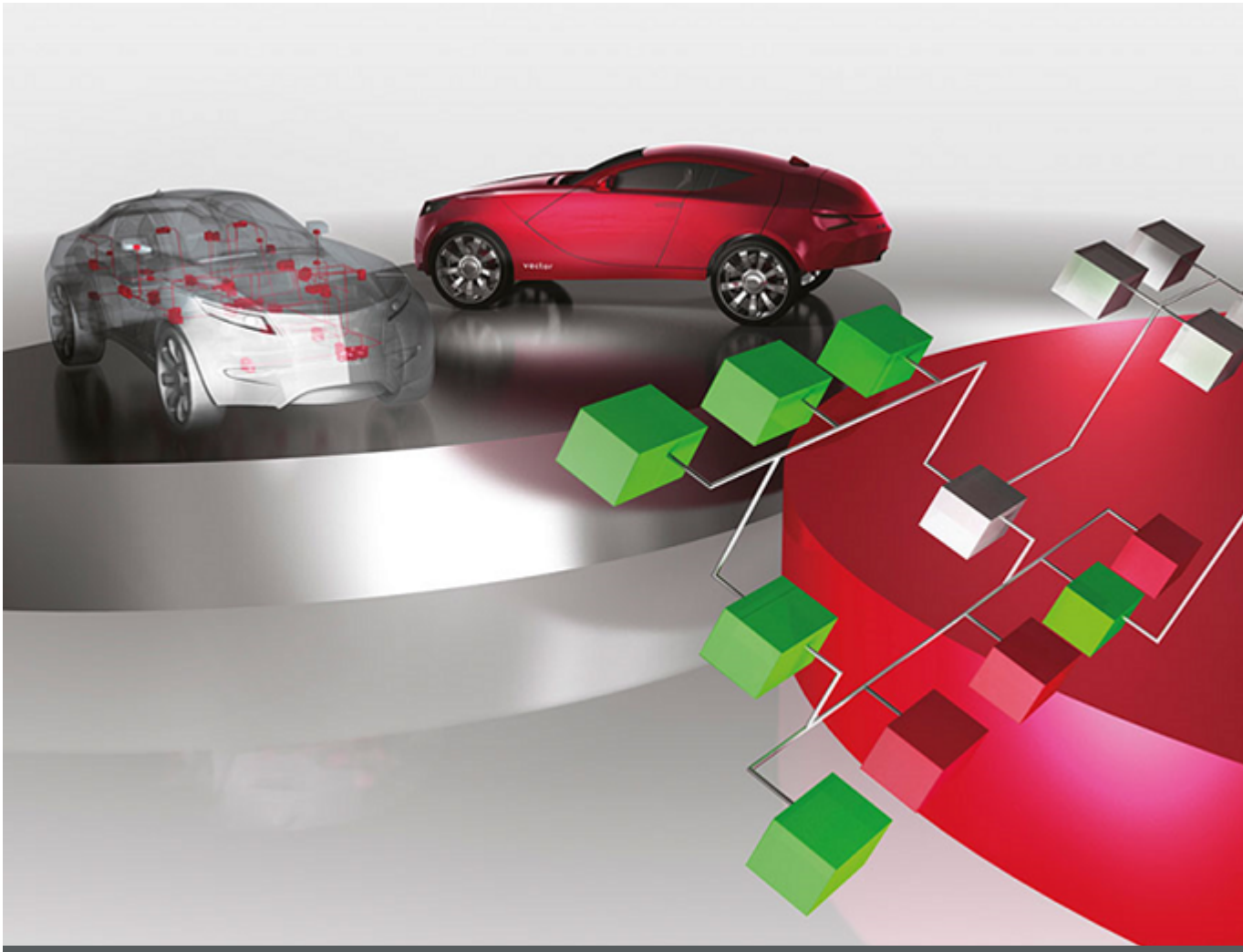
VECTOR



CANdb++ Manual

Version 3.1
English

Imprint

Vector Informatik GmbH
Ingersheimer Straße 24
D-70499 Stuttgart

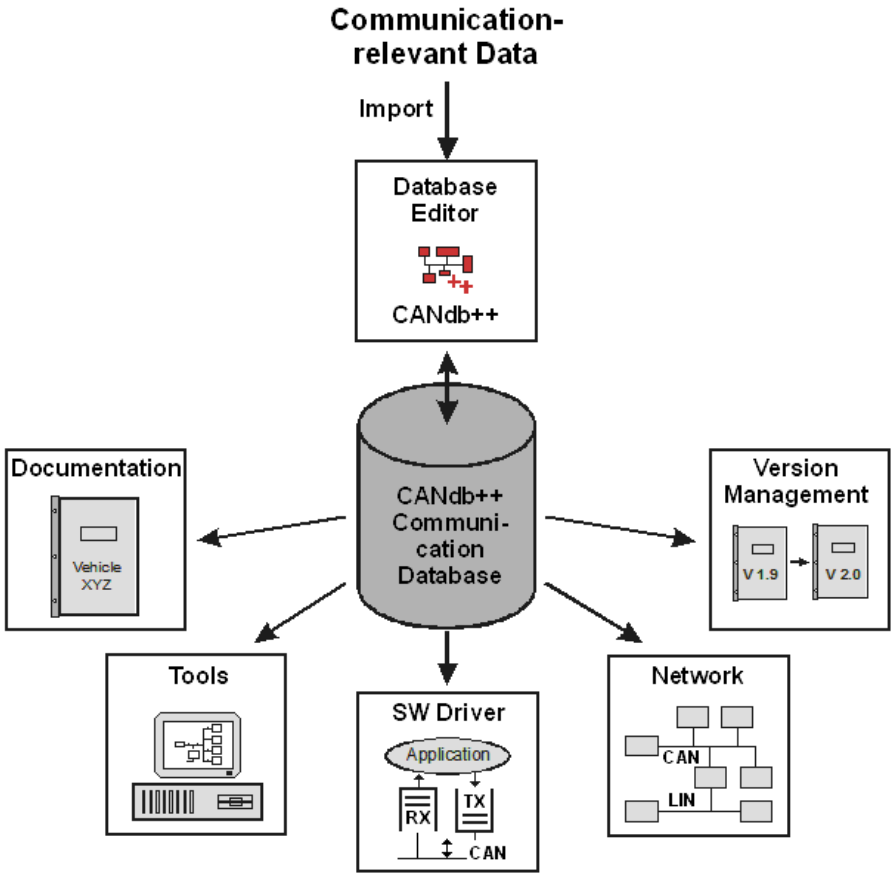
Table of Contents

1 Introduction

1.1 Concept

All communication-relevant data that are processed in a networked CAN bus system as well as their interrelationships are usually administered in a central communications database.

