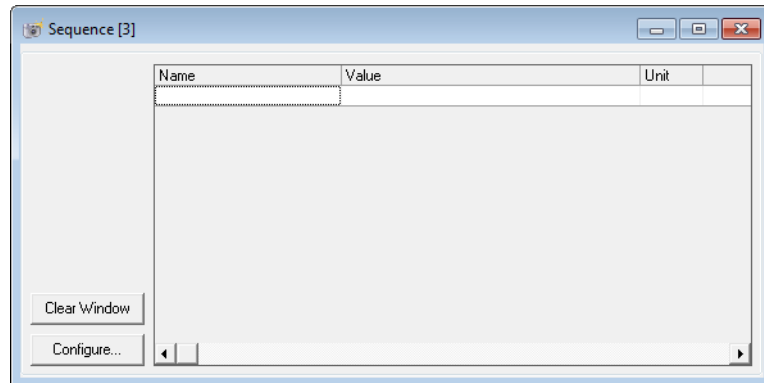
### 7.1.6 Sequence

The Sequence GUI is available for running sequences of Diagnostic Services – up to ten sequences can be defined and run.

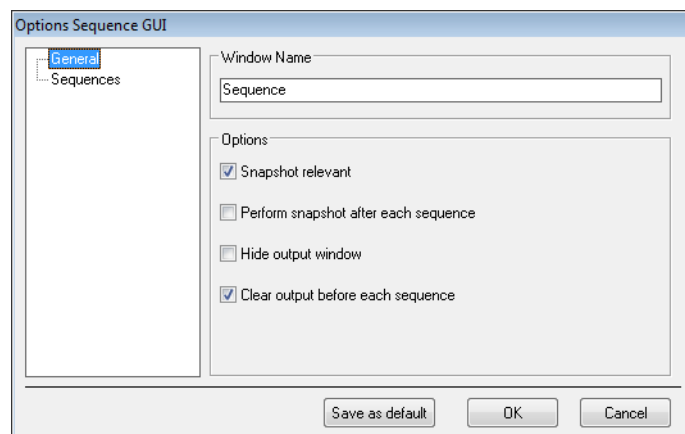
#### To make general changes

- Select **ODX** → **User views** → **Sequence**.

The “Sequence” GUI opens.



- Click **Configure...**  
The “Options Sequence GUI” window opens.
- Select “General”.



- Under “Window Name”, you can enter the title of the sequence window.
- Under “Options”, you can make the following settings:
  - Snapshot relevant  
This is used to specify whether you want to include the data in a snapshot.
  - Perform snapshot after each sequence  
A snapshot is performed after a sequence.

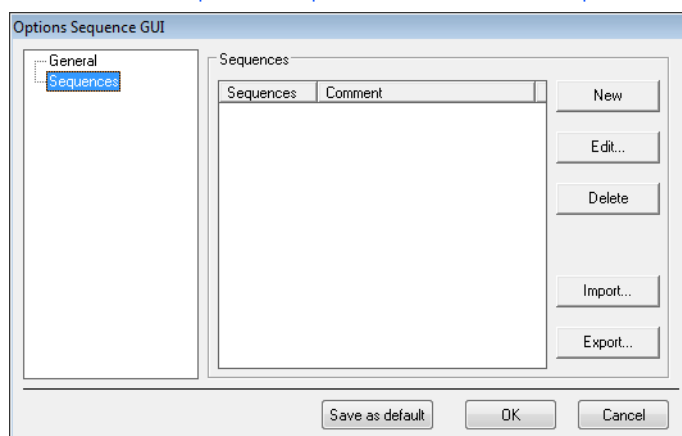


- Hide output window  
When this option is enabled, the part of the window in which data is output is hidden.
- Clear output before each sequence  
The content of the output window is cleared before a sequence is run.

### To define sequences

---

- Select "Sequences".  
The "Options Sequences GUI" window opens.



- Click **New...**  
The Sequence Editor opens.
- Create a sequence as described in the section "The Sequence Editor" on page 99.
- The sequence created there is then displayed in the list of sequences.

### To rename sequences

---

- To rename a sequence in the list, double-click it.  
The name can now be edited.
- You can also enter a comment in the second column.

### To edit sequences

---

- To edit a sequence from the list, select it and click **Edit...**  
The Sequence Editor opens.
- Edit the sequence as described in the section "The Sequence Editor" on page 99.

### To delete sequences

---

- To delete a sequence from the list, select it and click **Delete**.
- The sequence is deleted.

### To export sequences

---

You can also export individual sequences to an xml file:

- Select the sequence from the list.
- Click **Export...**  
A file selector window opens.
- Select a directory and enter a file name (with the extension .ols)
- Click **OK**.  
The sequence is saved in the file and can be reimported later.

### To import sequences

---

- Click **Import...**  
A file selector window opens.
- Select the sequence file to be imported.
- Click **OK**.  
The sequence is added to the list.

#### **Note**

*Importing a sequence also means importing its name - if a sequence of the same name already exists, the name of the sequence just imported is extended by an appended number.*

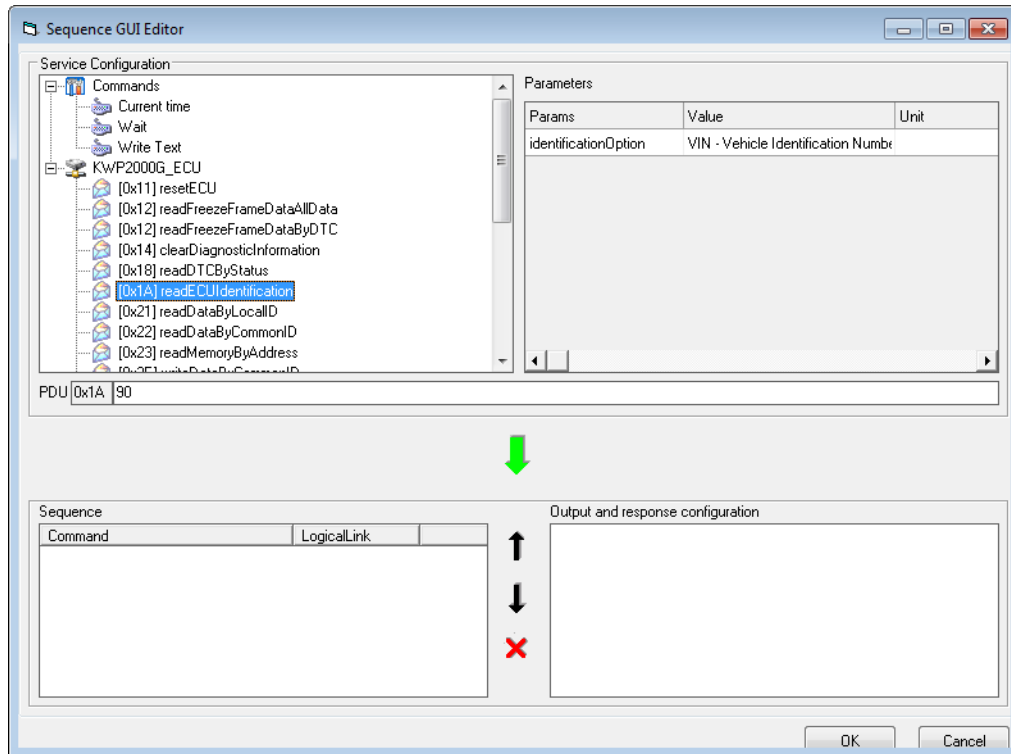
### The Sequence Editor

You use the Sequence Editor to create new sequences or edit existing ones.

#### To create a sequence in the Editor

- Click **New...** in the "Options Sequence GUI" window.

The Sequence Editor opens.



- Select the command or the request you want to add to the new sequence in the "Service Configuration" window.

#### Note

*Java jobs (see "Java Jobs" on page 75) can also be elements of a sequence.*

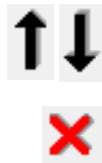
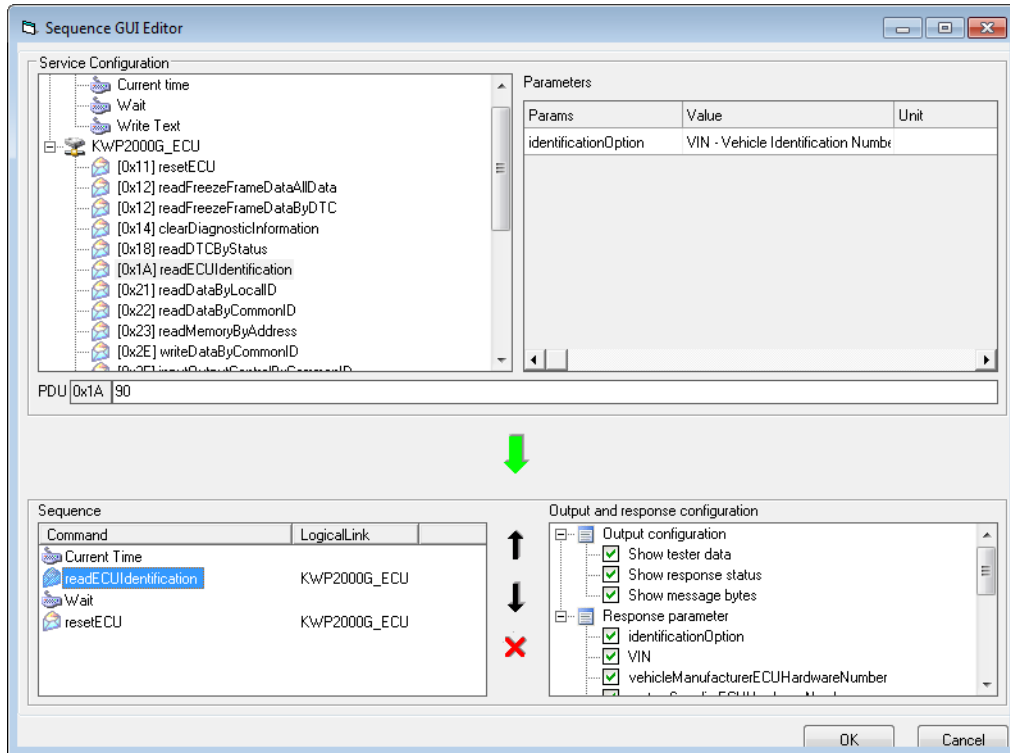


- Click the green arrow.
- The selected command or request is added to the "Sequence" list.

or

- Move the selected command or request to the "Sequence" list by Drag&Drop.
- Select the new command/request in the "Sequence" list.
- Enter the necessary parameters under "Parameters".

- Under “Output and response configuration”, you can configure the scope of the output of the response.
- Repeat the above steps until your sequence is complete.



- Use the black arrows to change the position of a command in the sequence (move up or down).
- Select the red cross to delete a command from the sequence.
- Close the Sequence Editor with **OK**.

### 7.1.7 OBD

This user view is used to query and display OBD-relevant data. Please read the section "The "OBD" User View" on page 69 on this subject.

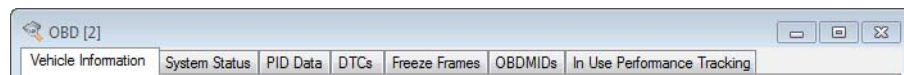
#### *Emission-Related Diagnostic Services (ISO 15031-5.4)*

The services \$01...\$09 are reserved to acquire emission-related diagnostic services:

- **Service \$01**  
Request current powertrain diagnostic data
- **Service \$02**  
Request powertrain freeze frame data
- **Service \$03**  
Show stored (emission-related) diagnostic trouble codes ("stored DTCs")
- **Service \$04**  
Clear/reset emission-related diagnostic information
- **Service \$06**  
Request on-board monitoring test results for specific monitored systems
- **Service \$07**  
Request emission-related diagnostic trouble codes detected during current or last completed driving cycle ("pending DTCs")
- **Service \$08**  
Request control of on-board system, test or component
- **Service \$09**  
Request vehicle information and information from In-Use Performance Tracking.
- **Service \$0A**  
Request permanently stored trouble codes ("permanent DTCs"). These emission-related trouble codes have a "permanent" status and cannot be deleted.

#### *Grouping into Different Tabs*

The information in the "OBD" user view is distributed to different tabs for reasons of clarity. This division does not, however, take place strictly in accordance with the functionality of the individual services, but is user-oriented.



**Fig. 7-1** The Tabs of the "OBD" User View

These are:

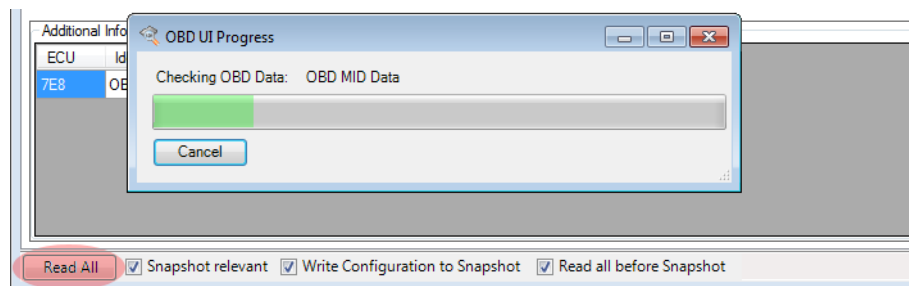
- "The "Vehicle Information" Tab" on page 104  
Information from service \$09 is contained in this tab.
- "The "System Status" Tab" on page 105  
Information from service \$01 is contained in this tab.

- "The "PID Data" Tab" on page 106  
Information from service \$01 is contained in this tab, i.e. current diagnostic data from the powertrain.
- "The "DTCs" Tab" on page 109  
Information from services \$03, \$04, \$07, and \$0A is contained in this tab.
- "The "Freeze Frames" Tab" on page 110  
Information from service \$02 is contained in this tab.
- "The "OBDMIDS" Tab" on page 113  
The OBD Monitor IDs of specially monitored systems (service \$06) are queried in this tab.
- "The "In Use Performance Tracking" Tab" on page 116  
In this tab the data of the In-Use Performance Tracking of service \$09 is displayed.

#### The **Read All** Button

If all OBD data is to be read at the same time, click **Read All** at the bottom edge of the "OBD" window. All the PIDs, OBD-MIDs, monitors, vehicle information data, DTCs etc. supported by the connected ECUs are then read automatically – changes made by hand when selecting data are taken into account.

As it can take some time to read all OBD data – depending on the quantity of supported data and the number of supported ECUs – progress is shown in a separate window.

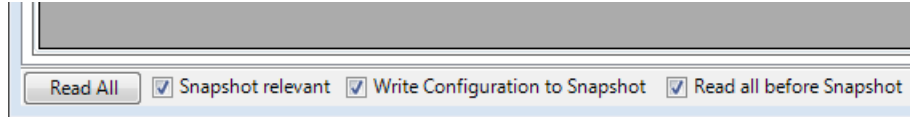


#### **Note**

*This function can also be executed automatically before a snapshot (see "Read All before Snapshot" on page 103).*

*Global Settings*

At the bottom of the OBD window, you can make two settings which are then valid globally for all tabs of the OBD user view:.



- **Snapshot relevant**  
 Activate this option if you want information from this user view to be included in the snapshot. A separate section is created in the snapshot file for every tab.
- **Write Configuration to Snapshot**  
 Activate this option if you also want the relevant configuration settings to be included in the snapshot.
- **Read All before Snapshot**  
 Activate this option if you want the **Read All** function to be executed before a snapshot. This ensures that the snapshot contains up-to-date ECU data.

*Displaying the Results*

The results of the service request are shown in the individual tabs in the form of tables.

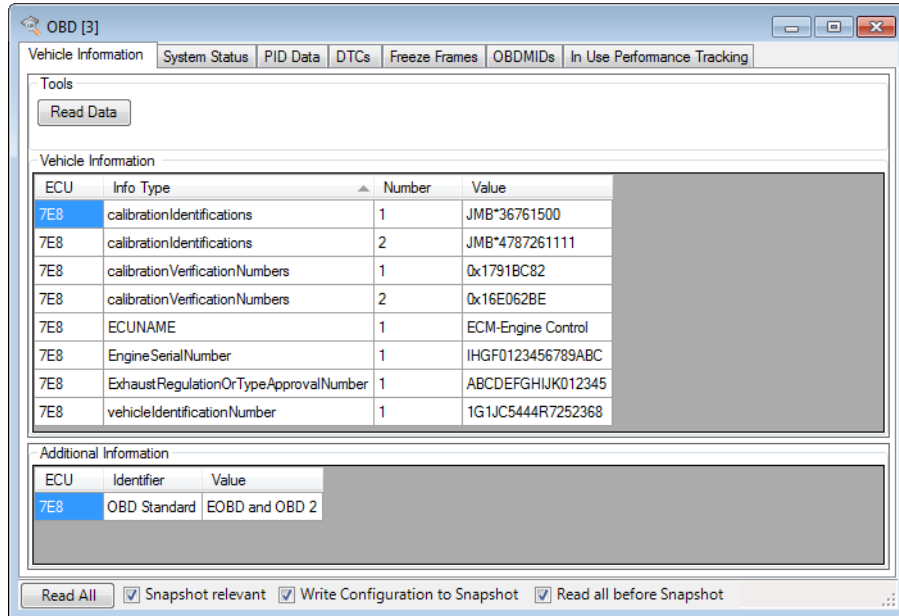
Diagnostic Trouble Codes

ECU	DTC ▲	Vehicle System	Type	DTC Name	DTC Text
7E8	0x196	Powertrain	pending	P0196	Engine Oil Temperature Sensor "A" Range/Performance
7E8	0x196	Powertrain	permanent	P0196	Engine Oil Temperature Sensor "A" Range/Performance
7E8	0x234	Powertrain	pending	P0234	Turbocharger/Supercharger "A" Overboost Condition
7E8	0x234	Powertrain	permanent	P0234	Turbocharger/Supercharger "A" Overboost Condition
7E8	0x2CD	Powertrain	permanent	P02CD	Cylinder 1 Fuel Injector Offset Learning At Max Limit
7E8	0x357	Powertrain	permanent	P0357	Ignition Coil "G" Primary Control Circuit/Open
7E8	0x4064	Chassis	stored	C0064	Roll Rate Sensor

The individual columns of this table can be moved by Drag&Drop – the lines can also be sorted in ascending or descending order of entries by clicking a column heading (in the above figure, sorting takes place in ascending order in accordance with the content of the "DTC" column).

*The “Vehicle Information” Tab*

Information from service \$09 is contained in this tab.



**Fig. 7-2** “OBd” User View – “Vehicle Information” Tab

The individual fields of the GUI contain the following functions and information:

**Tools:** This field is used to read the data – click **Read Data** for this purpose.

**ODX Database Vehicle Information:** The meaning of each individual entry in the list is described in the following table:

Column	Meaning
ECU	Name of the logical link (ECU) from hardware configuration
Info Type	InfoType for scaling and defining service \$09
Number	ID (if INFOTYPE contains several pieces of information)
Value	Physical value of INFOTYPE

**Tab. 7-2** ODX Database Vehicle Information

**Additional Information:** This list contains specific information of service \$01. The meaning of each individual entry in the list is described in the following table:

Column	Meaning
ECU	Name of the logical link (ECU) from hardware configuration
Identifier	Identifier of the information
Value	Value

**Tab. 7-3** Additional Information



The "System Status" Tab

Information from service \$01 is contained in this tab.

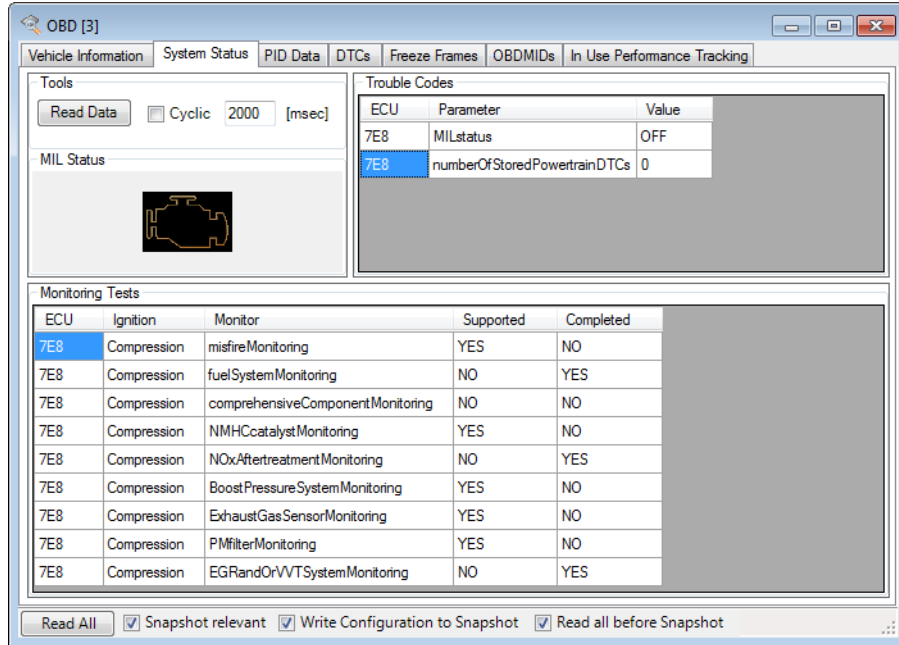


Fig. 7-3 "OBD" User View – "System Status" Tab

The individual fields of the GUI contain the following functions and information:

**Tools:** This field is used to read the data – click **Read Data** for this purpose. To read the relevant information periodically, check the "Cyclic" check box and enter the period duration (in ms).

**MIL Status:** The icon of the MIL (Malfunction Indicator Lamp) is shown in this field.

**Trouble Codes:** The meaning of each individual entry in the list is described in the following table:

Column	Meaning
ECU	Name of the logical link (ECU) from hardware configuration
Parameter	Name of the parameter
Value	Physical value of INFOTYPE

Tab. 7-4 Trouble Codes

**Monitoring Tests:** The meaning of each individual entry in the list is described in the following table:

Column	Meaning
ECU	Name of the logical link (ECU) from hardware configuration
Monitor	Monitor
Supported	Is the monitor supported?
Completed	Was the monitor ended?