CANdb++ is a data administration program with which these communication data-bases can be created and modified in the form of CAN databases.



1.2 About This User Manual

1.2.1 Conventions

In the two tables below you will find the notation and icon conventions used throughout the manual.

Style	Utilization
bold	Fields/blocks, user/surface interface elements, window- and dialog names of the software, special emphasis of terms **[OK]** Buttons in brackets **File|Save** Notation for menus and menu commands
Microsoft	Legally protected proper names
`Source Code`	File and directory names, source code, class and object names, object attributes and values
Hyperlink	Hyperlinks and references
<CTRL>+<S>	Notation for key combinations

Symbol	Utilization
	Dangers that could lead to damage
	Notes and tips that facilitate your work
	More detailed information
	Examples
	Step-by-step instructions
	Text areas where changes of the currently described file are allowed or necessary

Symbol	Utilization
	Files you must not change
	Multimedia files e.g. video clips
	Introduction into a specific topic
	Text areas containing basic knowledge
	Text areas containing expert knowledge
	Something has changed

1.2.2 Certification

Vector Informatik GmbH has ISO 9001:2008 certification. The ISO standard is a globally recognized standard.

1.2.3 Warranty

We reserve the right to modify the contents of the documentation or the software without notice. Vector disclaims all liabilities for the completeness or correctness of the contents and for damages which may result from the use of this documentation.

1.2.4 Support

You can get through to our hotline at the phone number

+49 (711) 80670-200

or you send a problem report to the Vector Informatik GmbH Support.

1.2.5 Trademarks

All brand names in this documentation are either registered or non registered trademarks of their respective owners.

2 Basics

2.1 Overview of CANdb++

Program Versions

CANdb++ is available in the following program versions:

- CANdb++
- CANdb++ Admin

Functional Features

Function	CANdb++	CANdb++ Admin
Creating and modifying CANdb++ databases (*.mdc)	—	•
Creating and modifying CANdb network files (*.dbc)	•	•
Creating and modifying user-defined attributes	•	•
Creating and modifying value tables	•	•
Showing the communication matrix	•	•
Consistency check of a CAN database	•	•
Creating and modifying objects of the "Vehicles" object type	—	•
Comparing objects	•	•
Comparing CAN databases Export of differences in text files (*.csv)	—	•
Import objects and CAN databases	—	•
Export objects and CAN databases	•	•
Import of attribute definitions	•	•
Timing analysis (estimation of bus load)	—	•

Integration in the Tool Chain

CANdb++ is a central tool in the Vector CAN tools toolbox, and it can be started directly from Vector CAN tools such as CANalyzer, CANoe, CANape and CANscope. This provides the Vector CAN tools with direct access to communication-relevant data via CANdb++.

Communication-relevant data are defined, modified and managed entirely within CANdb++; the Vector CAN tools only read-access these data. The communication-relevant data must exist in the form of CANdb network files (*.dbc) so that the Vector CAN tools can access them.

To also permit the use of data from CANdb++ databases (*.mdc) CANdb++ provides a data export option. CANdb++ databases can be exported to one or more CANdb network files (*.dbc). In this way the user can either export the data of an entire CANdb++ database or only those relevant to a Vehicle, Network or ECU.

2.2 Datamodel of CANdb++

Object Types

Various object types are available in the Overview Window for modeling the communications structure of a vehicle's CAN bus system.

Links and Relations

Adding a link establishes a connection (Relation) between two objects of different object types. By linking a signal with a message, for example, the user can define the message in which this signal should be transmitted.

	Object Type of the Object to Which a Link Can Be Made						
Object Type of the Object to Be Linked	Vehicles	Networks	ECUs	Node Groups	Network Nodes	Messages	Signal Groups
Networks	•	—	—	—	—	—	—
ECUs	•	—	—	—	—	—	—
Node Groups	—	•	—	—	—	—	—
Network Nodes	—	•	•	•	—	—	—
Messages	—	—	—	—	•	—	—
Message Signals	—	—	—	—	•	—	•
Signal Groups	—	—	—	—	—	•	—
Signals	—	—	—	—	—	•	—

2.3 Installation

To install and run CANdb++ your hardware must satisfy the following requirements:

- Processor: at least Pentium
- Working memory (RAM): According to the minimum requirements of the respective operating system
- Operating system: Windows 7 and Windows 10

Note

You must have administrator rights to install the software! You can switch the language of the menus and dialogs at any time after installation (see 4.1 INI Files).

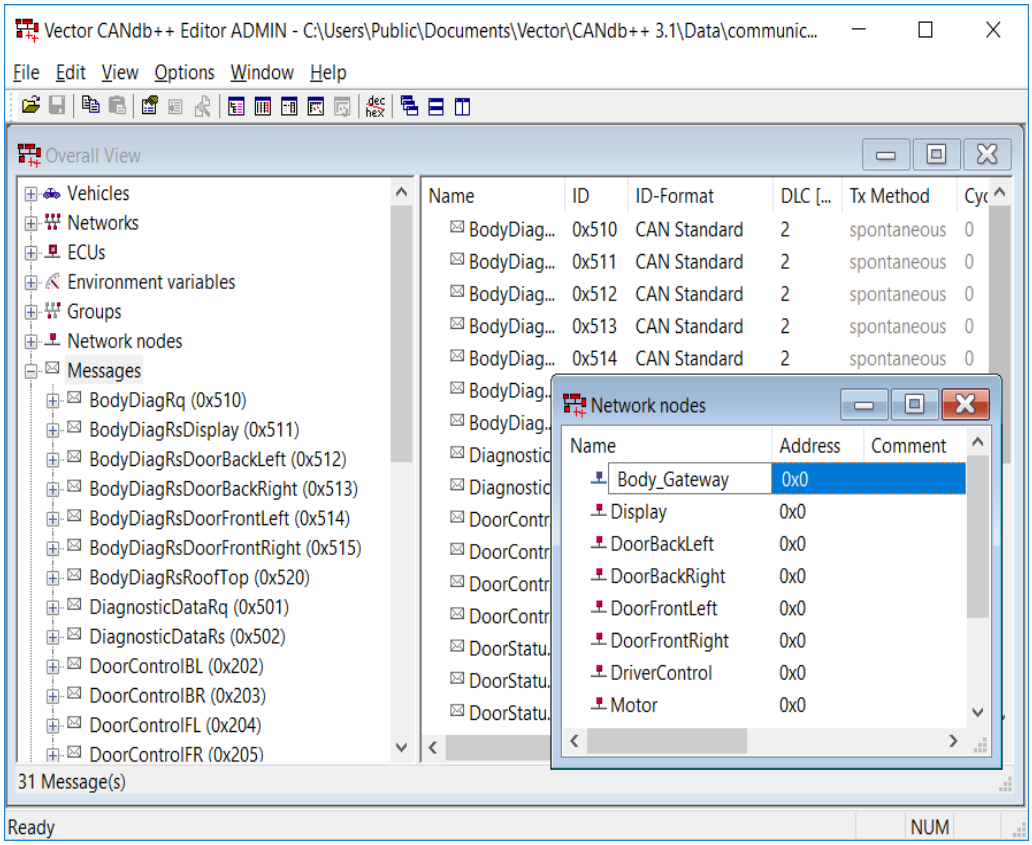
Installation

1. Insert the installation CD in your CD drive.
2. Call the installation program SETUP.exe.
3. Follow the instructions of the installation program.