**Tab. 7-5** Trouble Codes

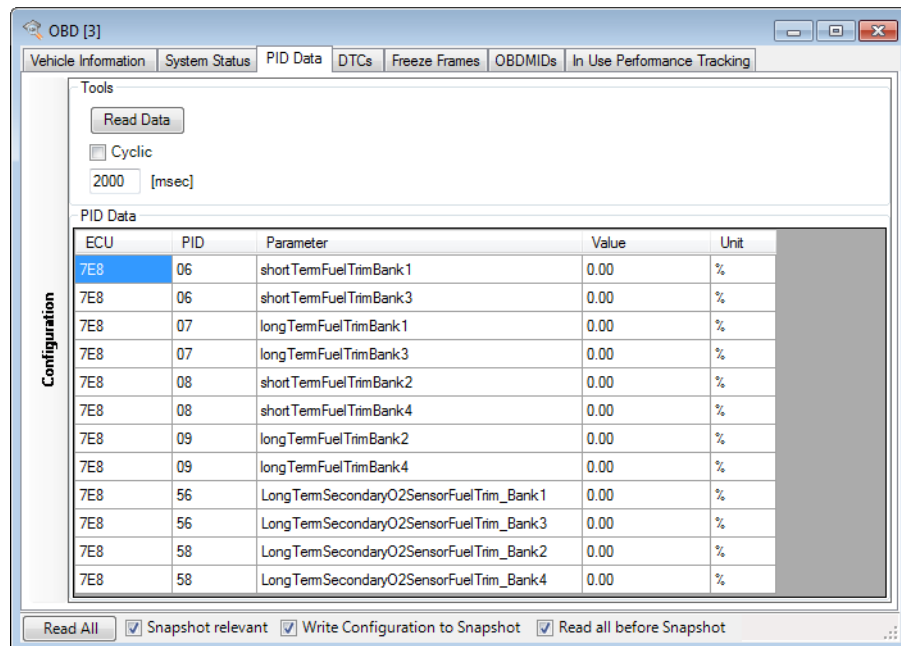
**Note**

*If an ECU identifies itself via service \$01, PID01 as a diesel ECU, the monitors relevant for diesel ECUs are displayed, otherwise the monitors for gasoline ECUs.*

*The "PID Data" Tab*

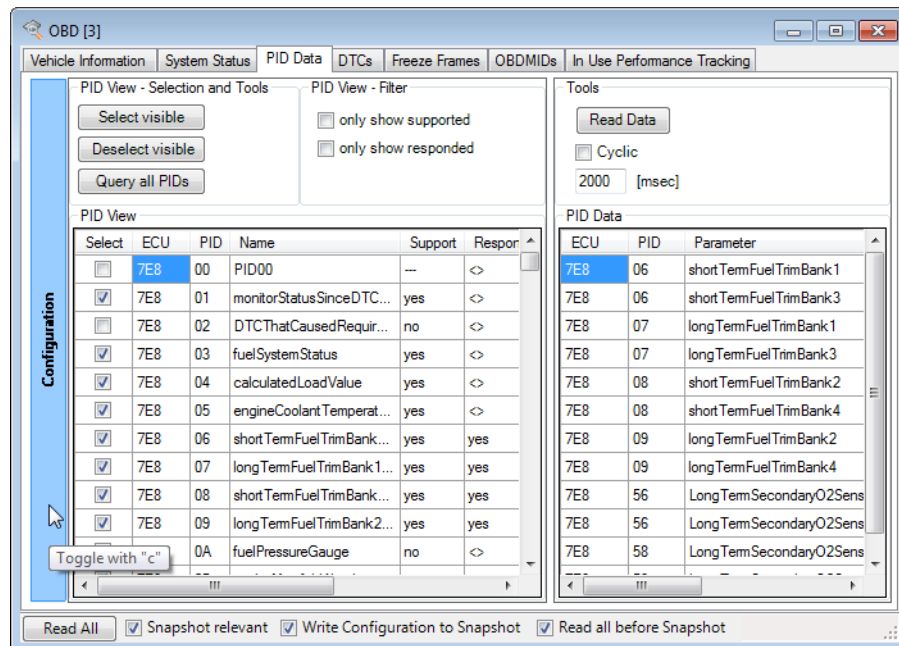
Information from service \$01 is contained in this tab, i.e. current diagnostic data from the powertrain.

PIDs (parameter identifiers) are the identifiers for the information supported by the engine ECU.



**Fig. 7-4** "OBID" User View – "PID Data" Tab

To select the PIDs to be queried, click **Configuration** – the window then shows additional fields (shown below with an \*).



**Fig. 7-5** “OBID” User View – “PID Data” Tab in Configuration Mode

The individual fields of the GUI contain the following functions and information:

**\*PID View - Selection and Tools:** Using these buttons, you can make a kind of global selection of the PIDs to be queried in the “PID View” list.

- **Select visible**

Selects all PIDs visible in the “PID View” list (see “\*PID View - Filter:” on page 107)

- **Deselect visible**

The selection of visible PIDs is undone

- **Query all PIDs**

Each individual PID is addressed and then checked to see if a response is returned

**\*PID View - Filter:** Uses filter criteria with regard to the display in the “PID View” list. The following options are available:

- **only show supported**

If this option is selected, only the PIDs supported by the ECU are made available for selection in the “PID View” list.

- **only show responded**

If this option is selected, only the PIDs answered by the ECU after **Query all PIDs** (see above) are made available for selection in the “PID View” list.

**Tools:** This field is used to read the data – click **Read Data** for this purpose. To read the relevant information periodically, check the “Cyclic” check box and enter the period duration (in ms).

**\*PID View:** This table displays the selected PIDs. The meaning of each individual entry in the list is described in the following table:

Column	Meaning
Select	Selection of the PID
ECU	Name of the logical link (ECU) from hardware configuration
PID	PID
Name	Explicit name of the PID
Support	Is this PID supported? (queried from ECU)
Responded	Was the query of this PID answered (via <a href="#">Query all PIDs</a> )?
Description	Explanatory text (if in the database)

**Tab. 7-6** PID View

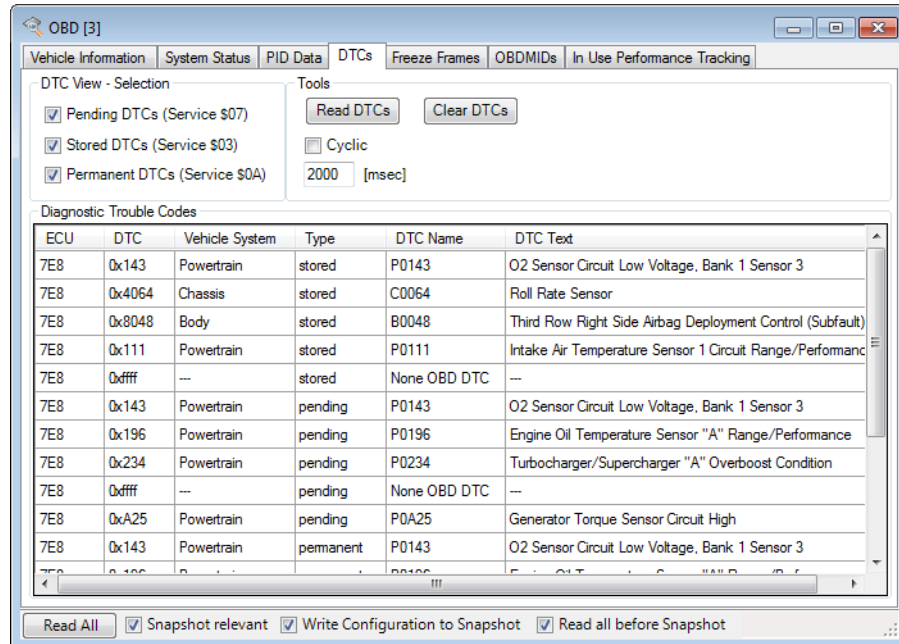
**PID Data:** This table displays the results of the query/queries. The meaning of each individual entry in the list is described in the following table:

Column	Meaning
ECU	Name of the logical link (ECU) from hardware configuration
PID	PID
Parameter	Explicit name as one PID can consist of several pieces of information
Value	Physical value of the parameter
Unit	Unit of PID (if available)

**Tab. 7-7** PID Data

### The "DTCs" Tab

Information from services \$03, \$04, \$07, and \$0A is contained in this tab.



**Fig. 7-6** "ODX" User View – "DTCs" Tab

The individual fields of the GUI contain the following functions and information:

**DTC View Selection:** Selection options for the display of the DTCs in the "Diagnostic Trouble Codes" list. The following options are available:

- Pending DTCs  
All pending DTCs (service \$07) are displayed
- Stored DTCs  
All stored DTCs (service \$03) are displayed
- Permanent DTCs  
All permanent DTCs (service \$0A) are displayed

**Tools:** This field is used to read the Diagnostic Trouble Codes – click **Read DTCs** for this purpose.

To read the relevant information periodically, check the "Cyclic" check box and enter the period duration (in ms). To reset all DTCs, click **Clear DTCs**.

**Diagnostic Trouble Codes:** This table displays the queried DTCs.

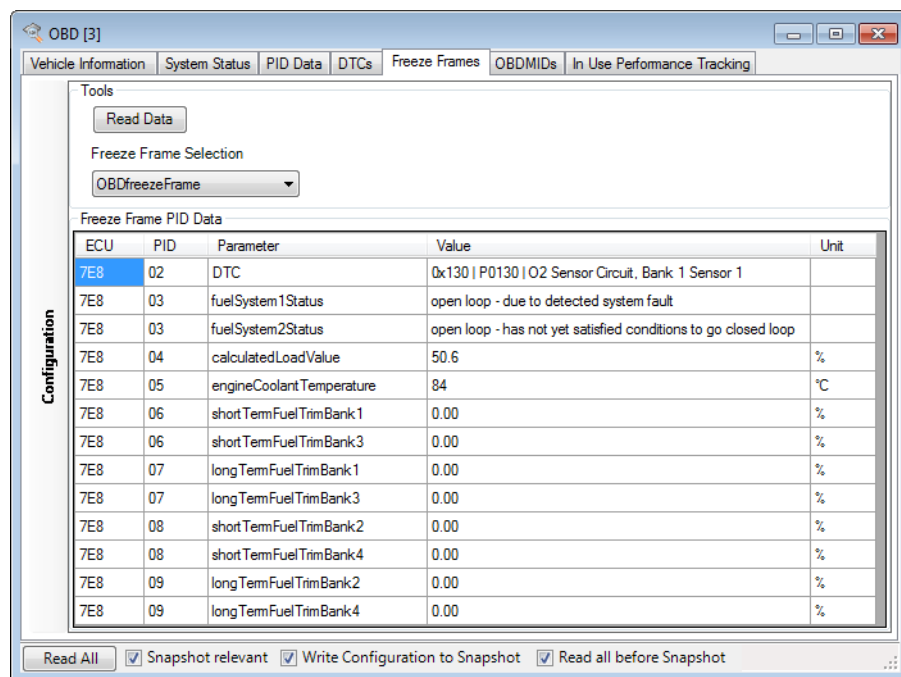
The meaning of each individual entry in the list is described in the following table:

Column	Meaning
ECU	Name of the logical link (ECU) from hardware configuration
DTC	DTC in hex notation (e.g. 0x143)
Vehicle System	"Powertrain", "Chassis", "Body" or "Network"
Type	"Pending" or "Stored"
DTC Name	Name of the DTC (e.g. P0143)
DTC Text	Explanatory text on the DTC (e.g. "O2 Sensor Circuit Low Voltage, Bank 1 Sensor 3")

**Tab. 7-8** Diagnostic Trouble Codes

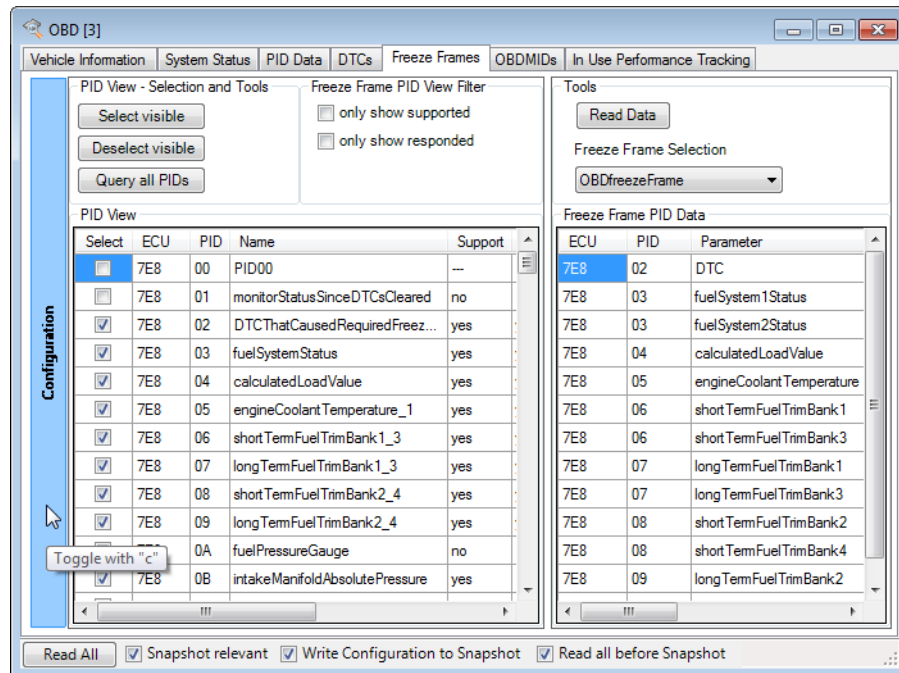
*The "Freeze Frames" Tab*

Information from service \$02 is contained in this tab.



**Fig. 7-7** "OBD" User View – "Freeze Frames" Tab

To select the freeze-frame data to be displayed, click **Configuration** – the window then shows additional fields (shown below with an \*).



**Fig. 7-8** "OBD" User View – "Freeze Frames" Tab in Configuration Mode

The individual fields of the GUI contain the following functions and information:

**\*PID View - Selection and Tools:** Using these buttons, you can make a kind of global selection of the freeze frame PIDs to be queried in the "PID View" list.

- **Select visible**

Selects all PIDs visible in the "PID View" list (see " \*Freeze Frame PID View Filter:" on page 111)

- **Deselect visible**

The selection of visible PIDs is undone

- **Query all PIDs**

Each individual freeze frame PID is addressed and then checked to see if a response is returned

**\*Freeze Frame PID View Filter:** Uses filter criteria with regard to the display in the „PID View“ list. The following options are available:

- Show supported only

If this option is selected, only those PIDs supported by the ECU are made available for selection in the „PID View“ list.

- Show answered only

If this option is selected, only the PIDs answered by the ECU after **Query all PIDs** (see above) are made available for selection in the „PID View“ list.

**Tools:** This field is used to read the data – click **Read Data** for this purpose.

In the "PID View" field, you can choose between general OBD Freeze Frames and manufacturer-specific Freeze Frames.

**\*PID View:** In this table, the Freeze Frame PIDs to be queried are selected. The meaning of each individual entry in the list is described in the following table:

Column	Meaning
Select	Selection of the PID
ECU	Name of the logical link (ECU) from hardware configuration
PID	PID
Name	Explicit name of the PID
Support	Is this PID supported?
Responded	Was the query of this PID answered (via <a href="#">Query all PIDs</a> )?
Description	Explanatory text (if in the database)

**Tab. 7-9** PID View

**Freeze Frame PID Data:** This table displays the results of the query/queries. The meaning of each individual entry in the list is described in the following table:

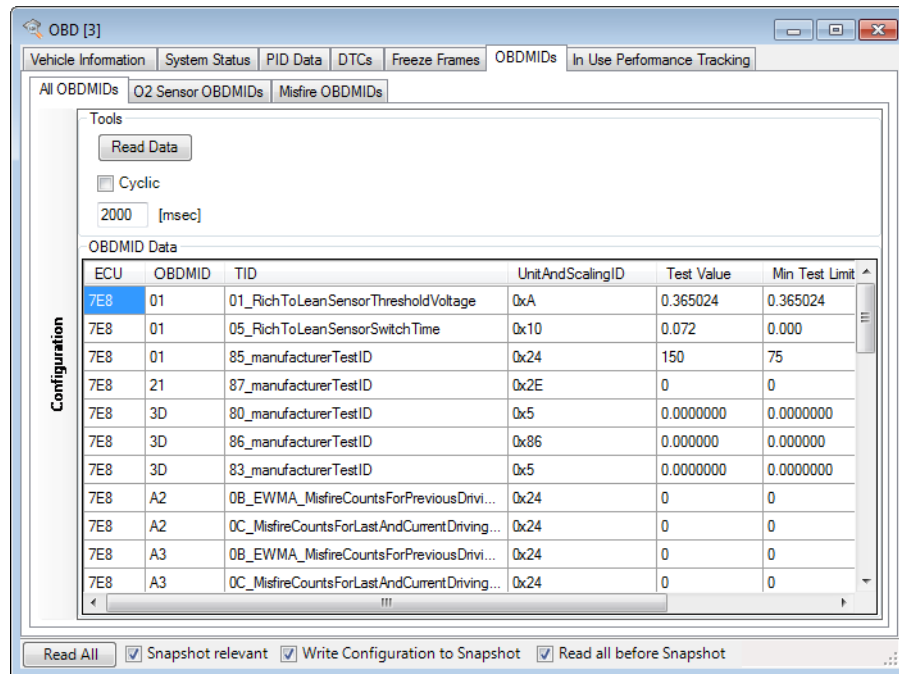
Column	Meaning
ECU	Name of the logical link (ECU) from hardware configuration
PID	PID
Parameter	Explicit name as one PID can consist of several pieces of information
Value	Physical value of the parameter

**Tab. 7-10** Freeze Frame PID Data



### The "OBDMIDS" Tab

The OBD Monitor IDs of specially monitored systems (service \$06) are queried in this tab.

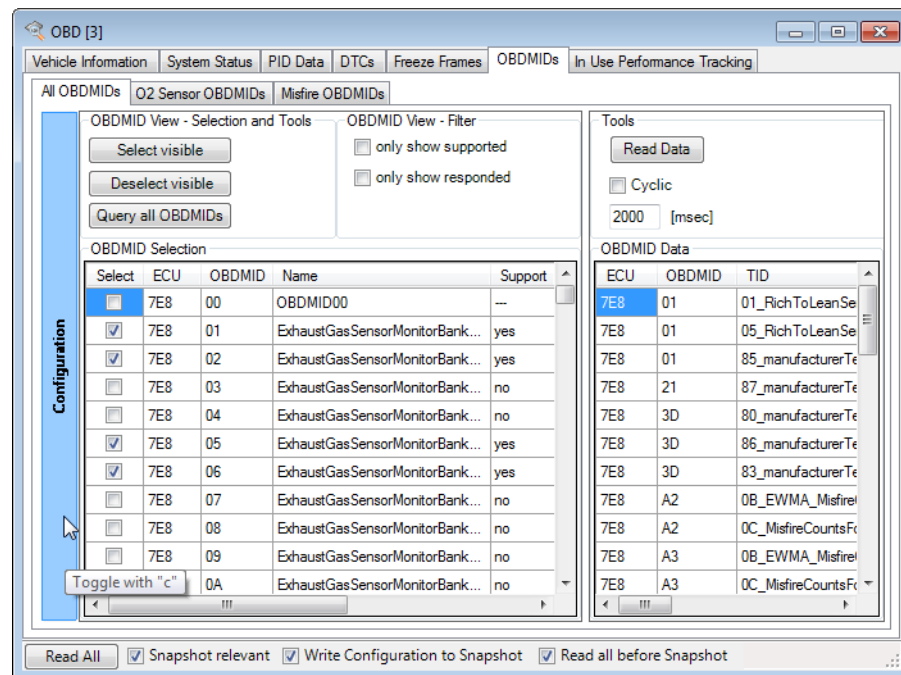


**Fig. 7-9** "OBD" User View – "OBDMIDS" Tab

For reasons of clarity, this window is divided into several tabs:

- "All OBDMIDS" tab  
All OBDMIDs are displayed in this tab
- "O2 Sensor OBDMIDS" tab  
All OBDMIDs connected with the O2 sensor monitor are displayed in this tab
- "Misfire OBDMIDS" tab  
All OBDMIDs connected with the misfire monitor are displayed in this tab

To select the OBDMIDs, click **Configuration** – the window then shows additional fields (shown below with an \*).



**Fig. 7-10** "OBD" User View – "OBDMIDS" Tab in Configuration Mode

The individual fields of the GUI contain the following functions and information:

**\*OBDMID View - Selection and Tools:** Using these buttons, you can make a kind of global selection of the OBDMIDs to be queried in the "OBDMID Selection" list.

- **Select visible**

Selects all OBDMIDs visible in the "OBDMID Selection" list (see "\*OBDMID View - Filter:" on page 114).

- **Deselect visible**

The selection of visible OBDMIDs is undone

- **Query all OBDMIDs**

Each individual OBDMID is addressed and then checked to see if a response is returned

**\*OBDMID View - Filter:** Uses filter criteria with regard to the display in the "OBDMID Selection" list. The following options are available:

- only show supported

If this option is selected, only those OBDMIDs supported by the ECU are made available for selection in the "OBDMID Selection" list.

- only show answered

If this option is selected, only the OBDMIDs answered by the ECU after **Query all OBDMIDs** (see above) are made available for selection in the "OBDMID Selection" list.

**Tools:** This field is used to read the data – click **Read Data** for this purpose. To read the relevant information periodically, check the “Cyclic” check box and enter the period duration (in ms).

**\*OBDMID Selection:** The meaning of each individual entry in the list is described in the following table:

Column	Meaning
Select	Selection of the OBDMID
ECU	Name of the logical link (ECU) from hardware configuration
OBDMID	On-Board Diagnostic Monitor ID
Name	Explicit name of the OBDMID
Support	Is this OBDMID supported?
Responded	Was the query of this OBDMID answered (via <b>Query all OBDMIDs</b> )?
Description	Explanatory text (if in the database)

**Tab. 7-11** OBDMID Selection

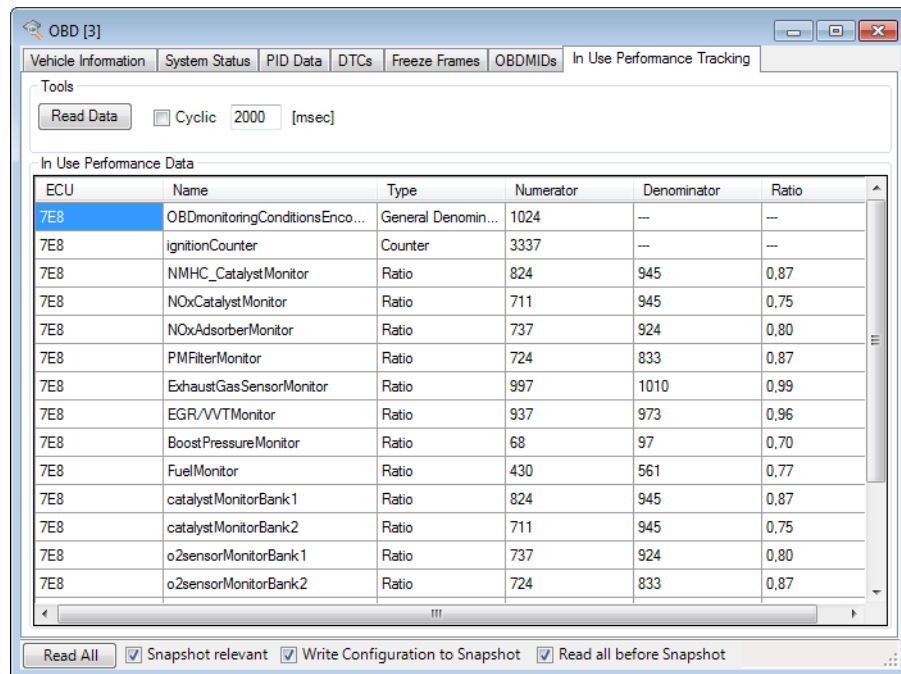
**OBDMID Data:** The meaning of each individual entry in the list is described in the following table:

Column	Meaning
ECU	Name of the logical link (ECU) from hardware configuration
OBDMID	On-Board Diagnostic Monitor ID
TID	Test ID of service 08
UnitAndScalingID	Unit and Scaling ID (1 Byte)
Test Value	Value read from the ECU
Min Test Limit	Minimum test limit
Max Test Limit	Maximum test limit
OBDMID Name	Explicit name of the OBDMID
Description	Explanatory text (if in the database)

**Tab. 7-12** Freeze Frame PID Selection

*The “In Use Performance Tracking” Tab*

In this tab the data of the In-Use Performance Tracking of service \$09 is displayed.



**Fig. 7-11** “OBDM” User View – “In Use Performance Tracking” Tab

The individual fields of the GUI contain the following functions and information:

**Tools:** This field is used to read the data – click **Read Data** for this purpose. To read the relevant information periodically, check the “Cyclic” check box and enter the period duration (in ms).

**In Use Performance Data:** This table displays the queried information of service \$09. The meaning of each individual entry in the list is described in the following table:

Column	Meaning
ECU	Name of the logical link (ECU) from hardware configuration
Name	Name of the parameter
Type	“General Denominator”, “Counter”, “Denominator” or “Numerator”
Numerator	Tracks the number of times that all conditions necessary for a specific monitor to detect a malfunction have been encountered
Denominator	Tracks the number of times that the vehicle has been operated in the specified conditions. These conditions are specified for each monitored component or system.
Ratio	Ratio of the values above

**Tab. 7-13** ODX Tracking Information

## 7.2 Data Logging Configuration

---

The "Data Logging Configuration" ([ODX → Data Logging Configuration](#)) is used to determine how snapshot data is saved. For more details on the snapshot function, refer to the sections "User Views" on page 68 and "Snapshots" on page 122.