2.4 Main Window

Elements of the Main Window

The following elements are arranged in the CANdb++ program window:



- Title bar
- Menu bar
- Toolbar
- Working area with various windows
- Status bar

3 Tutorial

3.1 Tutorial Overview

The purpose of this tutorial is to familiarize you with the CANdb++ user interface concept and the most important of the CANdb++ functions.

At the beginning of each chapter are problems that you can solve with the help of the explanations that follow.

When creating a new CAN database the following steps are performed in the sequence shown below:

Step by Step Procedure

1. Starting the program (see 3.2 Start the Program).
2. Setting up a new CAN database (3.3 Creating a New CAN Database).
3. Creating and modifying the necessary objects.
4. Creating new objects (see 3.3.2 Creating New Objects).
5. Copying existing objects (see 3.3.1 Copying Existing Objects).
6. Modifying existing objects (see 3.3.3 Modifying Existing Objects).
7. Linking the objects (see 3.4 Linking Objects).
8. Displaying the communications matrix (see 3.5 Display Communication Matrix).
9. Creating and assigning value tables (see 3.6 Value Tables and 3.7 Assign Value Tables).
10. Creating user-defined attributes and modifying values of the user-defined attributes of individual objects (see 3.8 Create User-Defined Attributes and 3.9 Modify Values of User-Defined Attributes).
11. Performing the consistency check and making necessary corrections to the CAN database (see 3.10 Perform Consistency Check).

3.2 Start the Program

Task

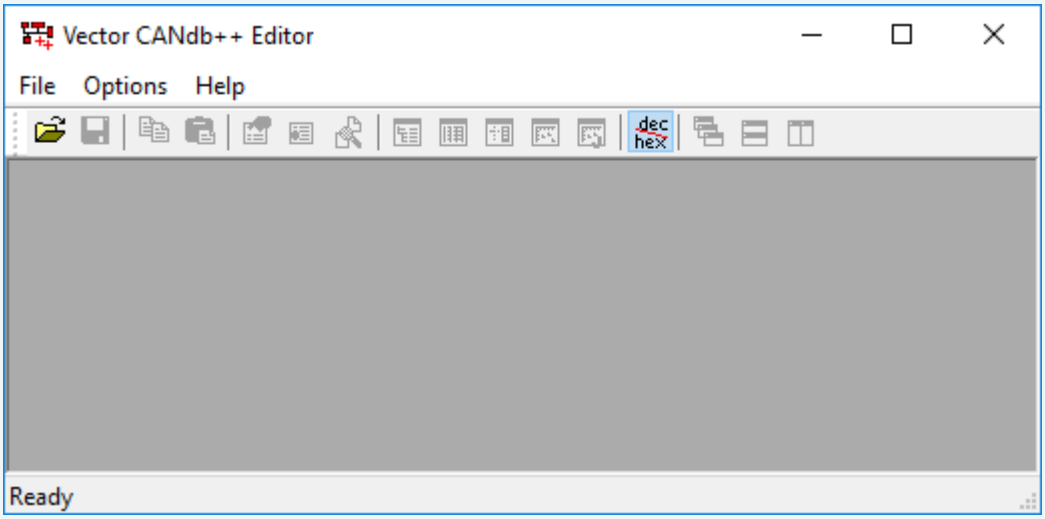
Start CANdb++.

Program Start

CANdb++ can be started as follows:

1. Double click the program icon in the CANdb++ program group.
2. Double click CANDB.exe (e.g. in Microsoft® Explorer).
3. Click the program name CANdb++ in the appropriate sub-folder of the Microsoft® Windows® Start menu.

After starting CANdb++ the CANdb++ program window appears. The working area is still empty, i.e. it does not contain any windows.



3.3 Creating a New CAN Database

Task

Create a new CANdb++ database with the name **Tutorial.mdc** or **Tutorial.dbc** in the Data directory of your CANdb++ installation.

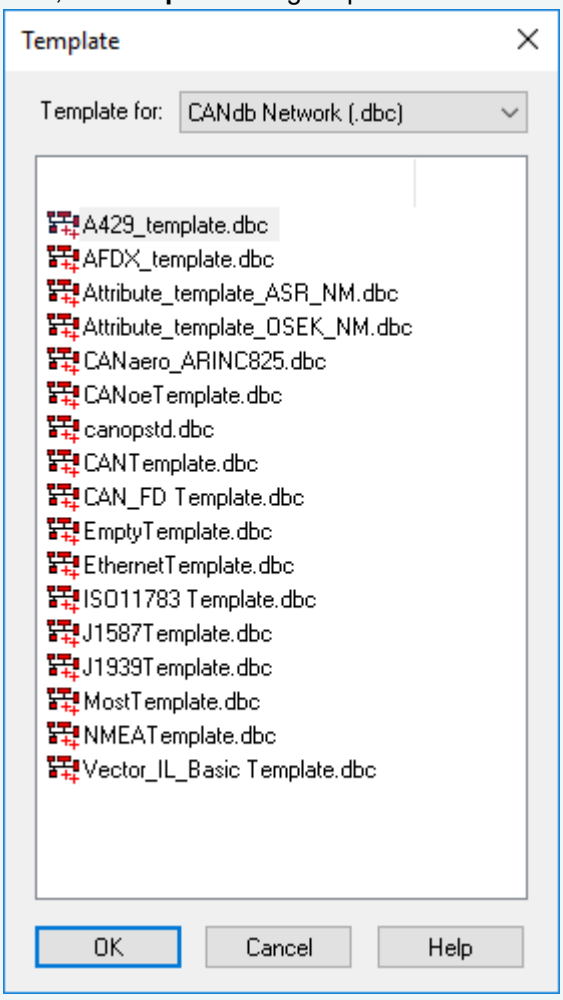
Creating a New Database

With CANdb++ you can set up CAN databases of the following types:

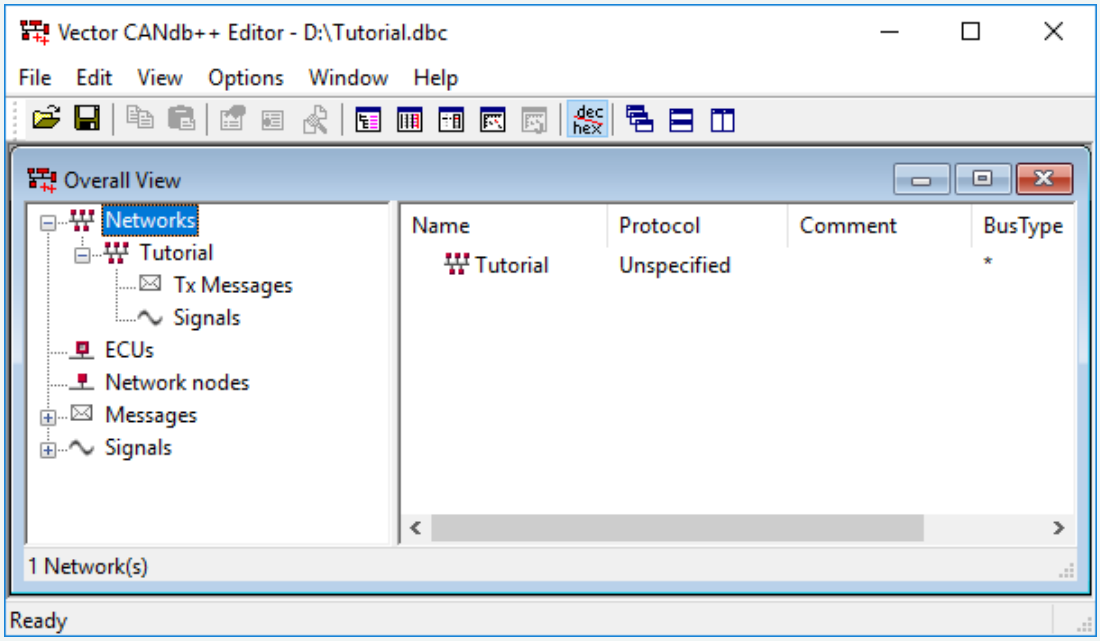
- CANdb++ databases (*.mdc) (CANdb++ Admin only)
- CANdb network file (*.dbc)

Proceed as follows to create a new CAN database:

1. Choose the **File|Create Database** command.
   First, the **Template** dialog is opened.



2. Here you can chose either a template for a CANdb network (.dbc) or for a CANdb++ database (.mdc).
3. Next you can select one of the available templates by a double click or by a single click and choosing **[OK]**.
4. After you have selected a template, the **New Database File** dialog is opened, in which the memory location, file name and file type can be defined for the CAN database to be created.
5. Select the directory in which you wish to save the new CAN database (Data directory of your CANdb++ installation or CANdb network).
   Select the desired file type (e.g. CANdb++ database) and enter the desired file name (e.g. Tutorial).
   It is not necessary to input a file name extension – it will be assigned automatically by CANdb++ according to the selected file type.
6. Click **[Save]**.
   The new CAN database (e.g. Tutorial.mdc or Tutorial.dbc) is set up and is shown in the Overview Window.
7. Shown on the left side of the Overview Window is a hierarchical overview of the available object types.



3.3.1 Copying Existing Objects

Example

Create a copy of the signal **RunMode** in the CAN database Tutorial.mdc or Tutorial.dbc and give it the name **OperateMode**.

Copy Existing Objects

Proceed as follows to create a copy of an existing object, e.g. of a signal:

1. Select the object that you wish to copy.
   To do this, click the object in the Overview Window.
   The selected object has a color background (Default: Blue).
2. Copy the selected object with the **Edit|Copy** command to the Clipboard.
3. Create a copy of the object located in the Clipboard using the **Edit|Paste** command.
4. Modify the copy of the object.

Note

After creating a copy of an object this copy is completely independent of the original object!

Changes to the original object are not mirrored in the copy, and if the changes are desired in the copy they must be made manually afterwards.

3.3.2 Creating New Objects

Task

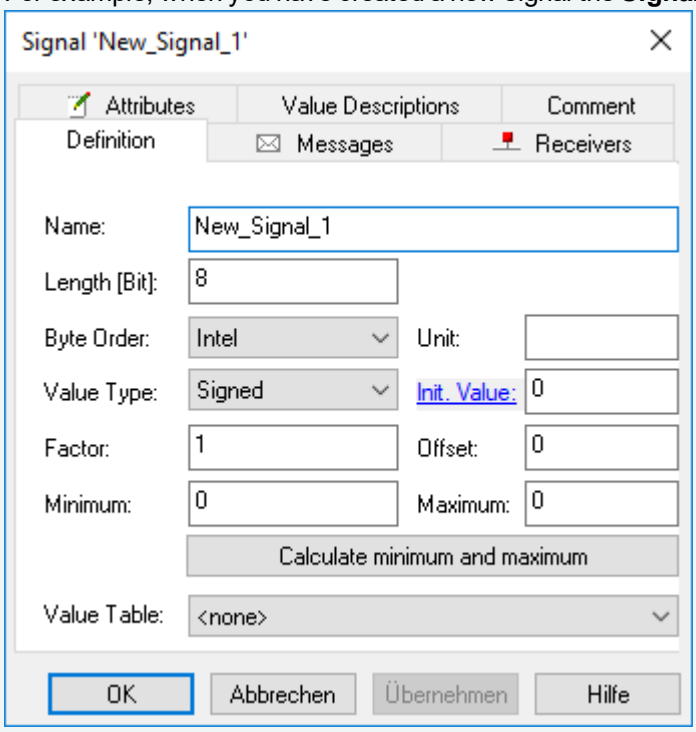
Create the following objects in the CAN database Tutorial.mdc or Tutorial.dbc:

- Vehicles (Coupe) (CANdb++ Admin only)
- Networks (Body, PowerTrain) (CANdb++ Admin only)
- Control units (Combi, DriverControl, ECU_Motor (CANdb++ Admin only)
- Network nodes (Body_Gateway, Display, Motor, Motor_Gateway, SteeringLock)
- Messages (DriverInfo, EngineControl, KeyData, TransmissionData)
- Signals (DisplayTemp, GearSelect, RunMode, Information)