### **Note**

*Please note that only the data of user views configured accordingly are recorded in the snapshots. The snapshot icon at the extreme left of the title bar of the user views indicates whether the data of the relevant user view is recorded in the snapshot.*

The settings which you specified can be exported to a file to be used later. Settings exported previously can be reimported from files exported previously.

The "DataLoggingConfig" window contains two tabs:

- **File**  
settings on file name and storage location of the snapshot
- **Header**  
meta information added to the snapshot data

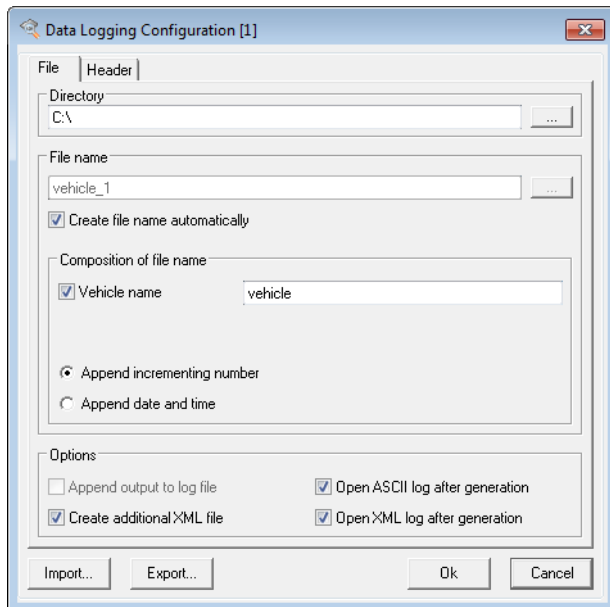
### **To configure Data Logging**

---

- Select [ODX → Data Logging Configuration](#).
- Set the configuration to correspond to your requirements. The following sections explain the significance of the input fields and options.
- Click **OK**.

*“File” Tab*

This is where you specify in which directory and with which file name the snapshot data is to be saved.



**Fig. 7-12** Data Logging Configuration - “File” Tab

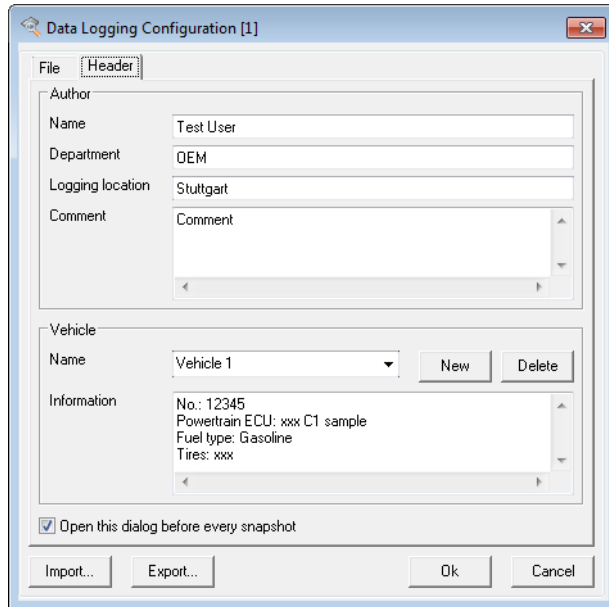
Input Field or Option	Meaning
Directory	Directory in which the snapshot files are saved
File name	File name for the snapshot files. If you activated the option “Create file name automatically”, you cannot enter a file name here.
Create file name automatically	The file name is created automatically
Composition of file name	This is where you specify the individual parts and composition of the file name
Vehicle name	Activate this option to include the vehicle name from the input field in the file name created automatically.
Append incrementing number	An automatically incrementing number is added to the file name.
Append date and time	The current date and the current time (format: "yyyymmdd_hhmmss") are added to the file name
Append output to log file	The snapshot data is appended to the existing data in the log file. This option is only available when the file name is generated manually.

<b>Input Field or Option</b>	<b>Meaning</b>
Open ASCII log after generation	The snapshot file is opened automatically
Create additional XML file	An additional XML file is created with the snapshot data
Open log file after generation	The XML file is opened automatically as HTML in your standard browser.

**Tab. 7-14** Data Logging Configuration - "File" Tab

*“Header” Tab*

This is where you specify which meta information should be added to the snapshot data.



**Fig. 7-13** Data Logging Configuration - “Header” Tab

Input Field or Option	Meaning
Name	Your name
Department	Your department within your company
Logging location	The location of the recorded data
Comment	Notes on data recording
Vehicle	List with vehicle names
Name	Name of vehicle
Information	Notes on vehicle
Open this dialog before every snapshot	Activate this option to open this dialog box before the snapshot data is saved to adapt the meta information for the snapshot.

**Tab. 7-15** Data Logging Configuration - “Header” Tab



### *Exporting and Importing Configurations*

---

All configuration data can be exported as XML files and then imported again. Proceed as follows:

#### **To export Data Logging Configuration**

---

- Select **ODX → Data Logging Configuration...**  
The "DataLoggingConfig" window opens.
- Click **Export...**  
A file selector window opens.
- Enter the required file name and click **Save**.

#### **To import Data Logging Configuration**

---

- Select **ODX → Data Logging Configuration...**  
The "DataLoggingConfig" window opens.
- Click **Import...**  
A file selector window opens.
- Enter the name of the file to be imported and click **Open**.

### 7.3 Snapshots

With the snapshot function (**ODX → Snapshot**) it is possible to save data read by from ECU in a file.

#### **Note**

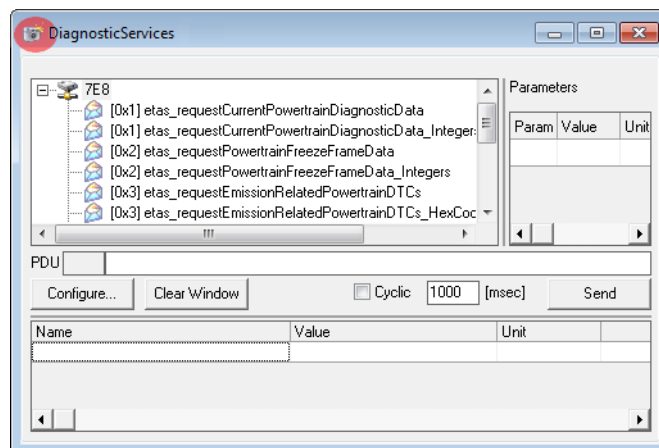
*A snapshot saves the diagnostic data currently stored in all snapshot-relevant INCA user views. Please ensure that the windows are updated manually before the snapshot function is triggered!*

Meta data you can configure is added to the data read from the ECU (see the section "Data Logging Configuration" on page 117).

You define which data is to be recorded in a snapshot (option "Snapshot relevant"). The data of the following "user views" can be recorded in snapshots:

- Diagnostic Services (see "Diagnostic Services" on page 72)
- ECU Identification (see "ECU Identification" on page 76)
- DiagTroubleCode (see "Diagnostic Trouble Code" on page 80)
- Memory Dump (see "Memory Dump" on page 87)
- Sequence (see "Sequence" on page 96)
- OBD (see "OBD" on page 101)

The Snapshot icon - a small camera - at the left of the title bar of the relevant dialog window indicates whether the data of a user view is recorded in the snapshot.



**Fig. 7-14** User View Dialog Box with Snapshot Icon

### To record a snapshot

---



- Click **Snapshot** from the toolbar of the INCA experiment environment.

or

- Select **ODX** → **Snapshot**.

If you activated the relevant option in the "Data Logging Configuration", a dialog box for the snapshot meta data is displayed. If you did not activate the option, the data is written immediately.

A screenshot of the 'Data Logging Configuration' dialog box. The dialog has a title bar 'Data Logging Configuration [1]' and a 'File' menu. It is divided into two main sections: 'Author' and 'Vehicle'. The 'Author' section contains fields for 'Name' (Test User), 'Department' (OEM), 'Logging location' (Stuttgart), and a 'Comment' text area. The 'Vehicle' section contains a 'Name' dropdown menu (Vehicle 1) with 'New' and 'Delete' buttons, and an 'Information' text area containing 'No.: 12345', 'Powertrain ECU: xxx C1 sample', 'Fuel type: Gasoline', and 'Tires: xxx'. At the bottom, there is a checked checkbox 'Open this dialog before every snapshot' and buttons for 'Import...', 'Export...', 'OK', and 'Cancel'.

- Enter the relevant meta data. For further details on "Data Logging Configuration", refer to the section "Data Logging Configuration" on page 117.

- Click **OK**.

The snapshot data is written. If you have set the "Data Logging Configuration" accordingly, the data written is displayed accordingly.

Author	
Name:	Test User
Department:	OEM
Location:	Stuttgart
Comment:	Comment

Vehicle	
Name:	Vehicle 1
Comment:	No.:12345 Powertrain ECU: xxx C1 sample Fuel type: Gasoline Tires: xxx

### 1. Snapshot

Date: 2012-04-11  
Time: 17:04:59

#### 1.1 Memory Dump (Memory Dump)

Name	Value	Unit
0x0	FF FE 1A A1 10 15 5C CE D0 E3 FF FE 1A A1 10 15	
0x10	5C CE D0 E3 FF FE 1A A1 10 15 5C CE D0 E3 FF FE	
0x20	1A A1 10 15 5C CE D0 E3 FF FE 1A A1 10 15 5C CE	
0x30	D0 E3 FF FE 1A A1 10 15 5C CE D0 E3 FF FE 1A A1	
0x40	10 15 5C CE D0 E3 FF FE 1A A1 10 15 5C CE D0 E3	
0x50	FF FE 1A A1 10 15 5C CE D0 E3 FF FE 1A A1 10 15	
0x60	5C CE D0 E3	

## 7.4 Diagnostic Signals in the INCA Variable Selection

The diagnostic data that can be read about the ODX project at the diagnostic interface of an ECU can also be measured and recorded in a measure file with INCA and ODX-LINK. The prerequisite for this is the definition of diagnostic signals in the ODX project.

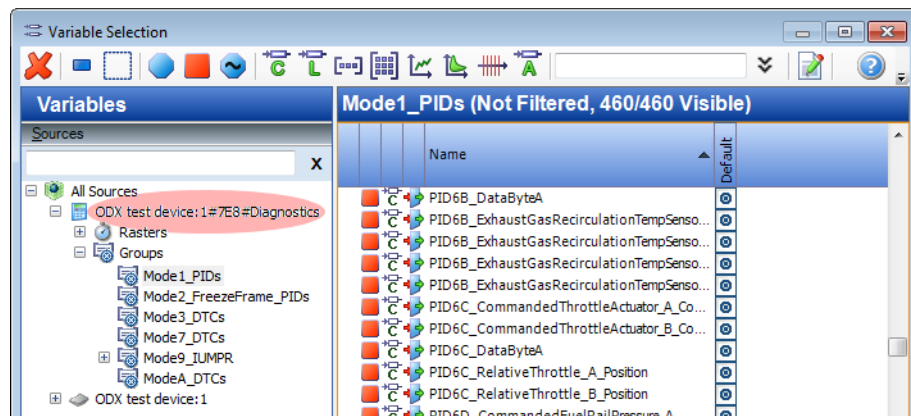
The diagnostic signals of the ODX project are described in the diagnostic signal list (DSL) – the signals defined there can be used in the same way as standard measure variables in INCA's "Variable Selection" window.

If users want to work with their own ODX data, it is possible to generate such signals in the INCA experiment and add them to the ODX project. This function is available for the user views "Diagnostic Services" and "ECU Identification" (see "Adding Diagnostic Signals" on page 126).

When you install ODX-LINK, the ODX project "OBDonCAN\_ETAS" already contains one DSL file. When using this ODX project, the "Variable Selection" window in INCA features an additional diagnostic device with the diagnostic signals of the `OBDSignalList.dsl` file contained.

The diagnostic device is named in accordance with the following conventions:

**<Device name>#<Logical Link>#Diagnostics**



### 7.4.1 Definitions in the Variable Selection Dialog Box

The diagnostic signals in the Variable Selection dialog box are divided into the following signal groups in the OBDonCAN\_ETAS project (with the included signal list):

- **Mode1\_PIDs**  
In the signal group "Mode1\_PIDs", you find information from service \$01 (see "OBD" on page 101), i.e. up-to-date diagnostic data from the powertrain.
- **Mode2\_FreezeFrame\_PIDs**  
In the signal group "Mode2\_FreezeFrame\_PIDs", you find information from service \$02 (see "OBD" on page 101).
- **Mode3\_DTCs**  
In the signal group "Mode3\_DTCs", you find signals that are saved and can be deleted again with service \$03 (see "OBD" on page 101).

- **Mode7\_DTCs**

In the signal group "Mode7\_DTCs", you find trouble codes discovered during the current or last completed driving cycle (see "OBD" on page 101).

- **Mode9\_IUMPR**

- Mode9\_IUMPR\_compression
- Mode9\_IUMPR\_spark

In the signal group "Mode9\_IUMPR", you find signals from In-Use Performance Tracking (see "OBD" on page 101).

- **ModeA\_DTCs**

In the signal group "ModeA\_DTCs", you find signals that have the status "permanent" and cannot be deleted (see "OBD" on page 101).

### Note

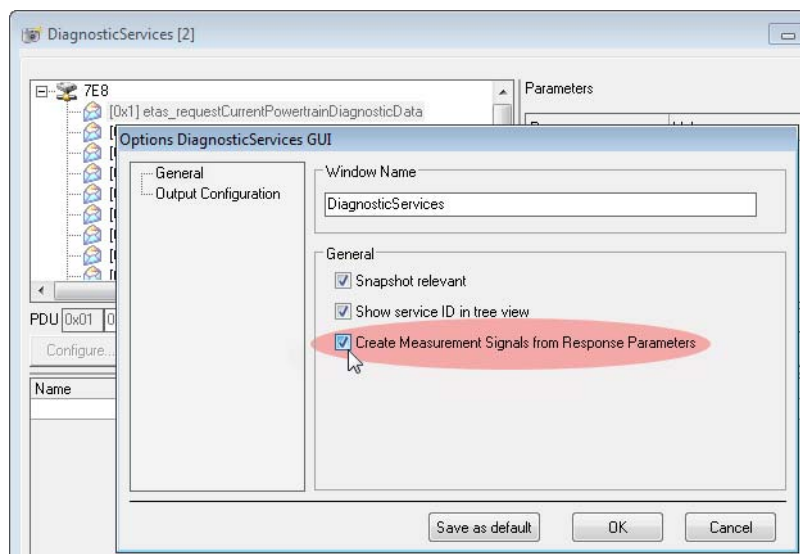
*In generated signal lists, the functional groups are called "ModeX" (with the ETAS OBD project) or "ServiceXY" (with other ODX projects).*

## 7.4.2 Adding Diagnostic Signals

To make diagnostic signals available for use in the Variable Selection dialog box, proceed as follows:

### To add diagnostic signals

- Open the experiment.
- Select **ODX** → **User views** → **DiagnosticServices**.  
The "DiagnosticServices" window opens.
- Select **Configure**.
- Select the option "Create Measurement Signals from Response Parameters".



- Click **OK**.  
This means that every response parameter issued is added to the signal list of the selected logical link of the ODX project currently open together with the integer/float value for the diagnostic service executed subsequently providing this signal is not already contained. A new signal list is created if there is not one available in the ODX project.

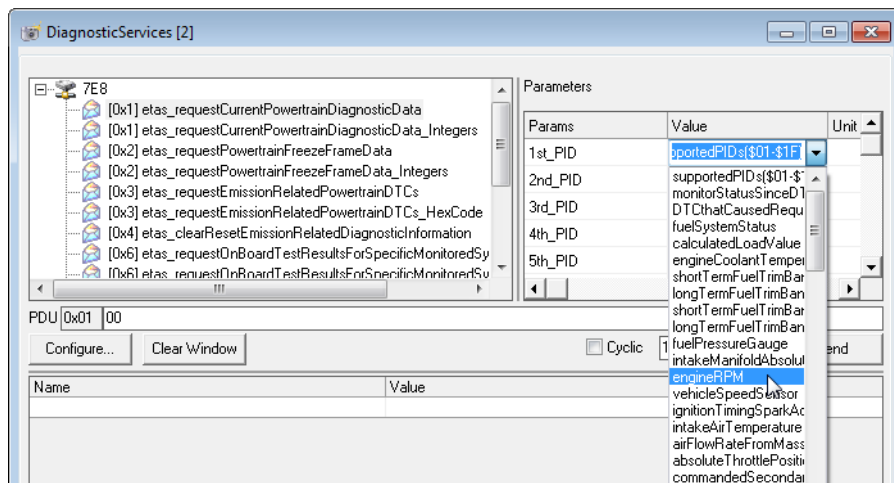
**Note**

Signals are created for the logical link (and thus also only for the corresponding device in the Variable Selection dialog box) for which you have selected the service!

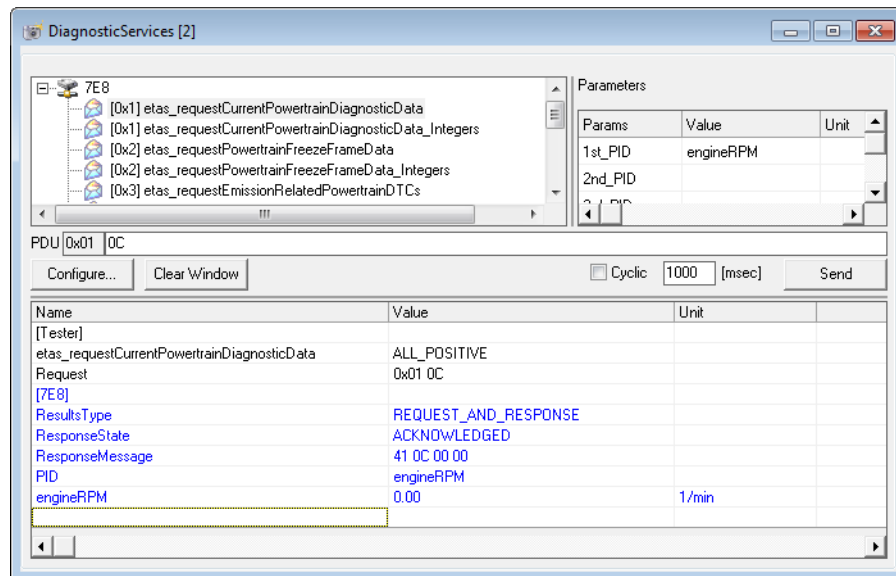
**Note**

The selection of this option only applies for the services sent subsequently within the current experiment session and is **not** saved in **Save as default!**

- Select the service and the parameter you want to measure (and thus add to the DSL).



- Click **Send**.

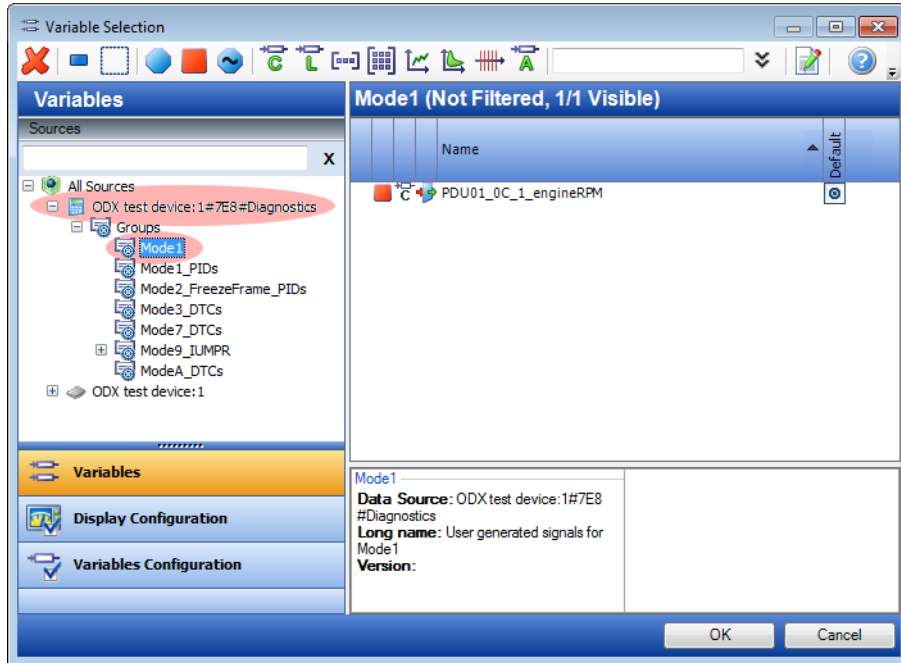


One signal is added to the DSL for every numeric value (Integer/Float) in the response.

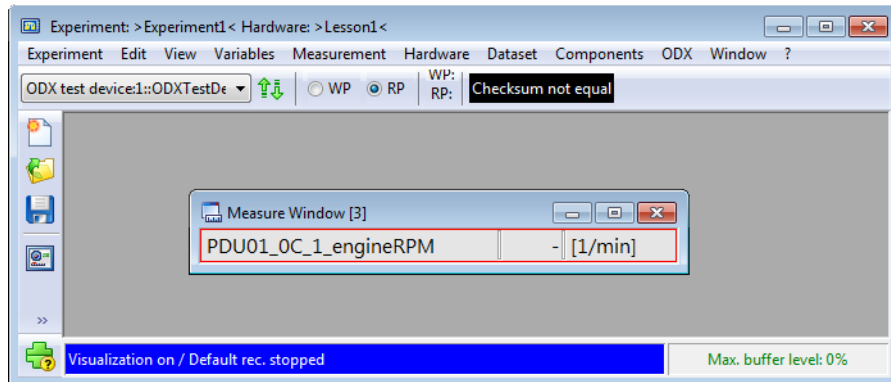
- Close the experiment.  
A note is displayed explaining that the ODX project has been modified (as new signals have been defined) (the diagnostic signal list is saved with the ODX project in the INCA database) and asking whether you want to save the ODX project.
- Select **Yes** to save the new signals.
- Open the experiment again.



- Select **Variables** → **Variable Selection**.  
The added diagnostic signals are displayed (here in group "Mode 1").

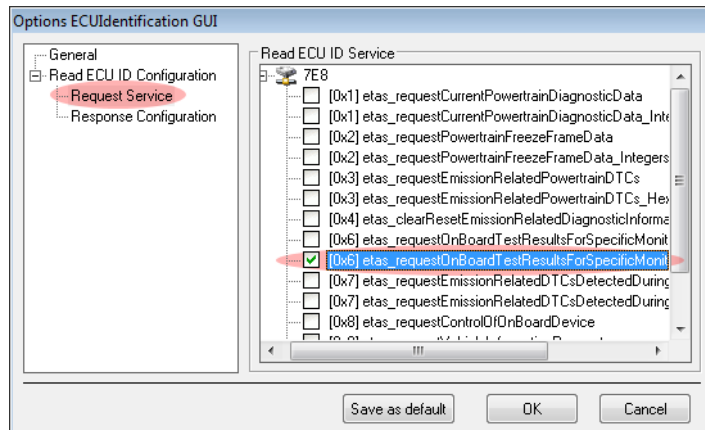


- Select the signal in the list and right-click it.
- Select **Add to** → **Layer\_1** → **New** → **Measure Window** to display the signals and click **OK**.
- Select **Measurement** → **Start Visualization**.  
The measured diagnostic signals are displayed.

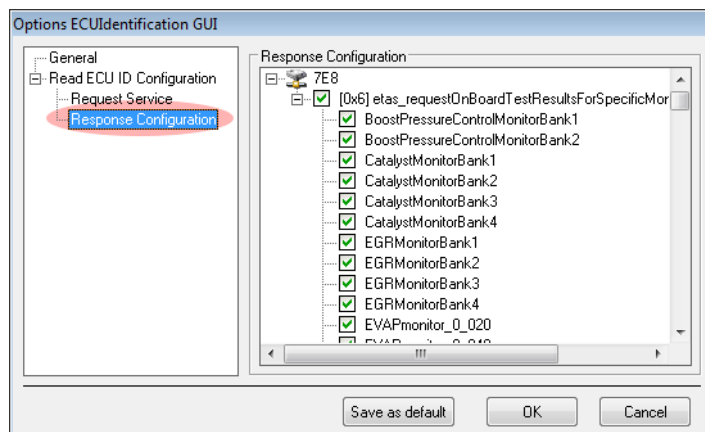


- Stop the measurement.
- To include signals for all data that can be called up using this service in the list of diagnostic signals, select **ODX** → **User views** → **ECUIdentification**.  
The "ECUIdentification" window opens.
- Select **Configure**.