Create New Objects

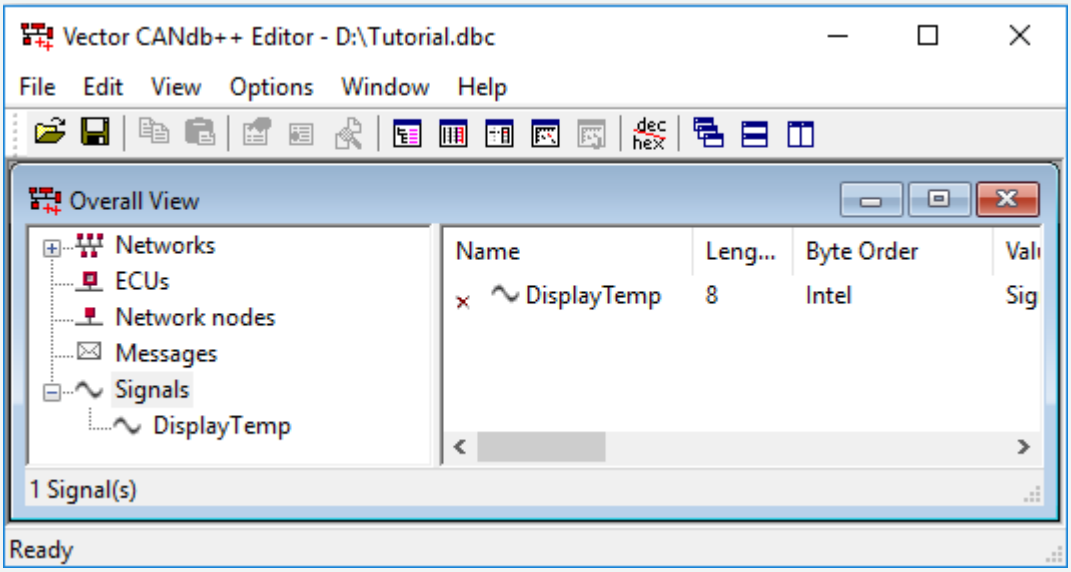
Proceed as follows to create a new object, e.g. a signal, in the Overview Window:

1. Select the object type for which you wish to set up a new object.
   To do this click the object type on the left side of the Overview Window.
   The selected object type gets a color background (Default: Blue).
   Appearing on the right side of the Overview Window are all objects of this type that have already been created.
   Using this method you would select, for example, the Signals object type if you wish to create a new signal.
2. Choose the **Edit|New command**.
   This opens an object dialog.
   Appearing in the input boxes of the object dialog are suggestions for concrete values of system parameters.
   For example, when you have created a new signal the **Signal…** object dialog appears.



3. Change the values of system parameters in the object dialog and click **[OK]**.

The newly created object appears in the Overview Window.
Appearing in the table columns on the right side of the Overview Window are the values of the object's system parameters.



3.3.3 Modifying Existing Objects

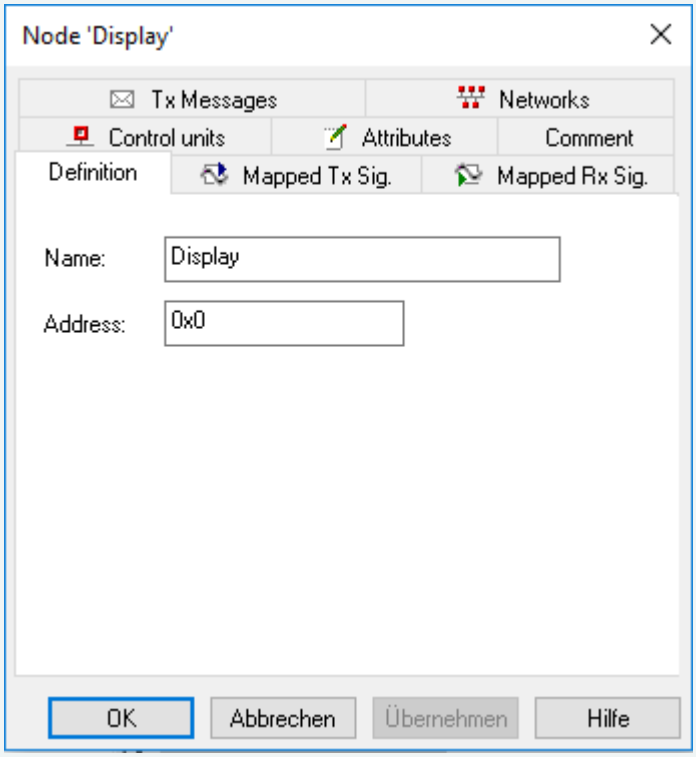
Task

Modify the network nodes in the CAN database Tutorial.mdc or Tutorial.dbc such that each node has a different address.

Solution 1: Modifying an Object's Parameters in the Object Dialog

Proceed as follows to modify the parameters of an object:

1. Select the object to be modified.
   To do this, click the object in the Overview Window.
   The selected object gets a color background (Default: Blue).
   In this manner you would select, for example, the network node **Display**.
2. Choose the **Edit|Edit** command.
   This opens an object dialog.
   Appearing in the input boxes of the object dialog are the concrete values for system parameters.
   For example, when you select the network node **Display** the **Node 'Display'** object dialog appears.



3. Modify the system parameters according to your requirements.
4. Click **[OK]**.
   The changed system parameters of the object appear in the table on the right side of the Overview Window.

Solution 2: Modifying an Object's Parameters in the Table in the Overview or List Window

Proceed as follows to modify an object's parameters directly in the table in the Overview Window or List Window:

1. Click the table cell containing the parameter of the object to be modified.
   The corresponding table line gets a color background (Default: Blue).
   The activated cell is shaded with a different color (Default: Light gray).
2. Activate the editing mode for the activated cell by pressing the **<F2>** key or by clicking the cell again.
   The cell gets a frame and a color background (Default: Blue), and if applicable a list of possible parameters is shown.
3. Change the parameter by entering a different value or selecting a different option from the list that is shown.
4. Press the **<Esc>** key if you wish to **abort** the procedure, i.e. you wish to leave the parameter value unchanged.
5. Press the **<Enter>** key if you wish to **accept** the changed value.

Note

You can switch the selection of the active cell within the table using the arrow cursor keys **<Left Arrow>**, **<Right Arrow>**, **<Up Arrow>** und **<Down Arrow>**.