- Here too, activate the option “Create Measurement Signals from Response Parameters”.
- Select (on the left-hand side of the window) “Request Service” and then the required service on the right-hand side of the window.

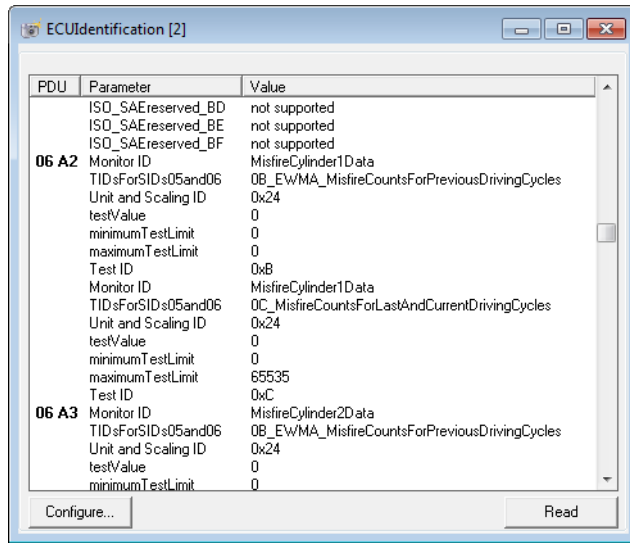


- If you need to configure the response, select “Response Configuration” and configure the response accordingly.



- Click **OK**.

- Click **Read** in the "ECUIdentification" window.

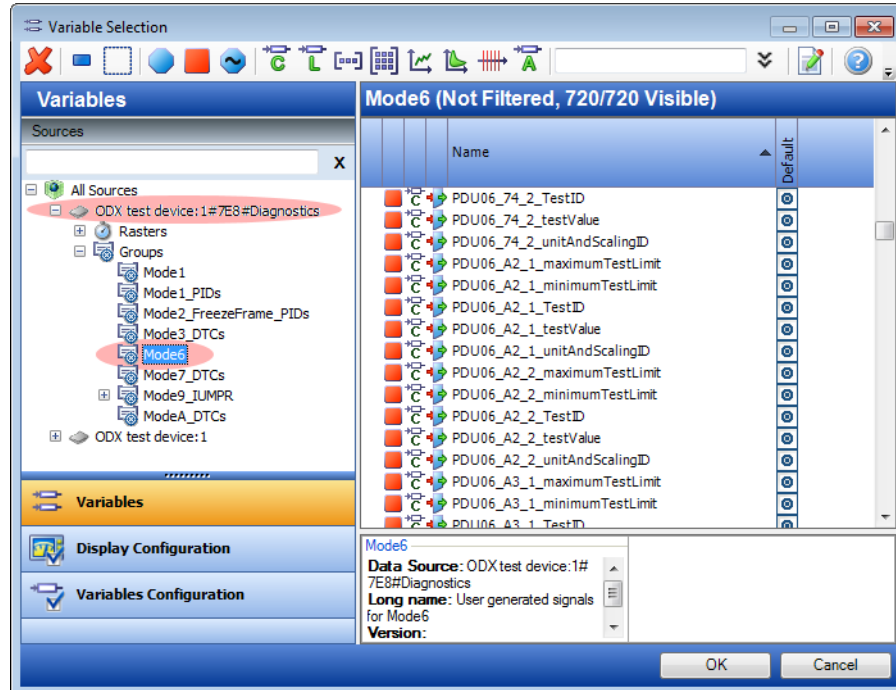


One signal is added to the DSL for every numeric value (Integer/Float) in the response.

- Close the experiment.  
A note is displayed explaining that the ODX project has been modified (as new signals have been defined) (the diagnostic signal list is saved with the ODX project in the INCA database) and asking whether you want to save the ODX project.
- Select **Yes** to save the new signals.
- Open the experiment again.

- Select **Variables** → **Variable Selection**.

The added diagnostic signals are displayed (here in group “Mode6”).



- Select the signals in the list, add them (as above) to a measure window and start the measurement.

#### 7.4.3 Storage Location of the DSL File

The DSL file is stored in the INCA database in the ODX project. This means the signals are available in all workspaces/experiments that use this ODX project.

These can also be transferred to other INCA databases by exporting/importing.

## 8 ODX-LINK Tutorial

---

In this tutorial, you will learn the major operational procedures for ODX-LINK V1.5.

### **Note**

*This tutorial assumes that you have installed INCA and ODX-LINK V1.5. For information on installing ODX-LINK V1.5, see Chapter "Installation" on page 11. This tutorial also assumes that you are familiar with basic INCA operations.*

The tutorial consists of the following lessons:

- "Creating an INCA Workspace" on page 134  
In this lesson, you will create a new INCA workspace for working with ODX-LINK.
- "Preparing an INCA Experiment for ODX without Hardware Connection" on page 139  
In this lesson, you will prepare the INCA experiment for using ODX without any external hardware. The ECU will be simulated, as in the previous lesson.
- "Working with ODX User Views" on page 142  
In this lesson, you will use the ODX user views to perform diagnostics and to display the responses from the ECU.
- "Preparing an INCA Experiment for ODX with Real Hardware" on page 146  
In this exercise, you will get an INCA experiment ready for working with your ECU. You need an A2L file and a HEX file for your ECU.
- "Configuration of ODX User Views" on page 150  
In this lesson, you will modify the configurations for Diagnostic Services, ECU Identification, and Diagnostic Trouble Code.
- "Using the OBD Protocol with ODX-LINK" on page 154  
In this lesson, you will configure INCA for using the OBD protocol and will use the ODX-LINK OBD database for OBD diagnostic communication with a real ECU.
- "Working with the "OBD" User View" on page 159  
In this lesson, you will work with the OBD user view.
- "Using Diagnostic Signal in the Experiment" on page 163  
In this lesson you will learn how diagnostic signals can be used in an experiment.
- "Measuring OBD Data on the Vehicle" on page 166  
In this lesson, you measure diagnostic signals on the vehicle and generate additional ECU-specific measurement signals.

## 8.1 Creating an INCA Workspace

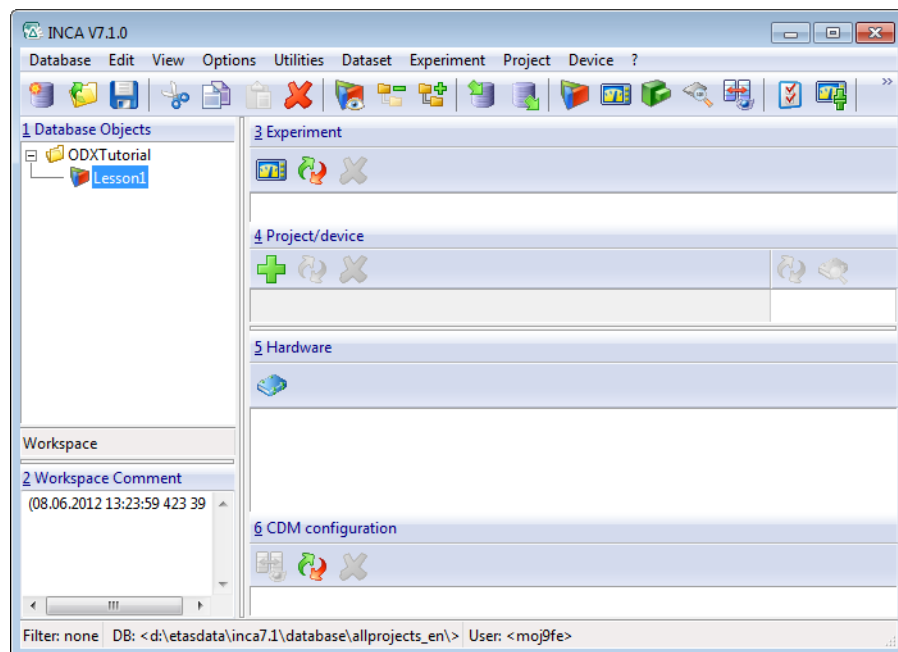
In this lesson, you will create a new INCA workspace for working with ODX-LINK. You do not need any additional devices for this lesson. The “K-Line Test Device” will simulate the ECU in this lesson.

### To create a top folder

- Start INCA.
- Choose **Edit** → **Add** → **Add top folder**.
- Enter `ODXTutorial` and press ENTER.

### To create an INCA workspace

- Choose **Edit** → **Add** → **Workspace**.
- Enter `Lesson1` and press ENTER.



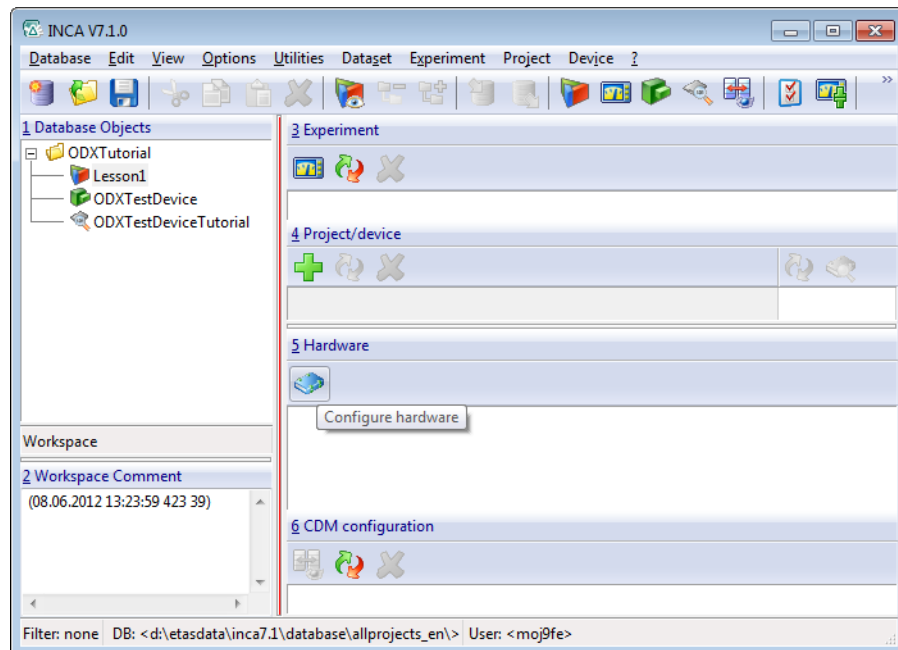
### To add an ECU project

- Select **Edit** → **Add** → **ECU-Project (A2L)...**  
A window opens in which you can select the project file.
- Select the  
`D:\ETASData\INCA7.1\Data\Demo\ODXTestDevice.a2l` file and click **Open**.  
A further window for selecting files opens.  
No ECU program file is required for simulating the ECU.

- So click **Cancel**.  
The ECU project is added to the top folder you selected.

### To add an ODX project

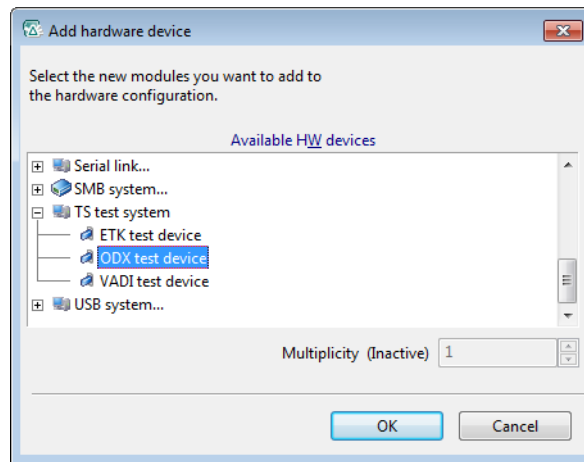
- Change to the INCA main window and select the ODXTutorial folder.
- Select **Edit** → **Add** → **ODX Project...**.  
A file selector window opens.
- Select the file  
C:\ETASData\ODX1\_5\_0\Projects\ODXTestDeviceTutorial\ODXTestDeviceTutorial.prj  
and click **Open**.  
The ODX project is added.



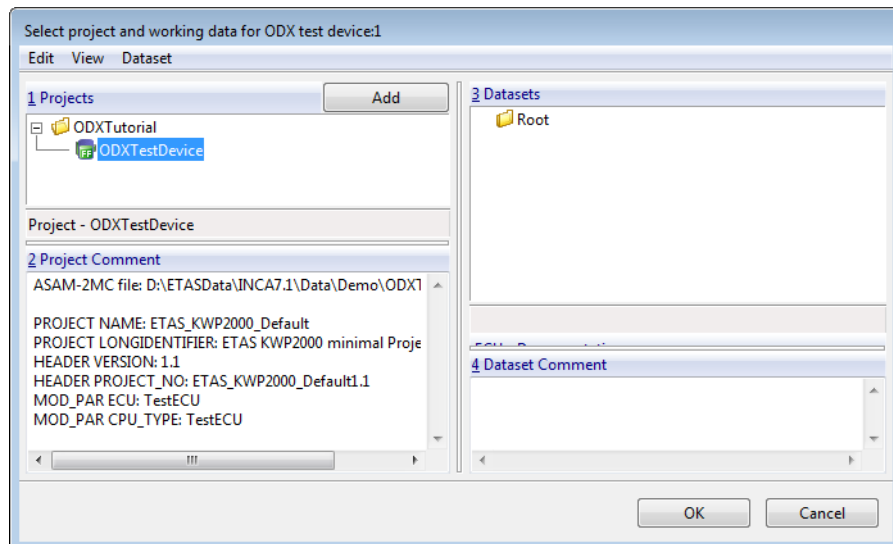
### To add a new device

- Select the "Lesson1" workspace.
- Click the **Configure Hardware** icon.  
The hardware configuration window opens.
- Select **Device** → **Insert**.  
A window for selecting the hardware opens.

- Extend the **TS test system** item.



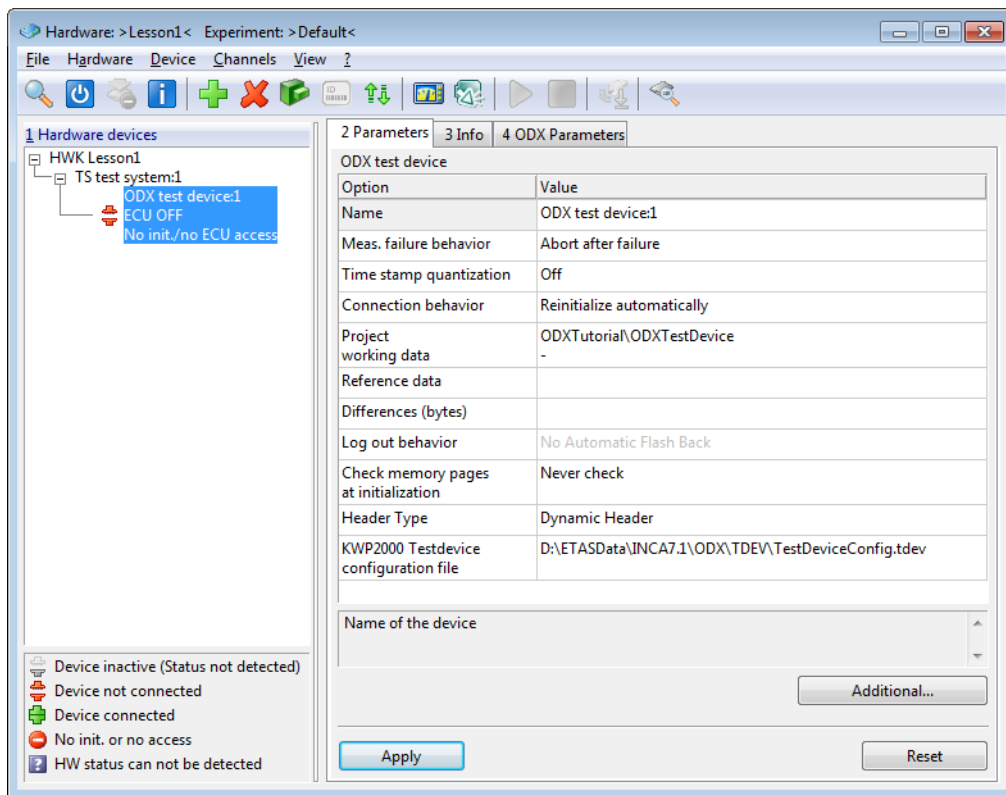
- Select “ODX test device” and click **OK**.  
The window for selecting the project opens.



- Select the “ODXTestDevice” project and click **OK**.  
A file selector window opens.
- Select the `TestDeviceConfig.tdev` file (in the `ETASData\INCA7.1\ODX\TDEV` folder).



- Click **Open**.  
This concludes the hardware configuration.

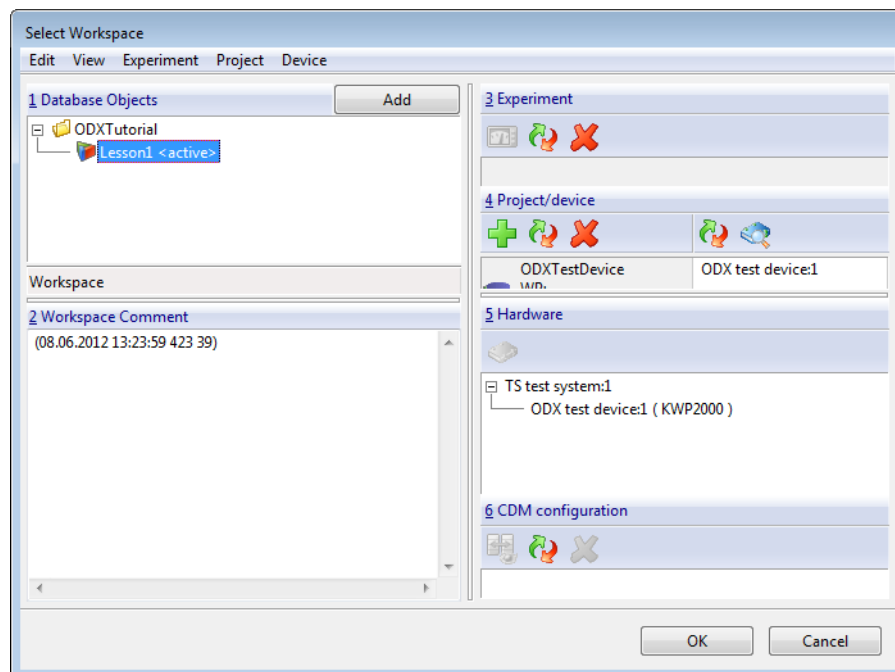


- Close the "Hardware: >Lesson1<" window.

### To add an experiment

- Choose **Edit** → **Add** → **Experiment**.
- Enter `Experiment1` and press ENTER.
- Double-click `Experiment1`.  
Now the dialog window for selecting the workspace will appear.

- Select Lesson1 and click **OK**.



This concludes the hardware and workspace configuration.

- Close the “Experiment >Experiment1<” window.

In this lesson, you have created a new INCA workspace, added an ECU project, an ODX project and hardware definition and created an experiment.

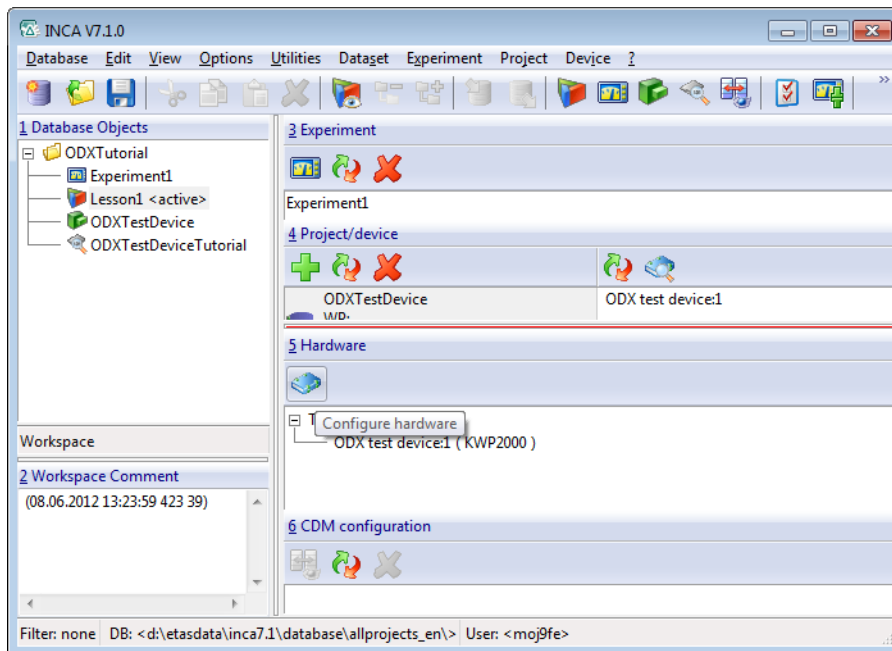
## 8.2 Preparing an INCA Experiment for ODX without Hardware Connection

In this lesson, you will prepare the INCA experiment for using ODX without any external hardware. The ECU will be simulated, as in the previous lesson.

The settings you created in Lesson "Creating an INCA Workspace" on page 134 are prerequisites for this lesson.

### To open an ODX configuration

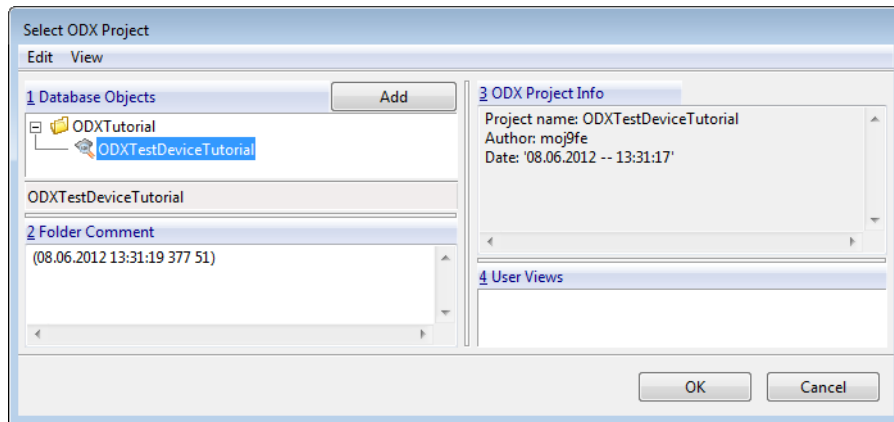
- Switch to the INCA main window.
- Open the `ODXTutorial` folder.
- Select the workspace `Lesson1`.
- Click the **Configure Hardware** icon.



The hardware configuration window opens.

- Choose **Hardware** → **Configure ODX**.

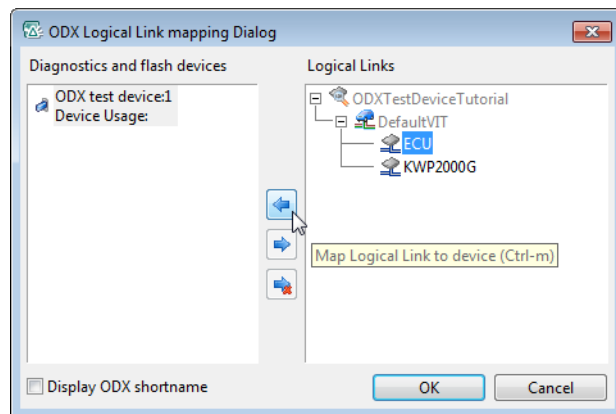
- The dialog window for selecting the ODX configuration is displayed.



- Select the ODXTestDeviceTutorial project file and click **OK**.

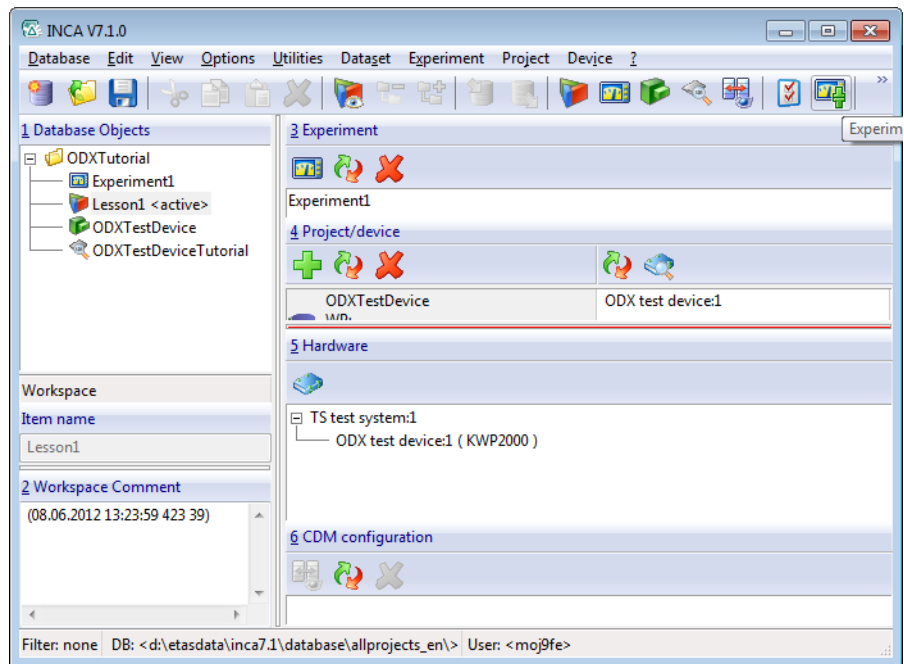
The dialog window for logical link mapping is displayed.

- Select "ECU".



- Click the left arrow.

- Click **OK**.



- Close the hardware configuration window.

In this lesson, you have opened an INCA experiment, opened an ODX configuration and assigned a logical link of the ODX project to an INCA device.

### 8.3 Working with ODX User Views

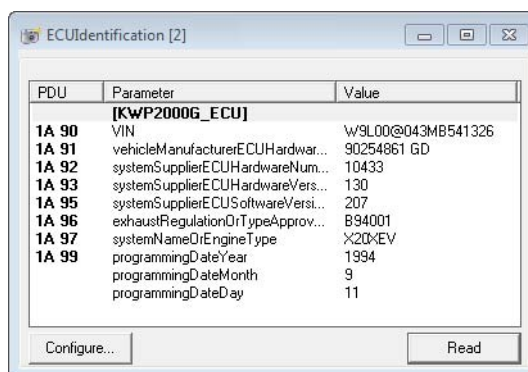
In this lesson, you will use the ODX user views to perform diagnostics and to display the responses from the ECU.

Since in this lesson you will not be working with external hardware either, all responses from the ECU will be simulated. The simulated ECU responses are defined in the `TestDeviceConfig.tdev` file.

The settings you created in Lesson "Preparing an INCA Experiment for ODX without Hardware Connection" on page 139 are prerequisites for this lesson.

#### To display the ECU identification

- Double-click the workspace "Lesson1".  
The experiment "Experiment1" is opened.
- Choose **ODX** → **User views** → **ECUidentification**.  
The dialog window for the ECU identification will be displayed.
- Click **Read**.  
The service request to query the ECU identification is sent to the ECU. The contents of the service request and the ECU response are displayed in the bottom section of the window.



- Close the dialog window.

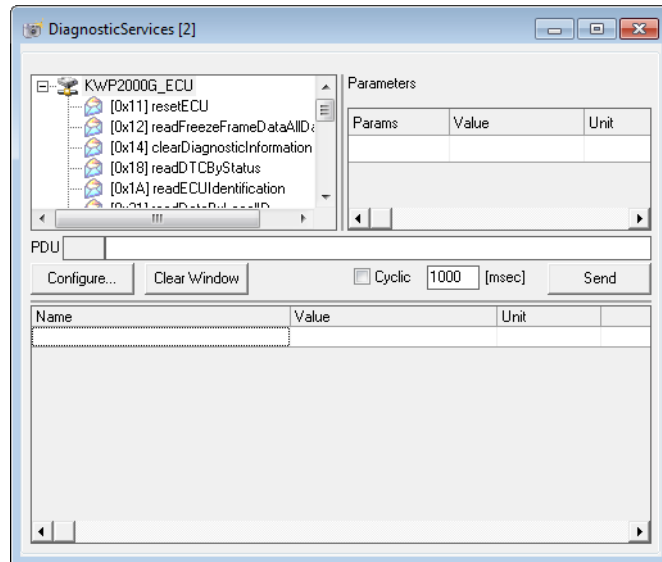
#### Note

After installation, this function is assigned the "[1A] readECUidentification" service identification from the KWP2000 protocol. You can, however, assign any service identification to this function. For further information, see "Configuration of ODX User Views" on page 150.

### To execute a diagnostic service

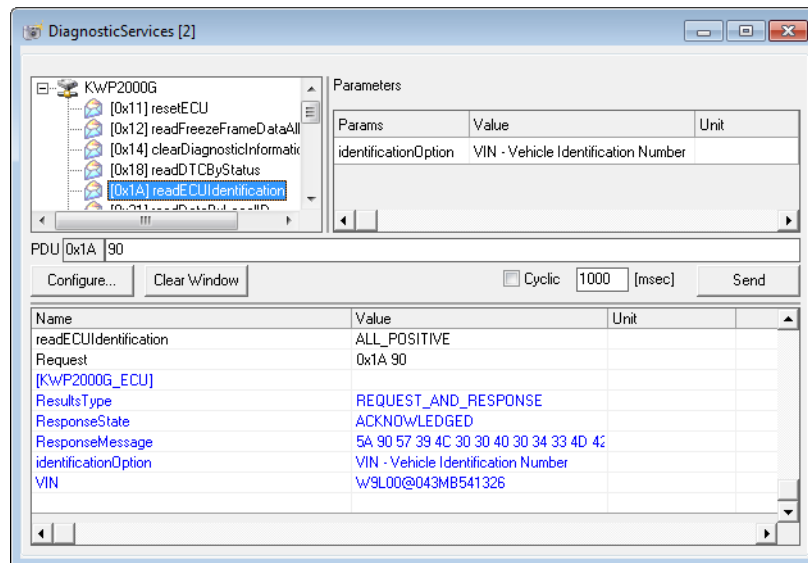
- Choose **ODX** → **User views** → **DiagnosticServices**.

The dialog window for the diagnostic services is displayed.



- Select the [1A] readECUIdentification service.
- In the "Value" column, click the first entry in the "Parameters" field. In the selection list, click the "VIN - Vehicle Identification Number" entry.

- Click **Send**.



The service request is sent to the simulated ECU. The contents of the service request and the ECU response are displayed in the bottom section of the window.

#### **Note**

*Some settings for this window are configurable. For further information, see "Configuration of ODX User Views" on page 150.*

### **To execute a diagnostic service on a regular basis**

- Select the "Cyclic" option.
- Enter 1500 as the cycle time.
- Click **Send**.  
The service request will be sent to the ECU every 1.5 seconds. The contents of the service request and the ECU response are displayed in the bottom section of the window.
- Click **STOP**.
- This stops the cyclic sending of the service request.
- Close the window.

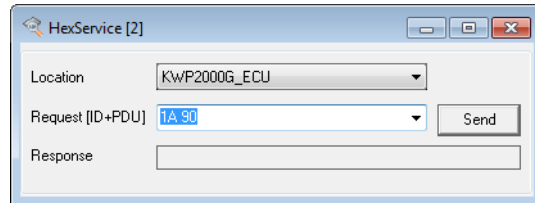


### To execute a freely configurable service request

---

- Choose **ODX** → **User views** → **HexServices**.

The dialog window for the freely configurable diagnostic services is displayed.



- In the "Request" field, enter the service request in hexadecimal notation: Type in 1A 90.
- Click **Send**.  
The ECU response is displayed in the "Response" field.
- Close the dialog window.

In this lesson, you have worked with the user views **ECUIdentification**, **DiagnosticServices** and **HexService**. You have sent the predefined service request for identifying the ECU, a service request from the ODX database, a freely configurable service request to the ECU, and have monitored the response from the ECU.

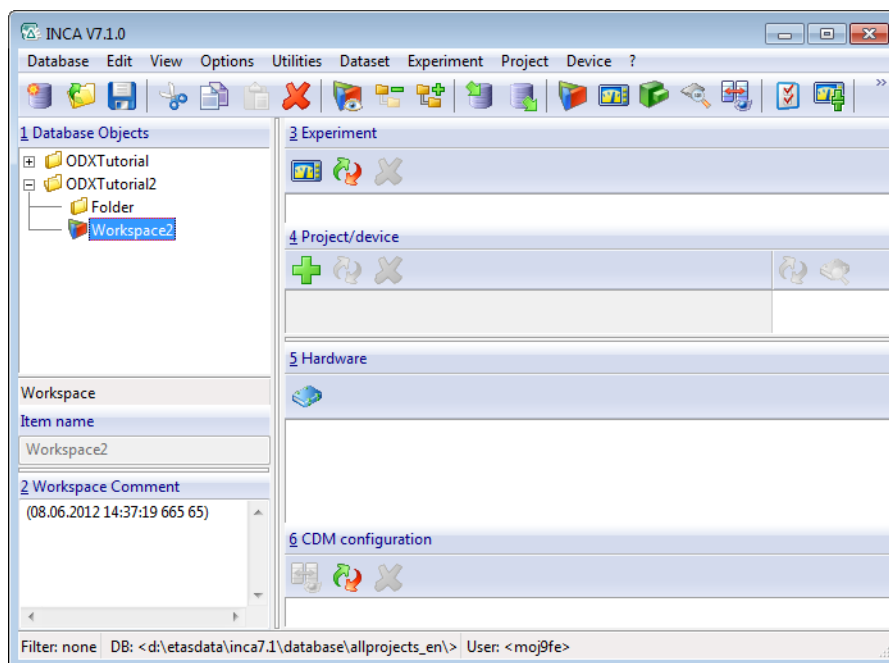
If you have not connected any additional hardware to your system, go on to "Configuration of ODX User Views" on page 150.

## 8.4 Preparing an INCA Experiment for ODX with Real Hardware

In this exercise, you will get an INCA experiment ready for working with your ECU. You need an A2L file and a HEX file for your ECU.

### To create a top folder and INCA workspace

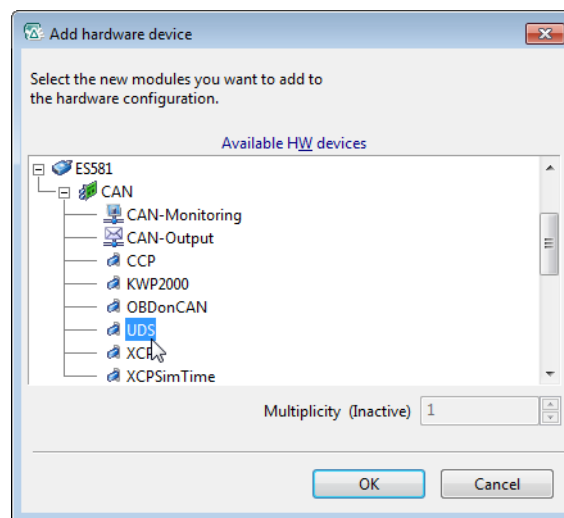
- Start INCA.
- Create a new top folder. Choose **Edit** → **Add** → **Add top folder**. Type in `ODXTutorial2` and press ENTER.
- Add a new workspace. Choose **Edit** → **Add** → **Workspace**. Type in `workspace2` and press ENTER.



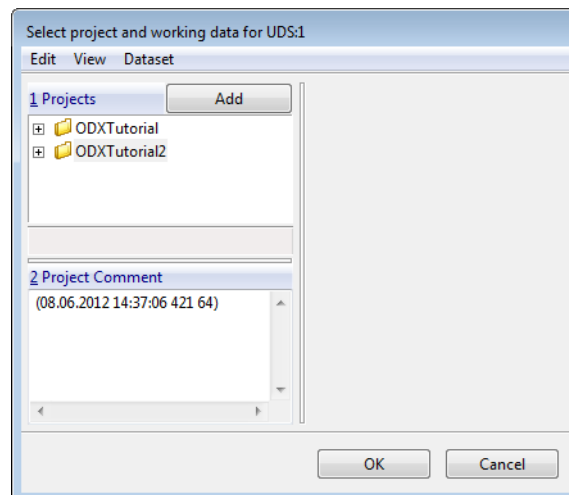
### To add a new device

- Choose **Device** → **Configure hardware**.  
The Hardware Configuration Editor is opened.
- Choose **Device** → **Insert** and select the device you are using.

In the example here, an ES581 with UDS via CAN has been selected. For your own ECU, you may have to select a different device.



- Click **OK**.  
The dialog window for selecting the project is displayed.



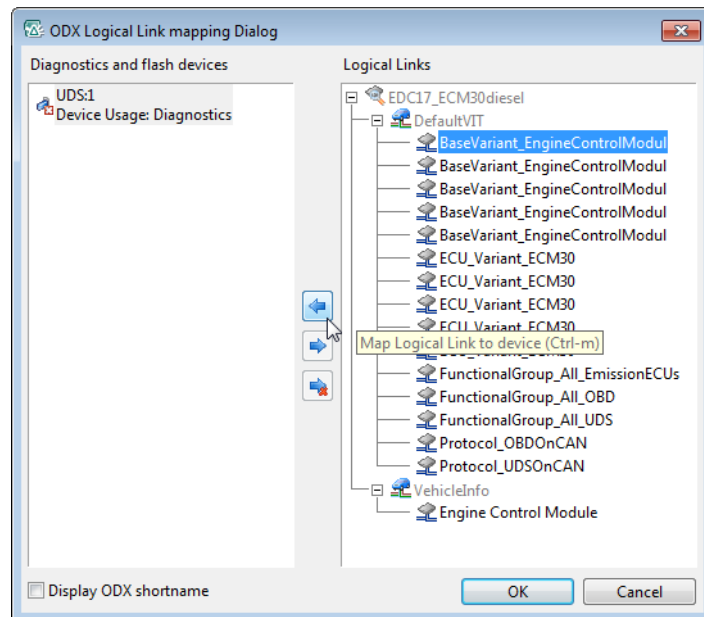
- Select a project or click **Cancel**.

#### Note

*The ODX configuration parameters can be determined from the ODX project - an A2L file is not required!*

### To configure ODX

- Select **Hardware** → **Configure ODX** from the Hardware Configuration Editor.
- Select your ODX project file and click **OK**.  
The ODX project opens.
- Select the logical link and assign it to the diagnostic device using the arrow button.



- Click **OK**.  
The hardware is now configured.
- Close the "Hardware: >Workspace2<" window.

### To add an experiment

- Switch to the "INCA" window and select ODXTutorial2.
- Choose **Add** → **Experiment**. Type in Experiment2 and press ENTER.
- Double-click Experiment2.  
A dialog window for selecting the workspace now appears.
- Select Workspace2 and click **OK**.  
The experiment is opened.
- Click **Close**.  
This concludes the configuration of the hardware and the workspace.
- Close the "Experiment >Experiment2<" window.

In this exercise, you have created an INCA experiment for working with your own ECU, have thus created a new INCA workspace, added your hardware components and a new experiment.

## 8.5 Configuration of ODX User Views

In this lesson, you will modify the configurations for Diagnostic Services, ECU Identification, and Diagnostic Trouble Code.

The following lessons are prerequisites for this lesson:

- "Creating an INCA Workspace" on page 134
- "Preparing an INCA Experiment for ODX without Hardware Connection" on page 139
- "Working with ODX User Views" on page 142

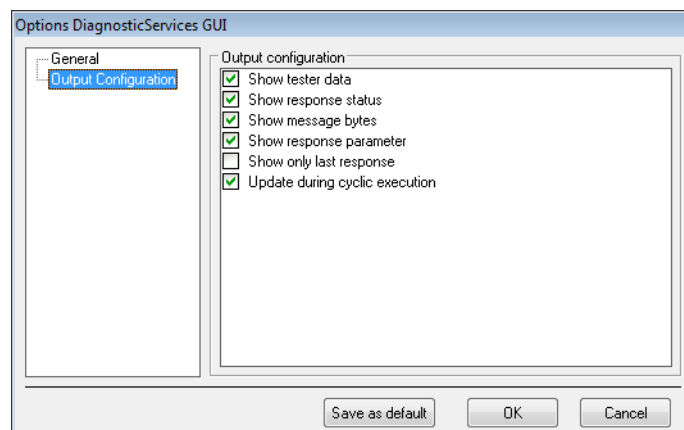
### To open an INCA experiment

- Switch to INCA.
  - Open the "ODXTutorial" folder.
  - Double-click "Lesson1."
- The experiment opens.

### To configure "DiagnosticServices"

- Choose **ODX** → **User views** → **DiagnosticServices**.
- Click **Configure**.
- Select "Output Configuration".

The configuration dialog window for the display will open.



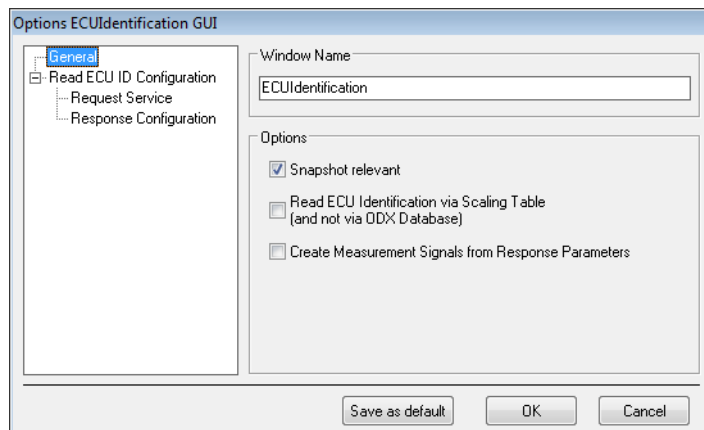
- If you have not already done so, please make the following settings:
  - Select "Show tester data" to enable the service request to be displayed.
  - Select the "Show response status" check box to enable a plain text display of the ECU response.

- Select the "Show message bytes" check box to enable the „response messages“ to be displayed.
- Select the "Show response parameter" check box to enable the interpreted response to be displayed.
- Select "Update during cyclic execution" to refresh the display whenever the service request is periodically repeated.
- Click **OK**.  
Your settings are saved.

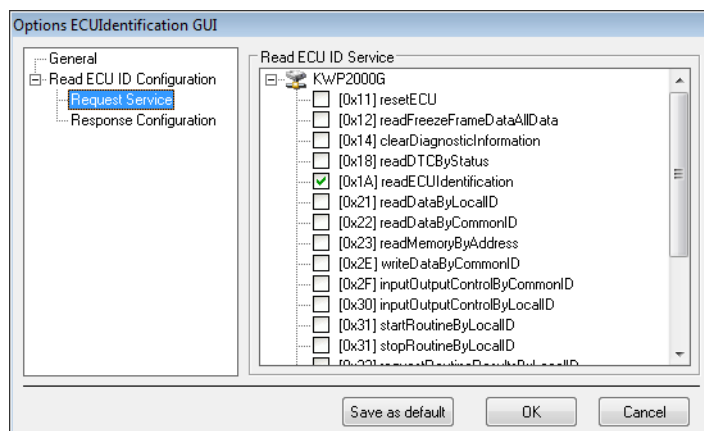
**To configure "ECUIdentification"**

- Choose **ODX → User views → ECUIdentification**.
- Click **Configure**.

The dialog window for configuring the service for identifying the ECU will be displayed.



- Select "Request Service" in the left-hand window.

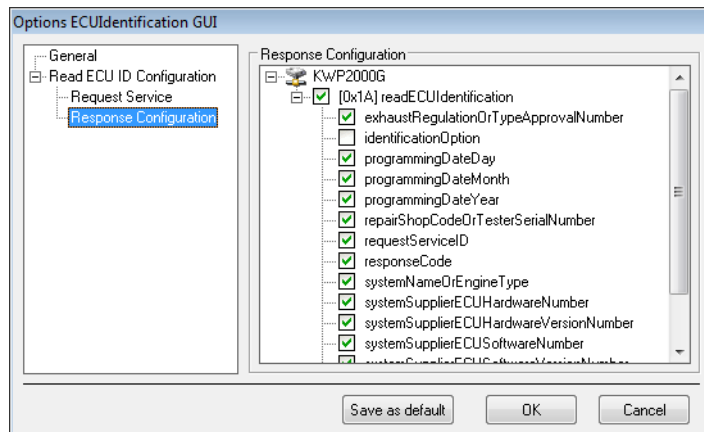


- Select the service to be used for reading the ECU identification.

#### Note

You can as well select any other service – the output always displays the ECU responses of the selected service.

- Select “Response Configuration” in the left-hand window.



- Select the options you want to read from the ECU.
- Click **OK**.  
Your settings are saved.

### To configure "Diagnostic Trouble Code"

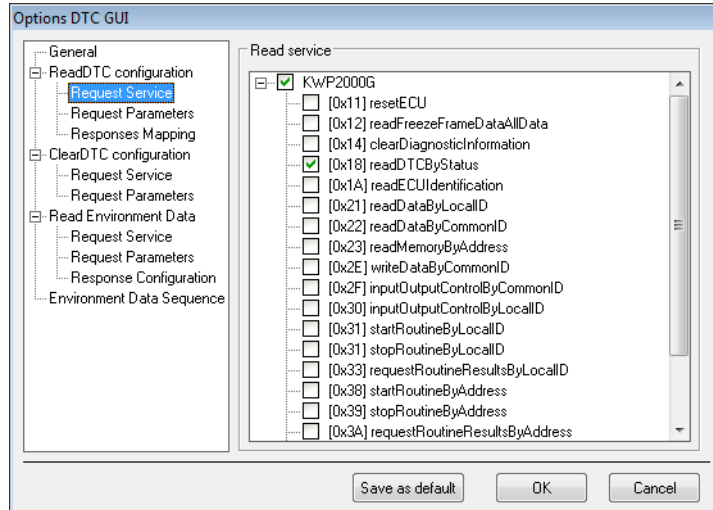
- Choose **ODX** → **User views** → **DiagTroubleCode**.

### To configure the service identification

- Click **Configure**.
- The Configuration dialog window will be displayed.
- Select “Request Service” for the function you want to configure in the left-hand window.



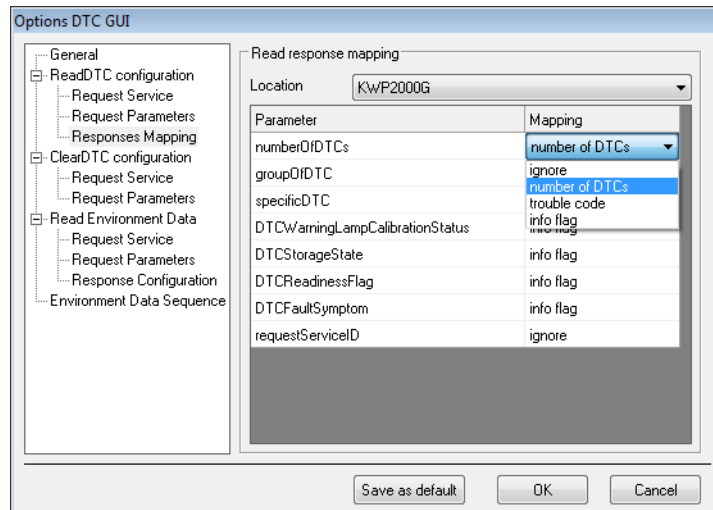
- In the “Services” group, select the service identification you want to use for reading or clearing the fault memory.



- Click **OK**.

**To configure the display for the trouble-log entries**

- Click **Configure**.
- The Configuration dialog window will be displayed.
- Select “Responses Mapping”.
- In the “Mapping” column, select the display mapping for each parameter.