3.4 Linking Objects

Task

Link the following objects in the CAN database Tutorial.mdc or Tutorial.dbc:

- Signals with messages
  (GearSelect-TransmissionData, DisplayTemp-DriverInfo, Information-DriverInfo, OperateMode-EngineControl, RunMode-KeyData)
- Messages with network nodes
  (TransmissionData-Display, EngineControl-Motor_Gateway, KeyData-SteeringLock, DriverInfo-Body_Gateway)
- Message signals with network nodes
  (DisplayTemp-Display, GearSelect-Body_Gateway, OperateMode-Motor, RunMode-Body_Gateway)
- Network nodes with node groups
  (Motor-PowerTrain_Base, Motor_Gateway-PowerTrain_Base)
  (CANdb++ Admin only)
- Network nodes with control units
  (Body_Gateway-Combi, Display-Combi, Motor-ECU_Motor, Motor_Gateway-Combi, SteeringLock-DriverControl)
  (CANdb++ Admin only)
- Network nodes with networks
  (Body_Gateway-Body, SteeringLock-Body, Display-Body, Motor-PowerTrain, Motor_Gateway-PowerTrain)
  (CANdb++ Admin only)
- Node groups with networks
  (PowerTrain_Base-PowerTrain)
  (CANdb++ Admin only)
- Control units with vehicles
  (Combi-Coupe, DriverControl-Coupe, ECU_Motor-Coupe)
  (CANdb++ Admin only)
- Networks with vehicles
  (Body-Coupe, PowerTrain-Coupe)
  (CANdb++ Admin only)

By linking two objects of different object types you can establish a connection (Relation) between these two objects.

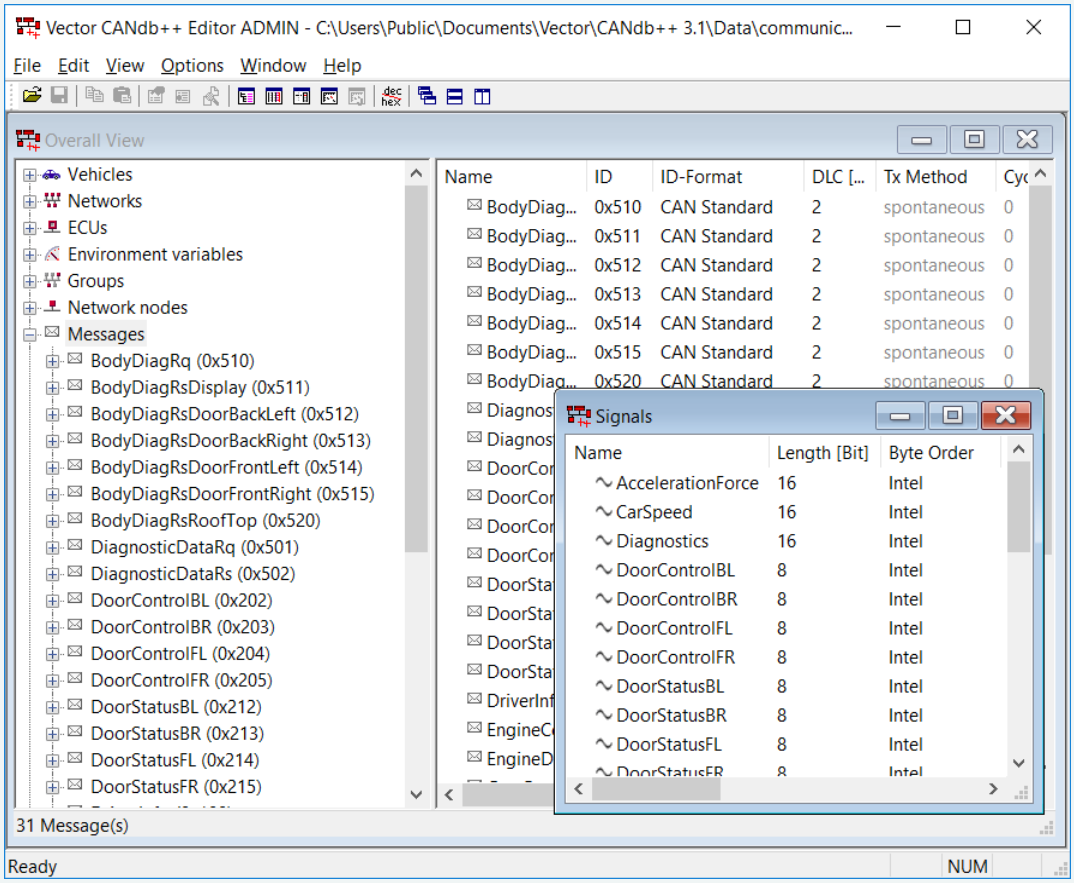
For example, by linking a signal with a message you can define the message in which this signal should be transmitted.

Solution 1: Linking by 'drag and drop'

Proceed as follows to link two objects by 'drag and drop':

1. On the left side of the Overview Window show the structure level that contains the object with which a link should be made.
   To do this, click the ⊞ symbol to the left of the object type for this object.
   The subordinate structure level is then shown.
   For example, if you wish to link a signal to a message click the ⊞ symbol to the left of the **Messages** object type.

2. Open a List Window for the object type of the object you wish to link.
   To do this, choose the **View|List...** command for the relevant object type.
   For example, if you wish to link a signal to a message choose the **List-Signals** command to open a List Window with signals.
   If necessary change the size of the Overview Window and the List Window so that the relevant areas are visible in both windows.



3. In the List Window select the object that you wish to link.
4. Link the two objects.
   To do this, drag the selected object from the List Window to the Overview Window while holding down the mouse button. Do not release the mouse button until the mouse pointer is located above the object in the Overview Window with which the link is to be made.
   The shape of the mouse pointer indicates whether the objects can be linked or not.

- The link is automatically added after releasing the mouse button.
- The linked object appears in the Overview Window below the object with which it was linked.
- The object symbol of the linked object changes to display the link.

For example, the object symbol of a linked signal, i.e. a message signal would appear as  .

As an alternative to this you could link objects by menu commands as described below:

Solution 2: Linking by Menu Commands

1. In the Overview Window select the object that you wish to link.
2. Copy the selected object to the Clipboard with the **Edit|Copy** command.
3. In the Overview Window select the object you would like to link to the object just copied.
4. Add the new link with the **Edit|Add link** command.
   The link is then added.

3.5 Display Communication Matrix

A communications matrix shows the communications relationships between signals, messages and network nodes in table format.