- Click **OK**.

In this lesson, you have configured the "Diagnostic Services", "ECU Identification", and "Diagnostic Trouble Code" functions.

## 8.6 Using the OBD Protocol with ODX-LINK

---

In this lesson, you will configure INCA for using the OBD protocol and will use the ODX-LINK OBD database for OBD diagnostic communication with a real ECU.

### **Note**

*You can only do this lesson if you have connected one of the hardware components supported by ODX-LINK (e.g. ES690/ES590/ES591, ES580 (CAN-LINK), ES581 (CAN Bus Interface USB Module), ES6510 (Vehicle Interface Module) or ES1222) and an ECU.*

For this lesson, you need an ECU which supports OBD communication in accordance with ISO15765-4 on CAN or ISO15031-5/SAE J1979.

### **To configure INCA for OBDonCAN**

---

- Start INCA.
- Create a new top folder with **Edit** → **Add** → **Add top folder**.
- Enter "OBDD Tutorial" as top folder name and press ENTER.
- Add a new workspace using **Edit** → **Add** → **Workspace**.
- Enter "OBDDLesson" as the name and press ENTER.

### **To add an ODX project for OBD**

---

- Change to the "INCA" window and select "OBDD Tutorial".
- Select **Edit** → **Add** → **ODX Project**.
- A file selector window opens for selecting the project or ODX/PDX file(s).
- Navigate to the  
C:\ETASData\ODX1\_5\_0\Projects\OBDonCAN\_ETAS\ODX folder.
- Select the file OBDonCAN\_ETAS.pdx.
- Click **Open**.  
The ODX project is added.

### **To configure hardware**

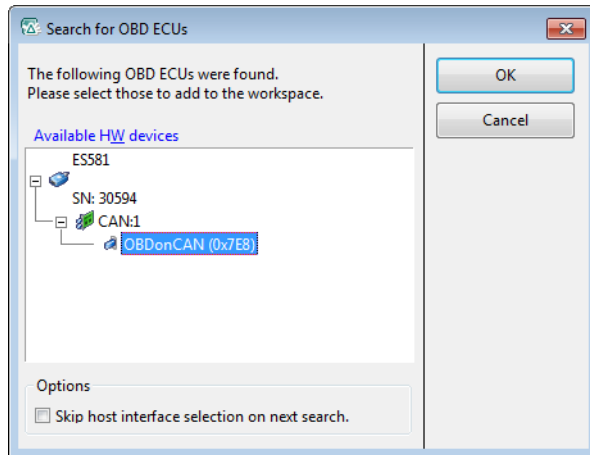
---

- Select the "OBDDLesson" workspace.
- Select **Device** → **Configure Hardware**.  
The hardware configuration window opens.
- Select **Hardware** → **Search for OBD ECUs**.

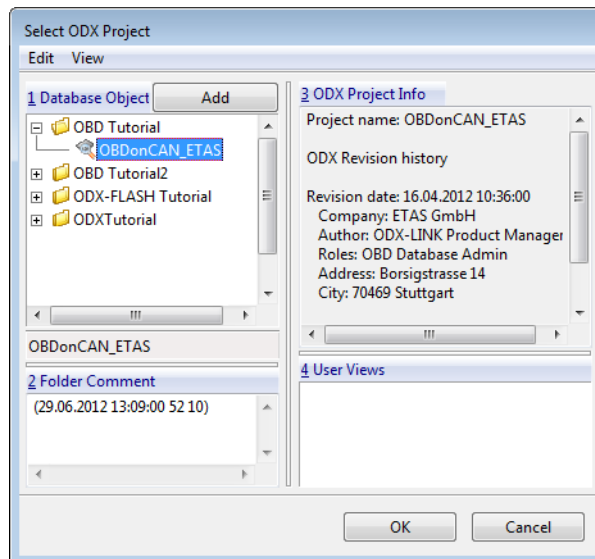
- From the list, select the interface types to be searched for and click **OK**.

A search takes place for connected hardware with OBDOnCAN support.

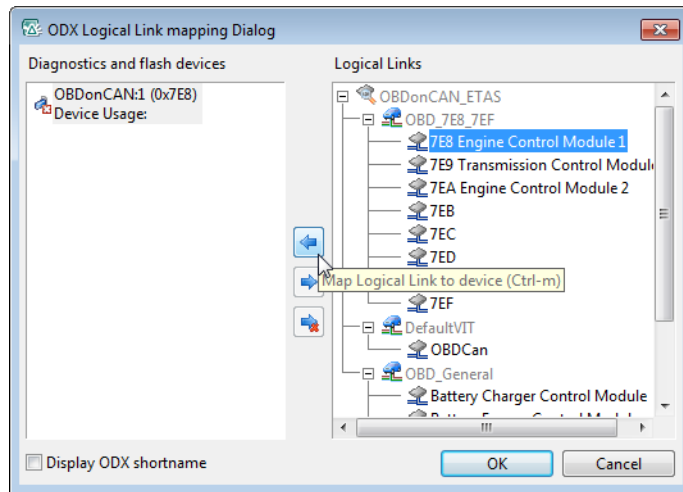
The result is shown in the "Search for OBD ECUs" window and can be selected there.



- Select an ECU found and click **OK**.
- Select the corresponding ODX project from the following window and click **OK**.

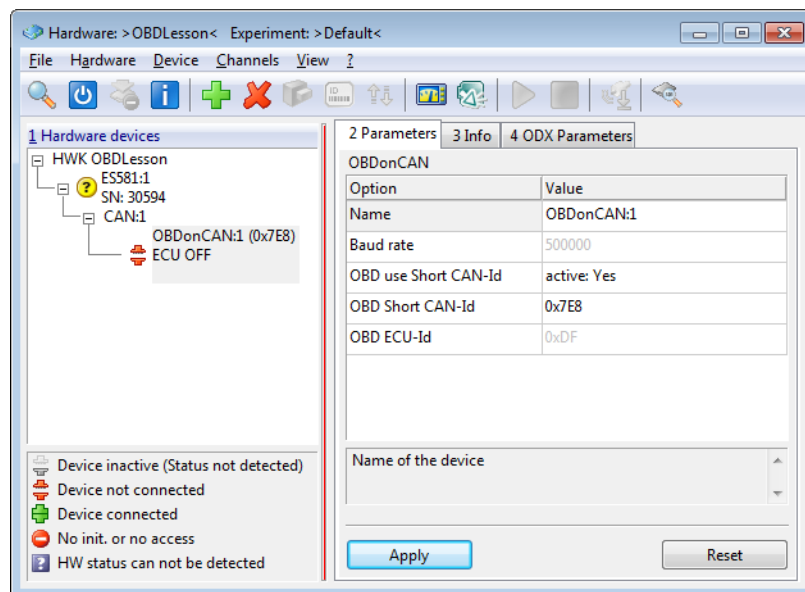


The “ODX Logical Link mapping Dialog” window opens.



- Select the required logical link and assign it to the diagnostic device by clicking the arrow pointing left.
- Click **OK**.

Parameters such as CAN-ID (in brackets beside the device name) and baud rate are generated and configured automatically.



- Select **Hardware** → **Initialize hardware** to test the parameters set.  
Once successfully initialized, hardware configuration is completed.

#### **Note**

*Like every INCA device, an "OBDonCAN" device can only communicate with one ECU at a time. If you need to communicate with several ECUs simultaneously, you have to create and configure an individual OBD device for each one.*

- Close the "Hardware >OBDLesson<" window.

#### **To add an experiment**

---

- Select **Edit** → **Add** → **Experiment**.
- Enter "OBDEXperiment" and press ENTER.
- Double-click "OBDEXperiment".  
A dialog box for selecting the workspace opens.
- Select "OBDLesson" and click **OK**.  
The INCA experiment window opens.

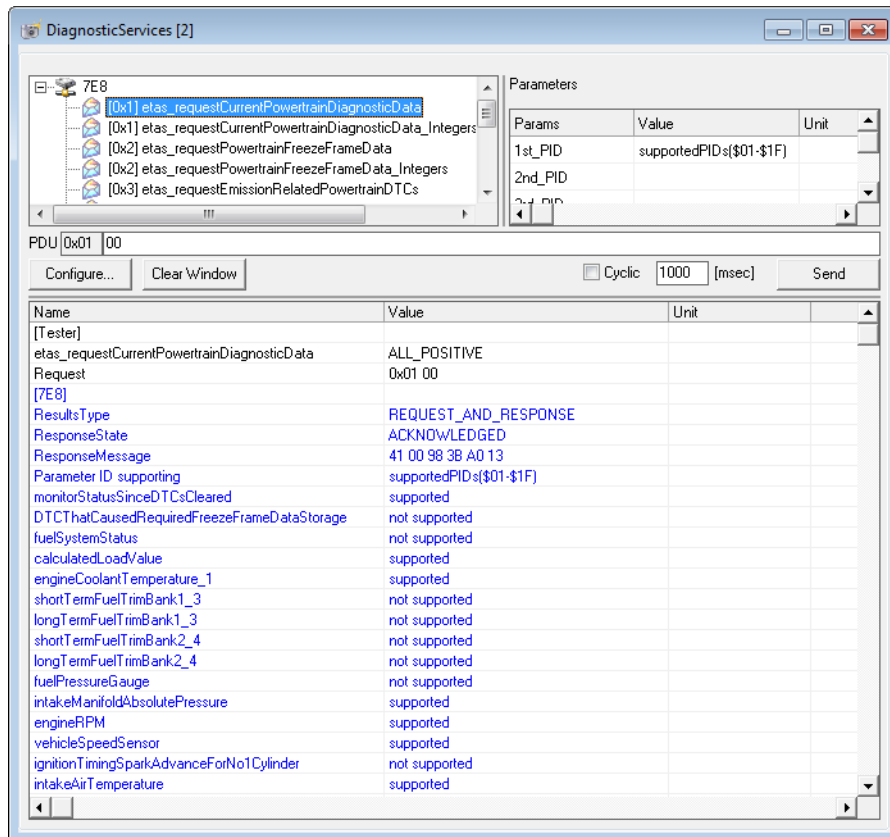
#### **To send OBD Modes/Service requests with ODX-LINK**

---

- Select **ODX** → **User views** → **DiagnosticServices**.  
The diagnostic services dialog box containing all the OBD services is displayed.
- Click **Configure...**  
The "Options DiagnosticService GUI" window for configuring the OBD service IDs display opens.
- Check the "Show service ID in tree view" check box and click **OK**.  
The diagnostic services dialog box displays the service IDs (OBD modes).
- Select the service "[0x1] etas\_requestCurrentPowertrainDiagnosticData".

- Select the PID “supportedPIDs(\$01-\$1F)” and click **Send**.

The service is sent and the response of the ECU shown in the output field.



**Note**

Your ECU may only support some of the possible PIDs and OBD services. If you select a PID or service which is not supported, the ECU may not respond at all resulting in a timeout error message in the INCA Monitor window.

**Note**

ODX-LINK makes ODX configurations available for all supported standard protocols (KWP2000, UDS, KWP2000 on CAN and OBD on CAN) as part of INCA export files in the ETASData\INCA7.1\Export\ODX folder. The ODX-LINK window configurations contained correspond to the relevant diagnostic databases and can be used immediately.

## 8.7 Working with the "OBD" User View

---

In this lesson, you will work with the OBD user view.

You will use the results from lessons "Creating an INCA Workspace" on page 134 and "Preparing an INCA Experiment for ODX without Hardware Connection" on page 139 for this purpose, but have to use a different ODX project (see "The "OBD" User View" on page 69).

### To add an OBD-ODX project

---

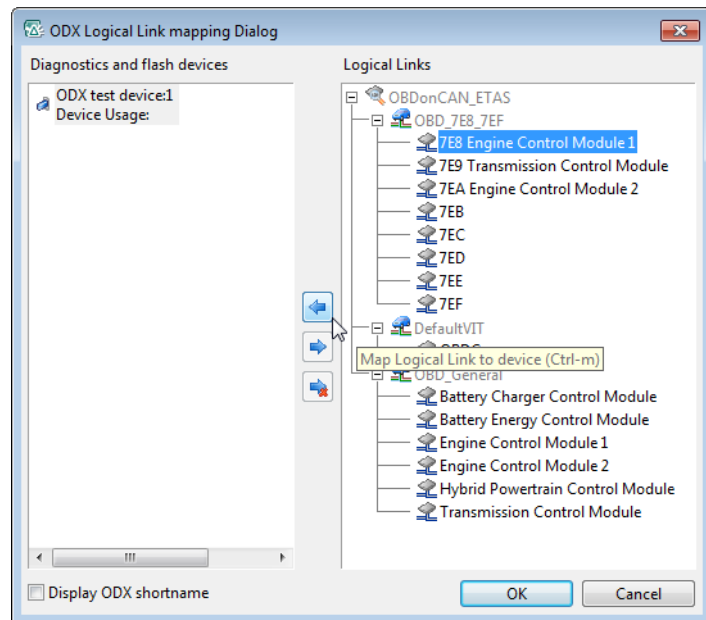
- Change to INCA.
- Open the "ODXTutorial" folder.
- Select **Edit** → **Add** → **ODX Project**.  
A file selector window opens for selecting the project or ODX/PDX file(s).
- Navigate to the  
C:\ETASData\ODX1\_5\_0\Projects\  
OBDonCAN\_ETAS\ODX folder.
- Select the file OBDonCAN\_ETAS.pdx.
- Click **Open**.  
The ODX project is added.

### To add an ODX configuration

---

- Select the "Lesson1" workspace.
- Select **Device** → **Configure hardware**.  
The hardware configuration window opens.
- Select **Hardware** → **Configure ODX**.  
The window for selecting the ODX project opens.
- Select the ODX project "OBDonCAN\_ETAS" and click **OK**.  
The logical link mapping window opens.

- Select the logical link "7E8\_7EF7E8 Engine Control Module1" and add it to the ODX test device by clicking on the button with the arrow pointing left.

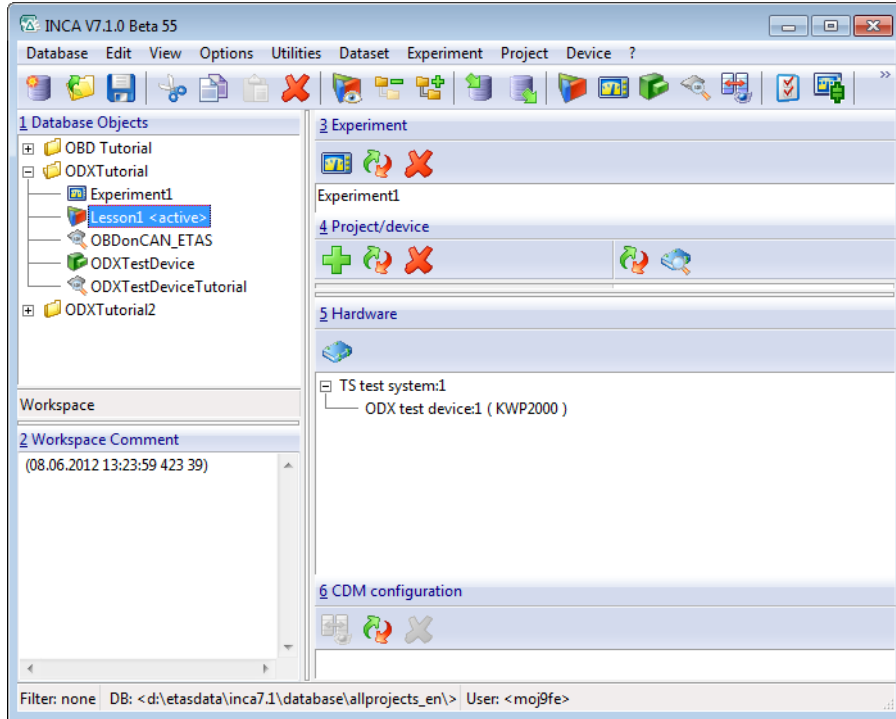


- Click **OK**.
- Close the hardware configuration window.
- Select **Database** → **Save** from the main INCA window.



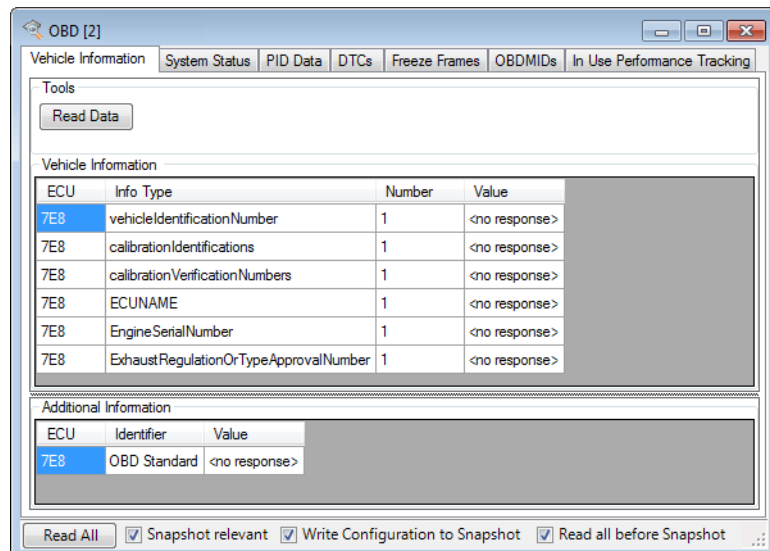
**To query vehicle information and DTCs**

- Double-click the workspace Lesson1.



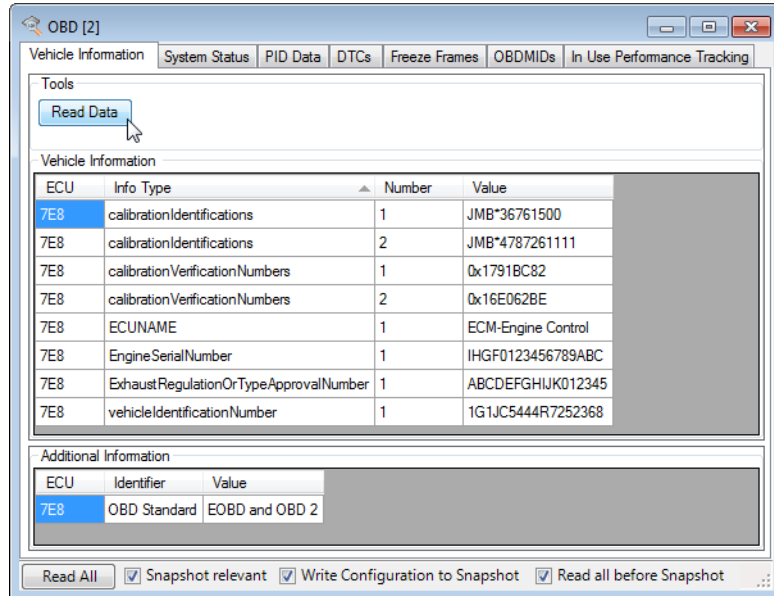
The experiment is opened.

- Select **ODX** → **User views** → **OBD**.  
The "OBD" user view opens.
- Select the "Vehicle Information" tab.



- Click **Read Data**.

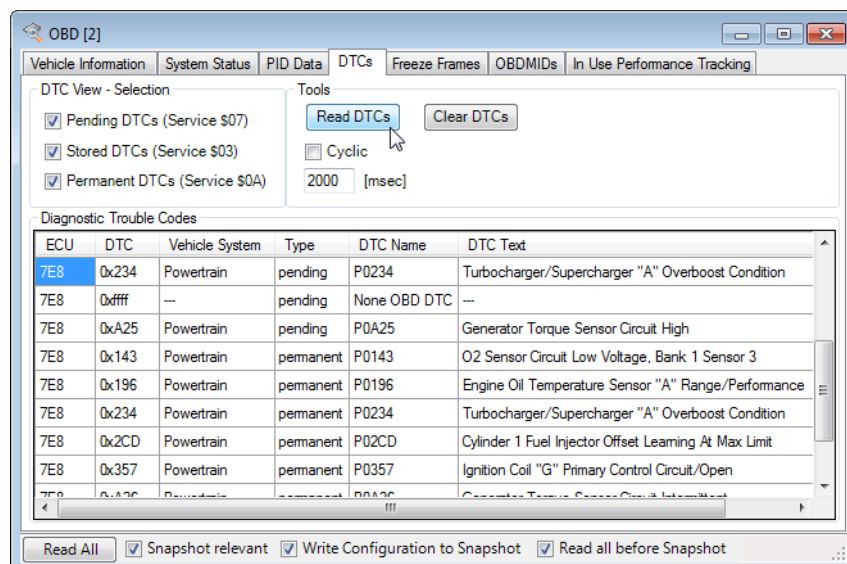
The information is queried and displayed in the "Vehicle Information" field.



In the "Additional Information" field, additional information on the OBD requirements of the vehicle (PID \$1C) are displayed.

- Select the "DTCs" tab.
- Click **Read DTCs**.

The DTCs are displayed in the "Diagnostic Trouble Codes" field.



## 8.8 Using Diagnostic Signal in the Experiment

In this lesson you will learn how diagnostic signals can be used in an experiment.

The diagnostic signals are selected in the variable selection dialog and are indicated in the experiment in measuring windows; a measurement is recorded and viewed in the MDA. For this, use the results from Lesson "Working with the "OBD" User View" on page 159.

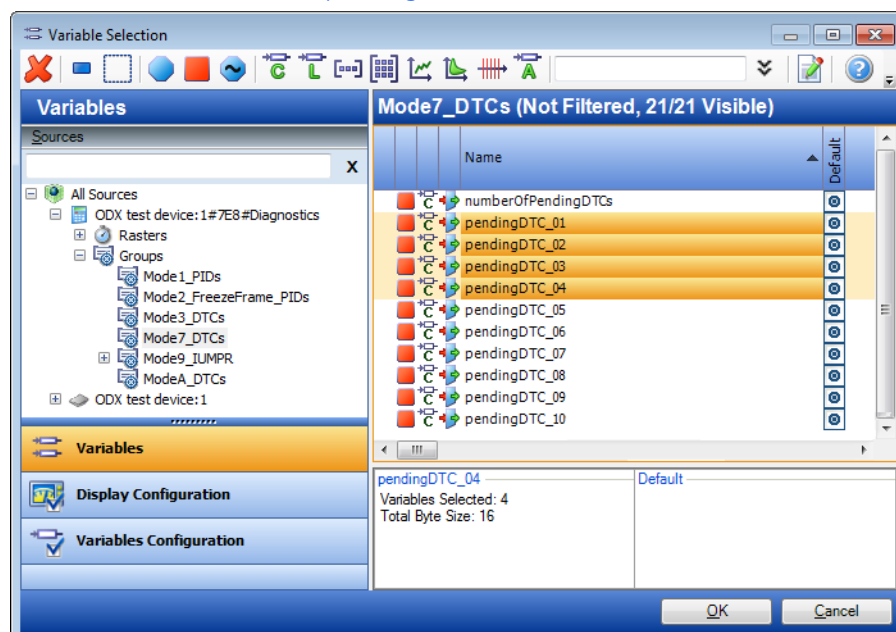
### Opening the INCA Experiment

- Open the folder "ODXTutorial".
- Double-click the workspace "Lesson1".

The INCA experiment is opened. In the next step select the diagnostic signals in the variable selection dialog and assign them to different measuring windows.

### Selecting diagnostic signals in the variable selection dialog

- Select the **Variables** → **Variable Selection**.  
The variable selection dialog is opened.
- Select the source *ODX test device:1#7E8#Diagnostics* and open the tree structure.
- Select under *Groups* the group *Mode7\_DTCs*.  
The diagnostic signals of the group *Mode7\_DTCs* will be displayed.
- Select the diagnostic signals *pendingDTC\_01-pendingDTC\_04*.



- Select in the context menu **Add to** → **Layer\_1** → **New** → **Measure Table**.
- Click **OK**.

The diagnostic signals are displayed in the experiment in a measuring table.

### Indicating diagnostic signals

---



- Click the **Start Visualization** symbol.

The diagnostic signals are displayed in the measuring table.



- Click the **Stop Measuring** symbol.

The display will be stopped. In the next step, the diagnostic signals are recorded in the experiment.

### Note

*The indicated values of the diagnostic signals are simulated by the use of the ODX-Test-Device and, hence, remain constant.*

### Recording diagnostic signals

---



- Click the **Start Recording** symbol.

The measurement will be recorded with a standard recorder.

- Wait for approx. 10 seconds.



- Click the **Stop Measuring** symbol.  
The dialog window **Output File Properties** is opened.

Output File Properties

**DefaultRecorder**

File

Path: d:\ETASData\VNCA7.1\Measure\

File: measure04.dat

Conversion: None

Comments

Insert default comment

Comment: Date: 09/17/2012  
Time: 05:30:53 PM  
Recording Duration: 00:00:14  
Database: DB  
Experiment: Experiment1  
Workspace: Lesson1  
Devices: ODX test device:1,ODX test device:1#7E8#Diagnostics  
Program Description: ODXTestDevice  
WP: Unknown  
RP: Unknown

User: user

Company:

Project:

Vehicle:

Write information to recorder configuration

Save Discard

- Click **Save**.  
The measurement file is stored.

## 8.9 Measuring OBD Data on the Vehicle

---

In this lesson, you measure diagnostic signals on the vehicle and generate additional ECU-specific measurement signals.

### To create a main directory and workspace

---

- Create a main directory called "OBD Tutorial2".
- Create an "OBDLesson2" workspace in this directory.

### To import the ODX project

---

- Add a diagnostic project  
(\ETASData\ODX1\_5\_0\Projects\OBDonCAN\_ETAS\ODX\OBDonCAN\_ETAS.pdx)

### To configure the hardware

---

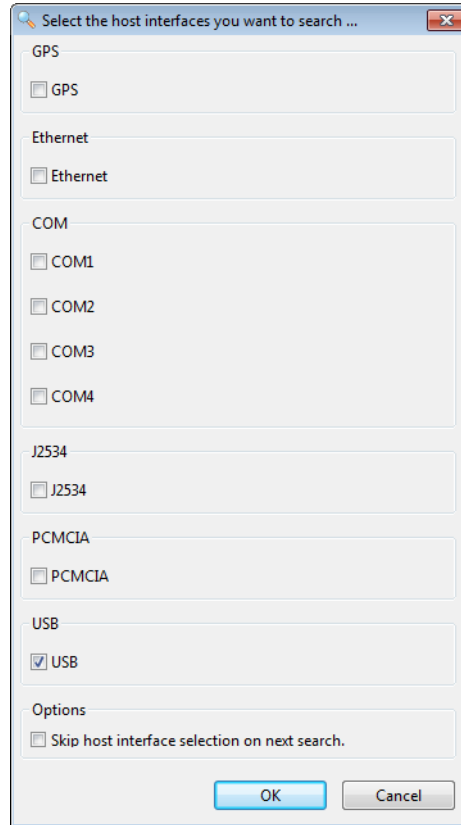
- Select the "OBDLesson2" workspace.
- Select **Device** → **Configure hardware**.  
The Hardware Configuration Editor opens.

#### **Note**

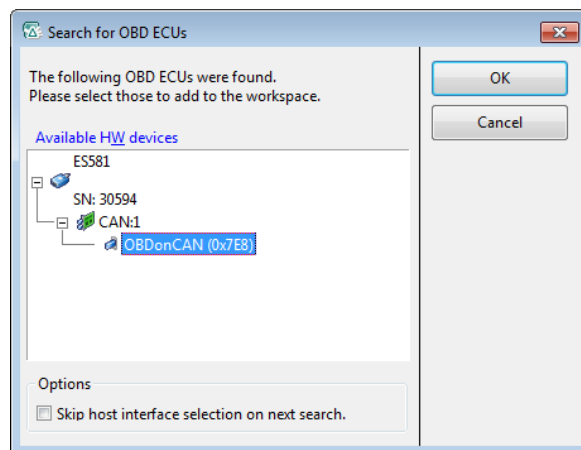
*Make sure that your measurement hardware (ES581, ES590, etc.) is connected to the ECU or vehicle via a CAN port!*

- Select **Hardware** → **Search for OBD ECUs**.

- In the following dialog box, select the interfaces where you want to search for hardware (e.g. Ethernet, USB etc.).



The list of OBD ECUs found is displayed.



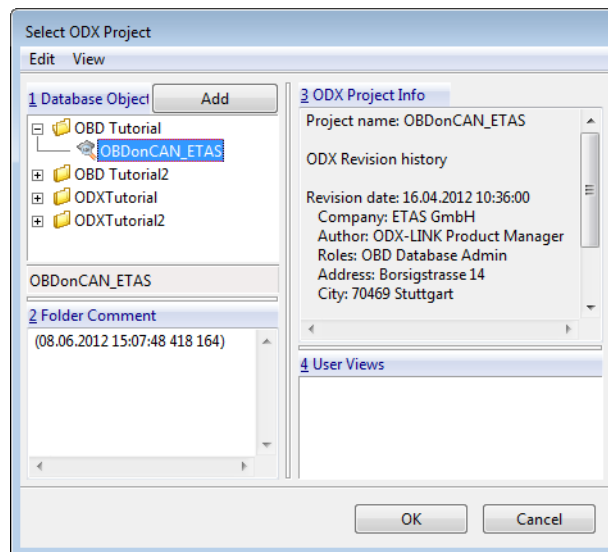
- Select the OBD ECU that you want to use for hardware configuration and click **OK**.

The relevant OBDonCAN devices are created and configured automatically.

#### Note

*If no ECU is found, the ECU or the vehicle either does not support OBDonCAN or is not connected correctly with the hardware.*

- Select the ODX project "OBDonCAN\_ETAS".



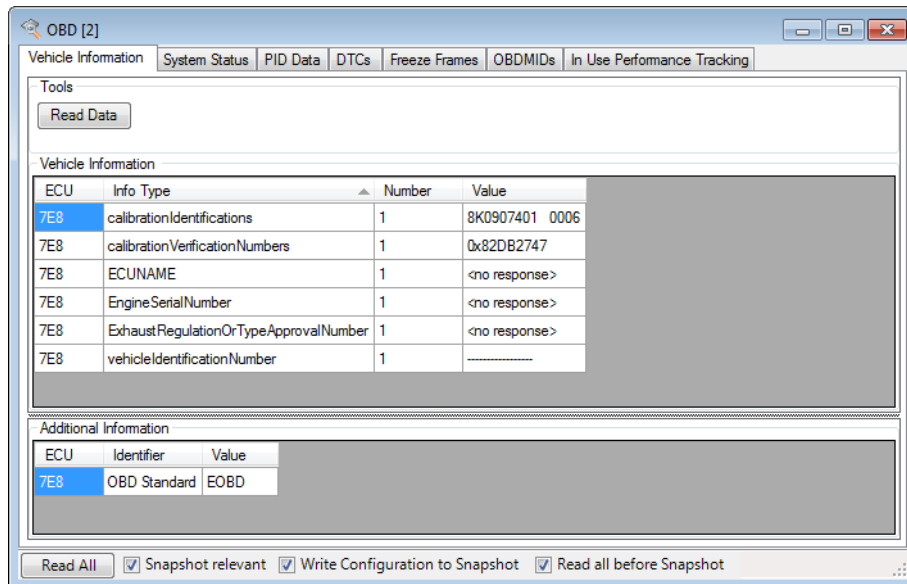
- In the "ODX Logical Link mapping Dialog" assign every OBDonCAN device from the hardware configuration a corresponding logical link from the ODX project (corresponding to the CAN-ID in the device name and in the name of the logical link) and click **OK**.
- Close the Hardware Configuration Editor.

#### To create an experiment and open OBD windows

- Create an experiment "OBDEXperiment2".
- Open the experiment and select the "OBDLesson2" workspace.
- Select **ODX** → **User views** → **OBD**.  
The "OBD" user view opens.



- Click **Read All** to read out all supported OBD data from the connected ECUs once.



- Select **ODX** → **Snapshot**.

or



- Click the snapshot icon.
- Enter the required data and options in the "Data Logging Configuration" window in the tabs "File" and "Header" (see "Data Logging Configuration" on page 117).

- Click **OK**.

The snapshot is taken and opened as a text or HTML file in the relevant editors.

## ODX-LINK Data

Project	
ODX-Project:	OBDonCAN_ETAS
INCA Device Mapping:	OBDonCAN: 1 to 7E8
Date:	2012-06-08
Time:	17:34:05

Author	
Name:	
Department:	
Location:	
Comment:	

Vehicle	
Name:	
Comment:	

---

### 1. Snapshot

Date: 2012-06-08  
Time: 17:34:05

#### 1.1 OBD (OBD [2])

##### 1.1.1 OBD Vehicle Information

Vehicle Information			
ECU	Info Type	Number	Value
7E8	calibrationIdentifications	1	8K0907401 0006
7E8	calibrationVerificationNumbers	1	0x82DB2747
7E8	ECUNAME	1	
7E8	EngineSerialNumber	1	
7E8	ExhaustRegulationOrTypeApprovalNumber	1	
7E8	vehicleIdentificationNumber	1	-----

Additional Information		
ECU	Identifier	Value
7E8	OBD Standard	EOBD

##### 1.1.2 OBD System Status

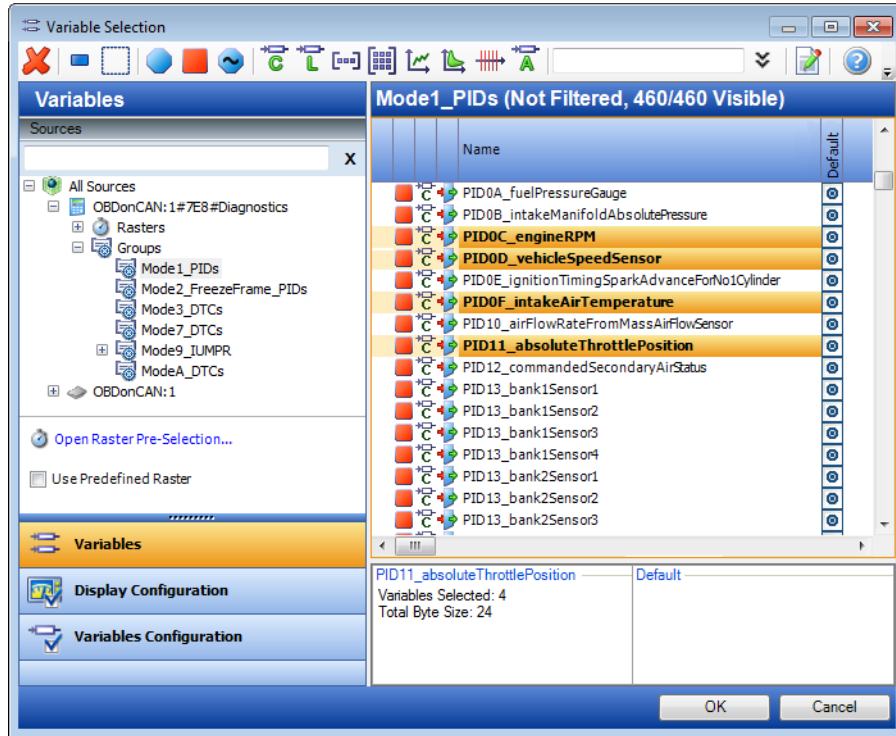
Trouble Codes		
ECU	Parameter	Value
7E8	MILstatus	OFF
7E8	numberOfStoredPowertrainDTCs	0

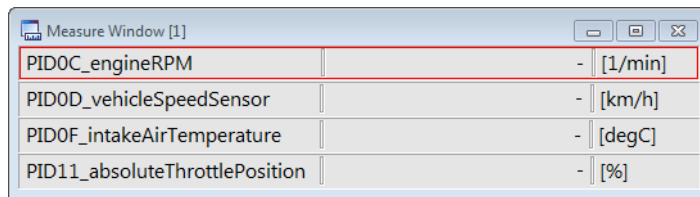
Monitoring Tests				
ECU	Ignition	Monitor	Supported	Completed
7E8	Soark	misfireMonitorina	NO	YES

**To select measurement signals**

- Select **Variables** → **Variable selection**.  
The Variable Selection dialog box opens.
- Select the signals to be measured (e.g. PIDs).



- Close the Variable Selection dialog box with **OK**.  
The selected signals are shown in a measure window.



### To start measuring

---

- Select **Measurement** → **Start visualization**.

The measure values of the selected diagnostic signals are displayed.

Signal Name	Value	Unit
PID0C_engineRPM	0.00	[1/min]
PID0D_vehicleSpeedSensor	254	[km/h]
PID0F_intakeAirTemperature	-40	[degC]
PID11_absoluteThrottlePosition	100.0	[%]

#### **Note**

*If no values are shown for some signals, this might be due to the fact that the ECU does not support the selected OBD signals or the connection has been interrupted.*

- Stop measuring with **Measurement** → **Stop Measuring**.

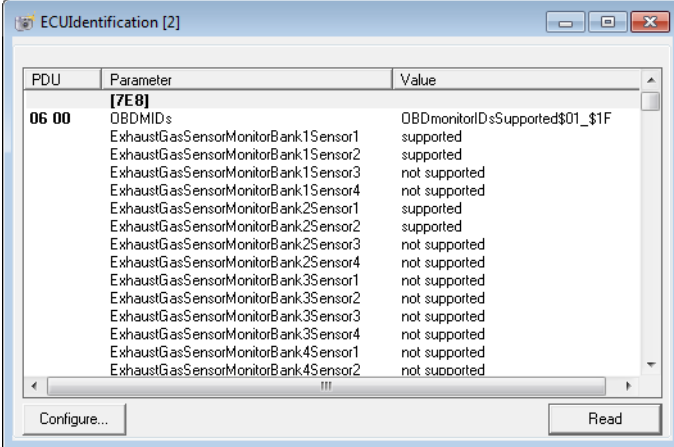
### To generate additional ECU-specific measurement signals for the OBD Mode 6 test results

---

- Select **ODX** → **User views** → **ECUIdentification**.  
The "ECUIdentification" user view opens.
- Click **Configure**.  
The configuration window opens.
- Select "General" and activate the option "Create Measurement Signals from Response Parameters".
- Select "Request Service".
- Activate the service "[0x6] .... 1MID".
- Click **OK**.

- Click **Read**.

All OBD Mode 6 MID's are queried and the results shown.

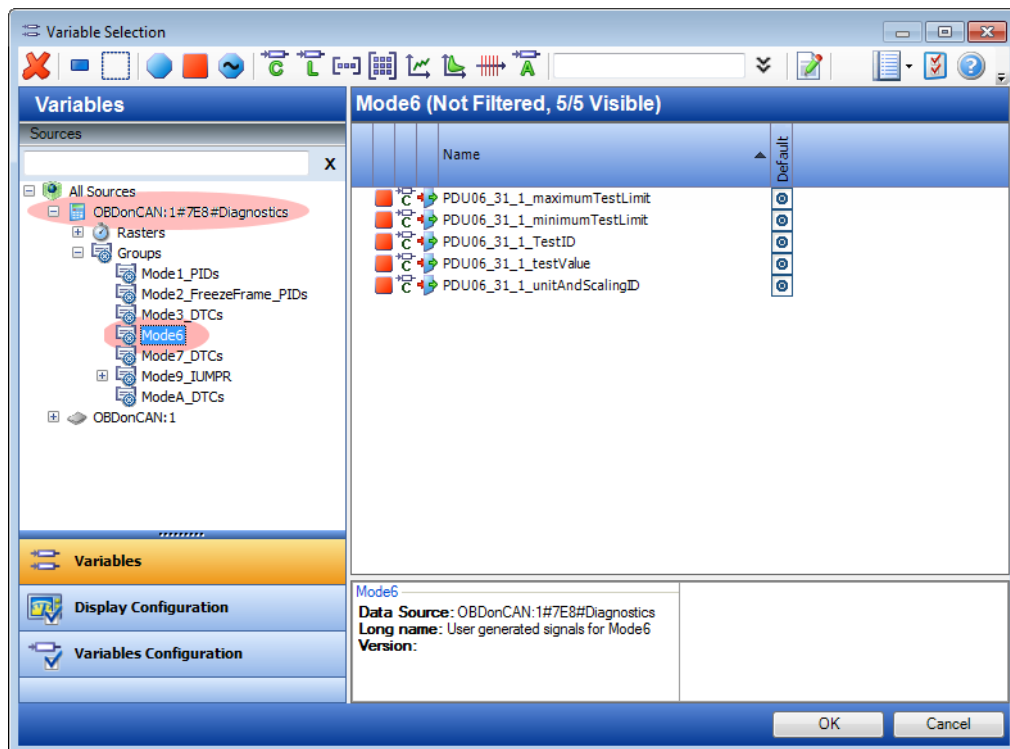


PDU	Parameter	Value
<b>06 00</b>	<b>[7E8]</b>	
	OBDMIDs	OBDMonitorIDsSupported\$01_\$1F
	ExhaustGasSensorMonitorBank1Sensor1	supported
	ExhaustGasSensorMonitorBank1Sensor2	supported
	ExhaustGasSensorMonitorBank1Sensor3	not supported
	ExhaustGasSensorMonitorBank1Sensor4	not supported
	ExhaustGasSensorMonitorBank2Sensor1	supported
	ExhaustGasSensorMonitorBank2Sensor2	supported
	ExhaustGasSensorMonitorBank2Sensor3	not supported
	ExhaustGasSensorMonitorBank2Sensor4	not supported
	ExhaustGasSensorMonitorBank3Sensor1	not supported
	ExhaustGasSensorMonitorBank3Sensor2	not supported
	ExhaustGasSensorMonitorBank3Sensor3	not supported
	ExhaustGasSensorMonitorBank3Sensor4	not supported
	ExhaustGasSensorMonitorBank4Sensor1	not supported
	ExhaustGasSensorMonitorBank4Sensor2	not supported

In the background, a diagnostic signal is created for each of the parameters displayed.

- Close the experiment.  
A note is displayed explaining that the ODX project has changed as new signals have been generated with a query about whether you want to save the changes.
- Confirm with **Ok** or **Yes**.
- Open the experiment again.
- Open the Variable Selection dialog box via **Variables** → **Variable Selection**
- From the "Variables" field, select "ODX test device: 1#7E8#Diagnostics".

- The new signals are in the “Mode6” group and can be selected for measurement there.



## 9 ODX-LINK Troubleshooting

### 9.1 Errors When Adding an ODX Project to the Database

Error Message	Error Description	Remedy
Unable to create ODX project. The MVCI Server is in use.	The applications ODX-FLASH and ODX-LINK, and the adding of an ODX project to the database cannot take place simultaneously.	End ODX-FLASH or ODX-LINK respectively.
ODX file import failed due to one of the following reasons: some ODX files or files referenced by the selected ODX files (e.g. java code) are missing	An error was reported by the MVCI Server during the conversion or verification of the ODX project.	Check the reference files of the project to be imported.
ODX file import failed due to one of the following reasons: the ODX files are not ODX V2.0.1 compliant		Create the ODX project and the relevant ODX files in accordance with the "ODX V2.0.1" specification.
ODX file import failed due to one of the following reasons: the ODX files are inconsistent and violate ASAM ODX checker rules.		Check whether the ODX files conform to the ASAM-ODX rules.
Reinstall ODX Add-on Installation.	Necessary directories or files of the ODX Add-on installation are missing.	Install ODX-Add-on again.
Can't save ODX project to file '*'	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.

Error Message	Error Description	Remedy
The ODX project is inconsistent. It contains no project file (*.prj)	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
	The database object for the ODX project is not valid.	Generate a new database object using the ODX files.
Decompressing failed	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
Can't create temporary directory '*'.	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
Can't remove temporary directory	INCA cannot remove a file from the ETAS temp temporary directory.	
	INCA is still accessing this file.	Shut down INCA, delete the contents of ETAS temp and reboot INCA.
	Another process is still accessing this file.	End all other applications which could be accessing the ETAS temp directory. It may be necessary to reboot your PC.
	You do not have the necessary user rights.	Contact your system administrator.
Can't copy file '*' to '*'	There is too little storage space in the ETAS temp folder.	Shut down INCA, delete the contents of ETAS temp and reboot INCA. It may be necessary to reboot your PC.
	You do not have the necessary user rights.	Contact your system administrator.



Error Message	Error Description	Remedy
Close ODX-LINK	The applications ODX-FLASH and ODX-LINK, and the adding of an ODX project to the database cannot take place simultaneously.	End ODX-FLASH or ODX-LINK respectively.
Error accessing MSCI Server	An error occurred when trying to access the MSCI Server.	Shut down INCA, delete the contents of <code>ETAS temp</code> and reboot INCA. End all other applications which could be accessing the <code>ETAS temp</code> directory. It may be necessary to reboot your PC.
Can't remove the file '*'	INCA cannot remove a file from the <code>ETAS temp</code> temporary directory.	Shut down INCA, delete the contents of <code>ETAS temp</code> and reboot INCA. End all other applications which could be accessing the <code>ETAS temp</code> directory. It may be necessary to reboot your PC.
	INCA is still accessing this file.	
	Another process is still accessing this file.	
Can't load ODX project '*'	You do not have the necessary user rights.	Contact your system administrator.
	An error occurred when adding an ODX project to the INCA database.	Shut down INCA, delete the contents of <code>ETAS temp</code> and reboot INCA. It may be necessary to reboot your PC.
The file * is missing.	A necessary file of the ODX- installation is missing.	Install ODX Add-on again.
The ODX project is inconsistent. It contains no file '*'	There is too little storage space in the <code>ETAS temp</code> folder.	Delete the contents of <code>ETAS temp</code> and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
	The database object for the ODX project is not valid.	Generate a new database object using the ODX files.

Error Message	Error Description	Remedy
'The directory * is missing.	A necessary directory of the ODX Add-on installation is missing.	Install ODX Add-on again.
The ODX project is inconsistent. File '*' is not readable	There is too little storage space in the ETAS temp folder.	Delete the contents of ETAS temp and reboot INCA.
	You do not have the necessary user rights.	Contact your system administrator.
	The database object for the ODX project is not valid.	Generate a new database object using the ODX files.
Can't delete temporary ODX project in file '*'	INCA cannot remove a file from the ETAS temp temporary directory.	
	INCA is still accessing this file.	Shut down INCA, delete the contents of ETAS temp and reboot INCA. It may be necessary to reboot your PC.
	Another process is still accessing this file.	End all other applications which could be accessing the ETAS temp directory. It may be necessary to reboot your PC.
	You do not have the necessary user rights.	Contact your system administrator.

Error Message	Error Description	Remedy
Can't delete temporary file '*'	INCA cannot remove a file from the ETAS temp temporary directory.	
	INCA is still accessing this file.	Shut down INCA, delete the contents of ETAS temp and reboot INCA. It may be necessary to reboot your PC.
	Another process is still accessing this file.	End all other applications which could be accessing the ETAS temp directory. It may be necessary to reboot your PC.
Error in TP_BLOP generation: Sending a Tester Present Message is required for *	You do not have the necessary user rights. The ODX-COM PARAMs define a "tester present" message without parameters. KWPOncAN does not support this.	Contact your system administrator. Set the value of the "CP_TesterPresentHandling" parameter to 1 or set the byte size of the "CP_TesterPresentMessage" parameter to a value > 0.