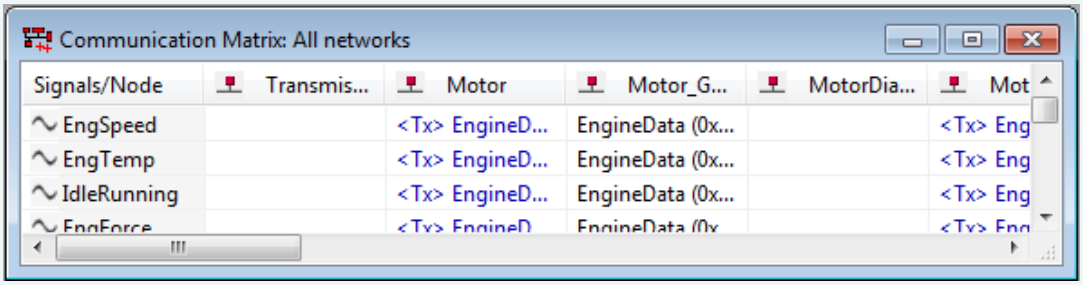
Task

Open the Communications Matrix Window that contains the communications matrix of the CAN database Tutorial.mdc or Tutorial.dbc.

Display Communication Matrix

Proceed as follows to have the communications matrix displayed:

1. Choose the **View|Communications matrix** command.
   This automatically opens the Communications Matrix Window with the communications matrix of the active CAN database.



In the Communications Matrix Window the signals are arranged by lines and the network nodes by columns. The table boxes with the signal names have gray shading. The messages shown in the remaining table boxes are the messages in which the signal is transmitted or received by the particular network node.

Transmitted messages are identified as follows:

- Message name in blue font
- "<Tx>" to the left of the message name

Note

When the Communications Matrix Window remains open it is automatically updated with each modification to the CAN database.

3.6 Value Tables

Task

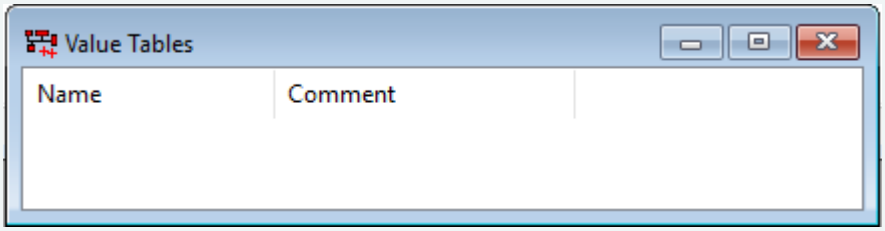
In the CAN database Tutorial.mdc or Tutorial.dbc create the value table **Colors** which assigns the concrete values 0, 1 and 2 to the symbolic identifiers 'red', 'yellow' and 'green'.

Creating a Value Table

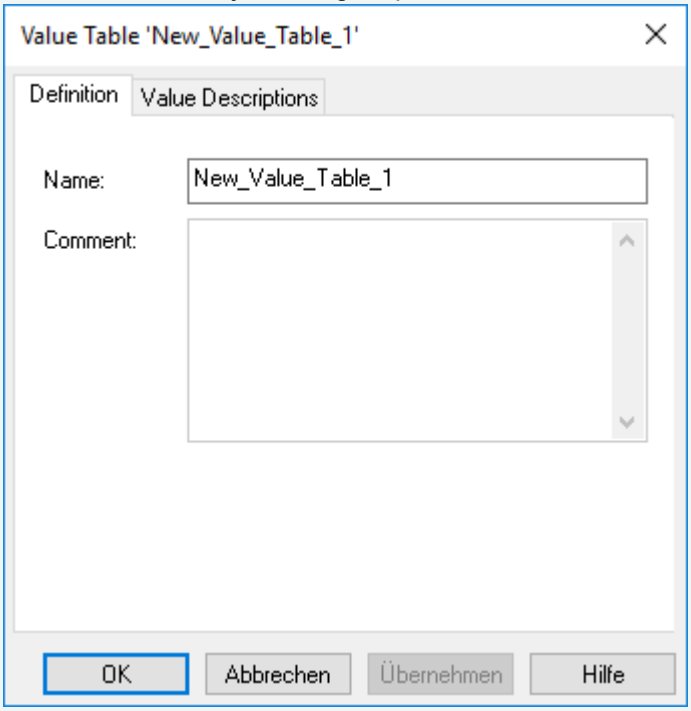
In a value table individual signal values can be as-signed to symbolic identifiers.

Proceed as follows to create a value table:

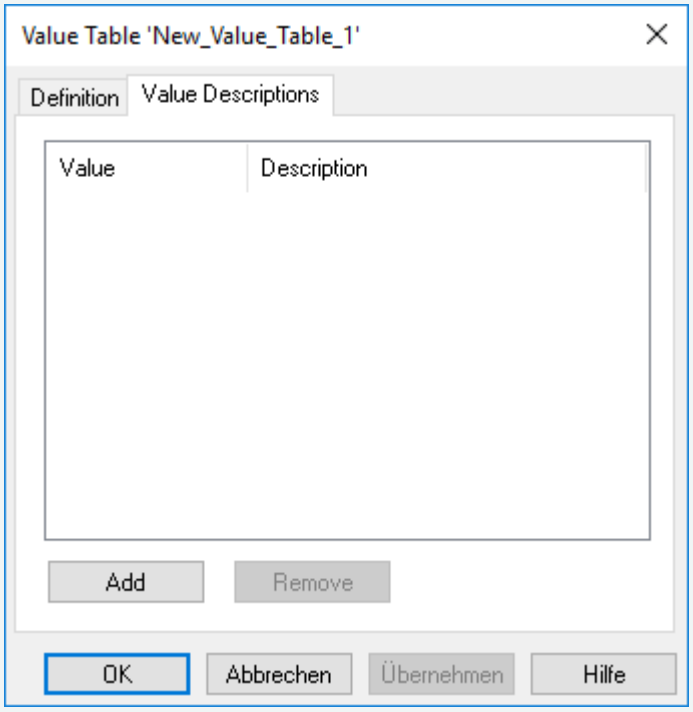
1. Choose the **View|Value Tables** command.
   The Value Tables Window is opened.



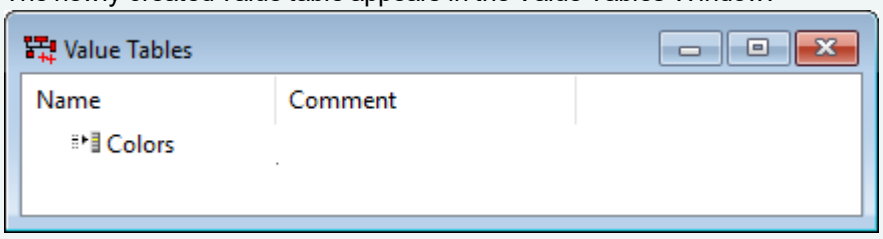
2. Choose the **Edit|New** command.
   The **Value Table** object dialog is opened.