## 9.2 Error when opening the ODX Project

Error Message	Error Description	Remedy
Diagnostic signal description file error: {Error, Line, Position}	A syntax error is contained in the description file (DSL file).	Exchange the DSL file or have the programmer correct the DSL file.
Failed to open DiagSignalListSchema.xsd.	The file DiagSignalListSchema.xsd was not found.	Check whether the file lies in the correct folder ( <i>ETAS\NCA7.1\ODX</i> ) and is properly named. If the file was not revised, instal ODX anew.
Failed to open VirtualDeviceTemplate.a2l.	The file VirtualDeviceTemplate.a2l was not found.	Check whether the file lies in the correct folder ( <i>ETAS\NCA7.1\ODX</i> ) and is properly named. If the file was not revised, instal ODX anew.
Logical Link '{logical link}' not found inside the diagnostic signal description.	The logical link which you have assigned to a hardware is not described in the DSL file. This means that no diagnostic signal is available to you with this logical link.	Assign the hardware another logical link, if necessary.
No valid request found in DSL file for signal: '{signal name}'	An appropriate request (inquiry) is missing or the referenced request is faulty for a diagnostic signal in the DSL file. A faulty Request is localized by one of the following error messages (see below).	Exchange the DSL file or have the programmer correct the DSL file.
Request: '{request name}' doesn't contain a parameter with name: '{parameter name}'	The Request contains no parameters with the appropriate name in the DSL file.	Exchange the DSL file or have the programmer correct the DSL file.

Error Message	Error Description	Remedy
Request: '{request name}' incorrect: value missing	In the DSL file, the applicable value is missing by a parameter with this Request.	Exchange the DSL file or have the programmer correct the DSL file.
Request: '{request name}' incorrect: short name missing.	In the DSL file, the appropriate short name is missing by a parameter with this Request. .	Exchange the DSL file or have the programmer correct the DSL file.
Request: '{request name}' incorrect: type not 'const'.	In the DSL file, the Request does not have the correct type ('const') for the above signal.	Exchange the DSL file or have the programmer correct the DSL file.
Request: '{request name}' incorrect: type for parameter '{parameter name}' isn't 'variable' or 'field'.	In the DSL file, the Request does not have the correct type ('variable' or 'field') for the above signal .	Exchange the DSL file or have the programmer correct the DSL file.
Request: '{request name}' doesn't contain a valid PDU: '{incorrect PDU}'	The data input in the DSL file for the PDU is missing or is faulty.	Exchange the DSL file or have the programmer correct the DSL file.

### 9.3 Error while starting a measurement

Error Message	Error Description	Remedy
Failed to create diagnostic service for signal: '{signal}'	In the DSL file, a diagnostic service is performed which does not exist in the ODX Project.	Correct the ODX Project. Exchange the DSL file or have the programmer correct the DSL file.
{signal name} getting TrgtSvr-Sink: '{sink name}'	The signal does not receive a Target Server Object.	Start the measurement anew. Load the experiment anew.
Unavailable parameter '{parameter name}' for Request with ID '{request ID}'. Failed to create diagnostic service for signal: '{signal name}'	In the DSL file, a request is performed whose parameter does not exist in the ODX Project.	Correct the ODX Project. Exchange the DSL file or have the programmer correct the DSL file.

## 9.4 Error during the measurement

Error Message	Error Description	Remedy
Service for the following diagnostic signals reported execution state: '{list of signals}' Execution state: '{text}' Description of the error: '{text}' Vendor description of the error: '{text}'	The response of a diagnostic service has not been answered "positively". All signals which could not be measured are listed in the error message. If no signals could be measured, the link to the hardware may not work correctly.	Remove the signals which are not supported by the control device. Check the connection with the hardware and initialize the hardware anew.
No response received for signal: '{signal name}'	In the response of the diagnostic service, no reply is present for the signal. Possibly, the control device does not support all diagnostic signals, which are included in the DSL file.	Remove the signals which are not supported by the control device. Check the connection with the hardware and initialize the hardware anew.
Parameter for the following diagnostic signal has an error: '{signal}' Description of the error: '{text}' Vendor description of the error: '{text}'	In the response, a parameter for the signal is faulty and could not be evaluated. This means that no physical value could be calculated. This could have been caused by a faulty ODX data input or a faulty response from the control device.	Correct the ODX Project or exchange it.

Error Message	Error Description	Remedy
Service for the following diagnostic signals has an error: '{list of signals}' Description of the error: '{text}' Vendor description of the error: '{text}'	In the answer, either a response or a parameter of the response is faulty. This could have been caused by a faulty ODX data input or a faulty response from the control equipment.	Correct the ODX Project or exchange it.
Exception by updating value for signal '{signal name}': '{exception}'	The conversion of a data type of the D-server in a data type, which INCA understands, has failed. This could have been caused by a faulty ODX data input or a faulty answer from the control device.	Correct the ODX Project or exchange it.
{signal name}writing value:'{exception}'	A value of the signal could not be written into the target server.	Start the measurement anew.



## 10 ODX Communication Parameters

---

The communication parameters are used to initialize the devices used in diagnosis or flash programming via ODX and are stored in the `ComparamSpec` section of the ODX project (odx-c).

Based on the existing ODX parameters, an ASAP2 TP\_BLOP is generated which is required to generate the ASAP1b driver.

The following sections explain the ODX parameters supported.

### **Note**

*Unlike ODX-FLASH, ODX-LINK can still work with the parameters from the ASAP2 file.*

*The following rule applies: If an ASAP2 project was assigned to a diagnostic device during hardware configuration, initialization takes place via the parameters from ASAP2 (as was the case so far) – if, however, no ASAP2 file was assigned, initialization takes place via the parameters of the ODX project.*

## 10.1 A2L Structure: TP\_BLOP

This structure is required for the following protocols:

- KWPOnKLine
- KWPOnCAN
- UDSONCAN

A2L Parameter	ODX Parameter	Value Range	Conversion (ODX(x) → A2L(y))	Default Value	Comment
blob version	-	-	-	KWPOnKLine: 0x201; KWPOnCAN: 0x201; UDSONCAN: 0x301	The default value is always used.
Protocol version	-	-	-	KWPOnKLine: VDA_1996; KWPOnCAN: VDA_1996; UDSONCAN: ISO14229_1_2003	The default value is always used.
byte order	-	-	-	KWPOnKLine: MSB_FIRST; KWPOnCAN: MSB_FIRST; UDSONCAN: BYTEORDER_MSB_FIRST	This parameter is only required for measuring/calibration jobs. The default value is used for flash/diagnostic jobs.

## 10.2 A2L Structure: K-Line

This structure is required for the following protocols:

- KWPOnKLine

A2L Parameter	ODX Parameter	Value Range	Conversion (ODX(x) → A2L(y))	Default Value	Comment
stimulation mode	CP_Initialisation Settings	[1; 3]	1 → Stimulation_5Baud; 2 → WuP; 3 → use default	KWPOnKLine: Stimulation_5Baud	
ECU address	CP_EcuRespSourceAddress	[0x0; 0xFF]	y = x	-	
tester address	CP_TesterSourceAddress	[0x0; 0xFF]	y = x	-	

### 10.3 A2L Structure: CAN

This structure is required for the following protocols:

- KWPOnCAN
- UDSONCAN

A2L Parameter	ODX Parameter	Value Range/ Unit	Conversion (ODX(x) → A2L(y))	Default Value	Comment
Baudrate	CP_Baudrate	[0x0; 0xFFFFFFFF] Unit: baud	$y = x$	-	
sample point	CP_BitSamplePoint	[0; 100]	$y = x$	-	
samplecount per bit	CP_SamplesPerBit	[0; 1]	$0 \Rightarrow 1; 1 \Rightarrow 3$	-	
BTL_CYCLES	-	-	Is generated using a "Best Match" algorithm from "CP_BitSamplePoint" and "CP_SyncJumpWidth".	-	
SJW length	CP_SyncJumpWidth	[0; 100]	$y = \text{round}(\text{CP\_SyncJumpWidth} * \text{BTL\_CYCLES} / 100)$	-	
SYNC_EDGE	CP_INCA_SYNC_EDGE	[0; 2]	$y = x$ <sup>1)</sup>	0	The default value is always used.

1) This is an optional INCA-specific parameter. If it is not defined, the value defined in the "Default Value" column is used.

## 10.4 A2L Structure: CAN Address

This structure is required for the following protocols:

- KWPOnCAN
- UDSOnCAN

A2L Parameter	ODX Parameter	Value Range	Conversion (ODX(x) → A2L(y))	Default Value	Comment
CAN_ID ECU	CP_CanRespUSDTId CP_CanRespUSDFormat	[0x0; 0x1FFFFFFF] [0x0; 0x3F]	4)	-	
CAN_ID tester	CP_CanPhysReqId CP_CanPhysReqFormat	[0x0; 0x1FFFFFFF] [0x0; 0x3F]	5)	-	
TGT_ECU	CP_CanRespUSDTEstAddr CP_CanRespUSDFormat	[0x0; 0xFF] [0x0; 0x3F]	6)	-	
TGT_tester	CP_CanPhysReqExtAddr CP_CanPhysReqFormat	[0x0; 0xFF] [0x0; 0x3F]	7)	-	

4)  $y = CP\_CanRespUSDTId \mid (bit\_3(CP\_CanRespUSDFormat) \ll 28)$ ; bit0 is LSB

5)  $y = CP\_CanPhysReqId \mid (bit\_3(CP\_CanPhysReqFormat) \ll 28)$ ; bit0 is LSB

6) if  $(bit\_5\_4(CP\_CanRespUSDFormat) = 0x30)$ : Error: Extended addressing for tester and normal addressing for ECU is not supported  
 if  $(bit\_5\_4(CP\_CanRespUSDFormat) = 0x10 \parallel bit\_5\_4(CP\_CanRespUSDFormat) = 0x20)$ :  $y = CP\_CanRespUSDTEstAddr$   
 otherwise no value is written; bit0 is LSB

7) if  $(bit\_5\_4(CP\_CanPhysReqFormat) = 0x30)$ : Error: Extended addressing for tester and normal addressing ECU is not supported  
 if  $(bit\_5\_4(CP\_CanPhysReqFormat) = 0x10)$ :  $y = CP\_CanPhysReqExtAddr$   
 otherwise no value is written; bit0 is LSB

## 10.5 A2L Structure: CAN TesterPresentOptions

This structure is required for the following protocols:

- KWPOnCAN

A2L Parameter	ODX Parameter	Value Range	Conversion (ODX(x) => A2L(y))	Default Value	Comment
tester present	CP_TesterPresentHandling, CP_TesterPresentReqRsp, CP_TesterPresentMessage	[0; 1] [0; 1] Each byte: [0x0; 0xFF], Byte length: 1 .. 12	See conversion parameters		

### Conversion parameters:

if (CP\_TesterPresentHandling = 0) or (bytesize of CP\_TesterPresentMessage = 0): 3 <sup>1</sup>

if (bytesize of CP\_TesterPresentMessage = 1): TesterPresent\_WithoutParameter

if (bytesize of CP\_TesterPresentMessage > 1 and CP\_TesterPresentReqRsp = 0):  
TesterPresent\_WithParameter\_NoResponseRequired

if (bytesize of CP\_TesterPresentMessage > 1 and CP\_TesterPresentReqRsp = 1):  
TesterPresent\_WithParameter\_ResponseRequired

<sup>1</sup> This does not conform to KWP2000 AML but is supported by the ETAS ASAP1b-driver

## 10.6 A2L Structure: SESSION TesterPresentOptions

This structure is required for the following protocols:

- UDSONCAN

A2L Parameter	ODX Parameter	Value Range	Conversion (ODX(x) => A2L(y))	Default Value	Comment
tester present	CP_TesterPresentHandling, CP_TesterPresentReqRsp, CP_TesterPresentMessage	[0; 1] [0; 1] Each byte: [0x0;0xFF], Byte length: 1 .. 12	See conversion parameters		

### Conversion parameters:

if (byte size of CP\_TesterPresentMessage = 1): Error: "Sending a Tester Present Message without parameters is not allowed"

if (CP\_TesterPresentHandling = 0 or byte size of CP\_TesterPresentMessage = 0): NoTesterPresent

if (CP\_TesterPresentHandling != 0 and byte size of CP\_TesterPresentMessage != 0 and CP\_TesterPresentReqRsp = 0):  
TesterPresent\_WithParameter\_NoResponseRequired

if (CP\_TesterPresentHandling != 0 and byte size of CP\_TesterPresentMessage != 0 and CP\_TesterPresentReqRsp = 1):  
TesterPresent\_WithParameter\_ResponseRequired

## 10.7 A2L Structure: CAN NETWORK\_LIMITS

---

This structure is required for the following protocols:

- KWPOnCAN
- UDSONCAN

A2L Parameter	ODX Parameter	Value Range	Conversion (ODX(x) => A2L(y))	Default Value	Comment
WFT_MAX	CP_CanMaxNumWaitFrames	[0; 1027]	$y = x$	-	
XDL_MAX	CP_INCA_XDL_MAX	[0x0; 0xFFFFFFFF]	$y = x$ <sup>1)</sup>	500	The default value is always used.

1) This is an optional, INCA-specific parameter. If it is not defined, the value defined in the "Default Value" column is used.



## 10.8 A2L Structure: DIAG\_BAUD

This structure is required for the following protocols:

- KWPOnKLine
- KWPOnCAN

A2L Parameter	ODX Parameter	Value Range/Unit	Conversion (ODX(x) => A2L(y))	Default Value	Comment
Baudrate	CP_INCA_DIAG_BAUD _Baudrate	[0x0; 0xFFFFFFFF] Unit: baud	$y=x$ <sup>1), 2)</sup>	10400 (default baud rate)	The default value is always used.
diagnostic mode	CP_INCA_DIAG_BAUD _DiagnocsticMode	[0;0xFF]	$y=x$ <sup>1)</sup>	0x86 (= default session)	The default value is always used.
BD_PARA	CP_INCA_DIAG_BAUD _BD_PARA	Each byte: [0x0;0xFF], Byte length:1 .. 12	$y=x$ <sup>1)</sup>	0x06 0x00 0x28 0xA0	The default value is always used.

1) This is an optional, INCA-specific parameter. If it is not defined, the value defined in the "Default Value" column is used.  
2) With this parameter, it is assumed that "baud" is the unit used. No deviating unit can be defined in ODX.

## 10.9 A2L Structure: TIME\_DEF KWP\_TIMING

This structure is required for the following protocols:

- KWPOnKLine
- UDSONCAN

A2L Parameter	ODX Parameter	Value Range/Unit	Conversion (ODX(x) => A2L(y))	Default Value	Comment
p1	CP_P1Max	[0; 250000] Unit: $\mu$ s	$y = \text{round}(x/1000)$ <sup>3)</sup>	-	
p2Min	CP_P2Min	[0; 250000] Unit: $\mu$ s		-	
p2Max	CP_P2Max	[0; 125000000] Unit: $\mu$ s		-	
p3Min	CP_P3Min	[0; 250000] Unit: $\mu$ s		-	
p3Max	CP_P3Max_Ecu	[0; 100000000] Unit: $\mu$ s		-	
p4	CP_P4Min	[0; 250000] Unit: $\mu$ s		-	

3) With this parameter, it is assumed that " $\mu$ s" is the unit used. No deviating unit can be defined in ODX.

## 10.10 A2L Structure: TIME\_DEF USDTP\_TIMING

This structure is required for the following protocols:

- KWPOnCAN

A2L Parameter	ODX Parameter	Value Range/Unit	Conversion (ODX(x) => A2L(y))	Default Value	Comment
As	CP_As	[0; 20000000] Unit: $\mu$ s	$y = \text{round}(x/1000)$ <sup>3)</sup>	-	
Bs	CP_Bs	[0; 20000000] Unit: $\mu$ s		-	
Cr	CP_Cr	[0; 20000000] Unit: $\mu$ s		-	

3) With this parameter, it is assumed that “ $\mu$ s” is the unit used. No deviating unit can be defined in ODX.

## 10.11 A2L Structure: USDTP\_TIMING\_DEFAULTS

This structure is required for the following protocols:

- UDSOnCAN

A2L Parameter	ODX Parameter	Value Range/Unit	Conversion (ODX(x) => A2L(y))	Default Value	Comment
As	CP_As	[0; 20000000] Unit: $\mu$ s	$y = \text{round}(x/1000)^3$	-	
Bs	CP_Bs	[0; 20000000] Unit: $\mu$ s		-	
Cr	CP_Cr	[0; 20000000] Unit: $\mu$ s		-	
p2Min	CP_P2Min	[0; 250000] Unit: $\mu$ s		-	
p2Max	CP_P2Max	[0; 125000000] Unit: $\mu$ s		-	
p3Min	CP_P3Min	[0; 250000] Unit: $\mu$ s		-	
p3Max	CP_P3Max_Ecu	[0; 100000000] Unit: $\mu$ s		-	
3) With this parameter, it is assumed that “ $\mu$ s” is the unit used. No deviating unit can be defined in ODX.					

## 10.12 A2L Structure: SESSION

This structure is required for the following protocols:

- UDSONCAN

A2L Parameter	ODX Parameter	Value Range	Conversion (ODX(x) => A2L(y))	Default Value	Comment
session identifier	CP_INCA_UDSSessionIdentifier	char length: 1 .. 99	<sup>1)</sup>	"UDSSession"	The default value is always used.
diagnostic mode	CP_INCA_USDSDiagnosticMode	[0;0xFF]	y=x <sup>1)</sup>	0x01 (= default session)	The default value is always used.

<sup>1)</sup> This is an optional, INCA-specific parameter. If it is not defined, the value defined in the "Default Value" column is used.

### 10.13 A2L Structure: ADDRESS\_AND\_LENGTH\_FORMAT\_IDENTIFIER

This structure is required for the following protocols:

- UDSOnCAN

A2L Parameter	ODX Parameter	Value Range	Conversion (ODX(x) => A2L(y))	Default Value	Comment
AALFI general setting	-	-	-	0x13	These parameters exist for reasons of compatibility to TP_BLOP and should not be used by the ASAP1b driver for diagnostic/flash jobs.
AALFI_FOR_CHECKSUM_CALCULATION	-	-	-	0x13	
AALFI_FOR_ERASE_MEMORY	-	-	-	0x13	
AALFI_FOR_WRITE_MEMORY_BY_ADDR	-	-	-	0x13	
AALFI_FOR_READ_MEMORY_BY_ADDR	-	-	-	0x13	
AALFI_FOR_DYNAMICALLY_DEFINE_DATA_ID	-	-	-	0x13	
AALFI_FOR_REQUEST_DOWNLOAD	-	-	-	0x13	

### 10.14 A2L Structure: SESSION SessionOpeningOrder

---

This structure is required for the following protocols:

- UDSONCAN

A2L Parameter	ODX Parameter	Value Range	Conversion (ODX(x) => A2L(y))	Default Value	Comment
session opening order	CP_INCA_ UDSSessionOpeningOrder	[0; 2]	1)	2	The default value is always used.

1) This is an optional, INCA-specific parameter. If it is not defined, the value defined in the "Default Value" column is used.

### 10.15 A2L Structure: CAN Transport Protocol Version

---

This structure is required for the following protocols:

- UDSONCAN

A2L Parameter	ODX Parameter	Value Range	Conversion (ODX(x) => A2L(y))	Default Value	Comment
transport protocol version	-	-	-	ISO15765_2_2003	The default value is always used.

## 10.16 Parameters that can be changed by the flash job

---

### CP\_RC78 Handling.

This parameter is supported by the following protocols:

- UDSONCAN

ODX Parameter	Value Range	Description	Default Value	Comment
CP_RC78Handling	{0,2}	0 = Disabled 2 = Continue handling unlimited (until disabled)	2	-

### CP\_StMinOverride:

Dieser Parameter wird für folgende Protokolle unterstützt:

- UDSONCAN

ODX Parameter	Value Range	Description	Default Value	Comment
CP_StMinOverride	{100,1000000} Unit: µs	This parameter sets the separation time INCA should use to transmit segmented CAN messages to the ECU. If this parameter is set in the Java Flash Job, the separation time "STmin" transmitted in the flow control frame is ignored and "CP_StMinOverride" will be used instead.	-	-



## 11 Glossary

---

This chapter gives you definitions and explanations of terms that are used in connection with ODX-LINK V1.5.

### Diagnostic database

The diagnostic database contains all services, their parameters and the possible ECU responses in hexadecimal and plain text notation. The diagnostic database is stored in a file with the ".dat" extension.

### Communication path

A communication path is the combination of ECU, communication interface (CAN, K-Line and others) and communication parameters. The communication path is also referred to as the "ODX Database Location."

### ODX project

An ODX project combines all components required for communicating and interpreting data. This includes, for example, the diagnostic database and interface definition.

### ODX project file

The ODX project file contains the following information:

- A pointer to the database with the services that have been defined for your ECU. This database contains all the services, their parameters and the possible ECU responses in hexadecimal and plain text notation.
- The hardware interface configuration
- The definitions for the links between logical and physical interfaces

The ODX project file has the ".prj" file extension.

### PDU

Protocol Data Unit. The data transmitted to the ECU after the service ID. The PDUs specify the parameters and the associated values for each service.



## 12 ETAS Contact Addresses

---

### *ETAS HQ*

---

ETAS GmbH

Borsigstraße 14

70469 Stuttgart

Germany

Phone: +49 711 89661-0

Fax: +49 711 89661-106

WWW: [www.etas.com](http://www.etas.com)

### *ETAS Subsidiaries and Technical Support*

---

For details of your local sales office as well as your local technical support team and product hotlines, take a look at the ETAS website:

ETAS subsidiaries WWW: [www.etas.com/en/contact.php](http://www.etas.com/en/contact.php)

ETAS technical support WWW: [www.etas.com/en/hotlines.php](http://www.etas.com/en/hotlines.php)



---

## Figures

<b>Fig. 3-1</b>	ISO Standards and ASAM Specifications.....	23
<b>Fig. 7-1</b>	The Tabs of the "OBD" User View.....	101
<b>Fig. 7-2</b>	"OBD" User View – "Vehicle Information" Tab .....	104
<b>Fig. 7-3</b>	"OBD" User View – "System Status" Tab.....	105
<b>Fig. 7-4</b>	"OBD" User View – "PID Data" Tab.....	106
<b>Fig. 7-5</b>	"OBD" User View – "PID Data" Tab in Configuration Mode.....	107
<b>Fig. 7-6</b>	"OBD" User View – "DTCs" Tab .....	109
<b>Fig. 7-7</b>	"OBD" User View – "Freeze Frames" Tab .....	110
<b>Fig. 7-8</b>	"OBD" User View – "Freeze Frames" Tab in Configuration Mode .....	111
<b>Fig. 7-9</b>	"OBD" User View – "OBDMIDS" Tab.....	113
<b>Fig. 7-10</b>	"OBD" User View – "OBDMIDS" Tab in Configuration Mode.....	114
<b>Fig. 7-11</b>	"OBD" User View – "In Use Performance Tracking" Tab .....	116
<b>Fig. 7-12</b>	Data Logging Configuration - "File" Tab .....	118
<b>Fig. 7-13</b>	Data Logging Configuration - "Header" Tab .....	120
<b>Fig. 7-14</b>	User View Dialog Box with Snapshot Icon .....	122



---

## Index

### A

Append date and time 118  
Append increasing number 118  
Append output to log file 118

### C

Comment 120  
Communication path 201  
Composition of file name 118  
Configuration  
    default 69  
Create additional XML file 119  
Create file name automatically 118

### D

Data Logging Configuration 117  
Default configuration 69  
Department 120  
Diagnostic database 201  
Diagnostics 21  
    without A2L file 27  
Diagnostics Signal List 125  
Directory 118

### E

Environment data recording 81  
ETAS Contact Addresses 203  
ETAS license models 13

### F

File name 118  
Flash ID 38

### H

Hide environmental data display 81  
Hide request 81

### I

Information 120  
Installation 11

### J

Java Jobs 75

### L

License  
    borrowing 18  
    expiration warning 17  
    grace mode 16  
    license file 15  
    license models 13  
Logging location 120

### M

Manual  
    conventions 9  
    representation of information 8  
    structure of the ~ 7  
Memory Dump 87

**N**

Name 120

**O**

OBDonCAN devices

    automatic search 28

ODX Flash ID 38

ODX project 201

ODX project file 201

ODX standard 22

Open ASCII file after creation 119

Open XML file after creation 119

**P**

PDU 201

Project file 201

**R**

Read environment data 81

**S**

Show message bytes 73

Show only last response 73

Show response parameter 73

Show response status 73

Show tester data 73

Snapshot 69, 81, 122

**T**

Tutorial 133

**U**

Update during cyclic execution 73

**V**

Vehicle 120

Vehicle name 118