3. Change the name of the value table.
   Enter the new name in the **Name** input box.
4. Activate the **Value Description** page by clicking its page.



5. Click **[Add]**.
   A new line is added to the table.
   Appearing in the **Value** column is a suggested concrete value for the signal to which a symbolic identifier should be assigned.
   Appearing in the **Description** column is a suggested text for this symbolic identifier.
6. Click the cell whose text you wish to change.
   The cell is selected by a frame and can now be edited.
7. Change the values and symbolic identifiers.
8. Press the **<Esc>** key if you wish to **abort** the procedure, i.e. you wish to leave the value or symbolic identifier unchanged.
   Press the **<Enter>** key if you wish to **accept** the changed value or changed identifier.
9. Click **[OK]**.
   The newly created value table appears in the Value Tables Window.



3.7 Assign Value Tables

Task

In the CAN database Tutorial.mdc or Tutorial.dbc assign the **Colors** value table to the Information signal.

Assigning Value Tables

The value table must be assigned to a signal or so that the symbolic identifiers that are assigned to individual values in a value table will indeed be assigned to the signal values.

Proceed as follows to assign a value table to a signal:

1. Select the signal to which the value table should be assigned.
   To do this, click the specific object in the Overview Window or List Window.
2. Choose the **Edit|Edit** command.
   This opens an object dialog.
   For example, if you select the **Information** signal the **Signal 'Information'** object dialog appears.
3. In the **Value Table** list box select the value table that you wish to assign to the signal.
4. Click **[OK]**.

3.8 Create User-Defined Attributes

Task

In the CAN database Tutorial.mdc or Tutorial.dbc create the user-defined attribute Release for Vehicles.

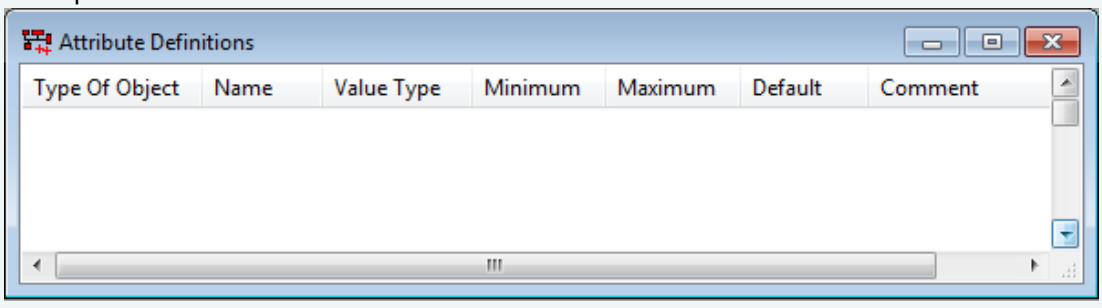
Concrete values available for the **Release** attribute should be: '**pending**', '**confirmed**' and '**denied**'.

Step by Step Procedure

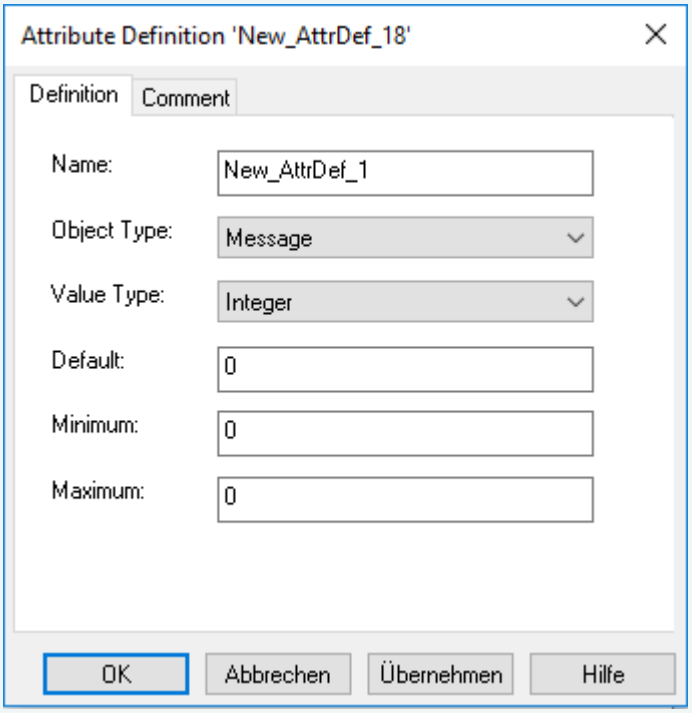
In addition to system parameters that must be defined when creating an object, objects can also be assigned user-defined attributes.

Proceed as follows to create a user-defined attribute:

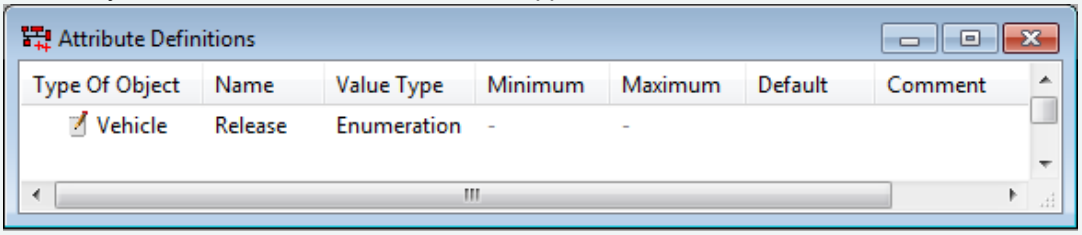
1. Choose the **View|Attribute Definitions** command.
   This opens the **Attribute Definitions** Window.



2. Choose the **Edit|New** command.
   The **Attribute Definition** object dialog is opened.
   Appearing in the input boxes are suggested concrete values for parameters of the user-defined attribute.



3. Change the parameters of the user-defined attribute.
   Enter the new name and select the object type and value type.
4. To provide concrete texts as values you must select the **Enumeration** value type.
   These texts can then be entered in the **Value Range** input box that then appears.
   The value in the **Default** input box is assigned as the default value of the user-defined attribute for each object of the selected object type.
5. Click **[OK]**
   The newly created user-defined attribute now appears in the Attribute Definitions Window.



Like system parameters, the values of user-defined attributes also appear in the columns on the right side of the Overview Window or in the corresponding List Windows.

Note

User-defined attributes of objects which still have their default values are identified by a * after the attribute value on the right side of the Overview Window.

3.9 Modify Values of User-Defined Attributes

Task

In the CAN database Tutorial.mdc or Tutorial.dbc set the user-defined attribute Release for the Vehicle object Coupe to the value denied.

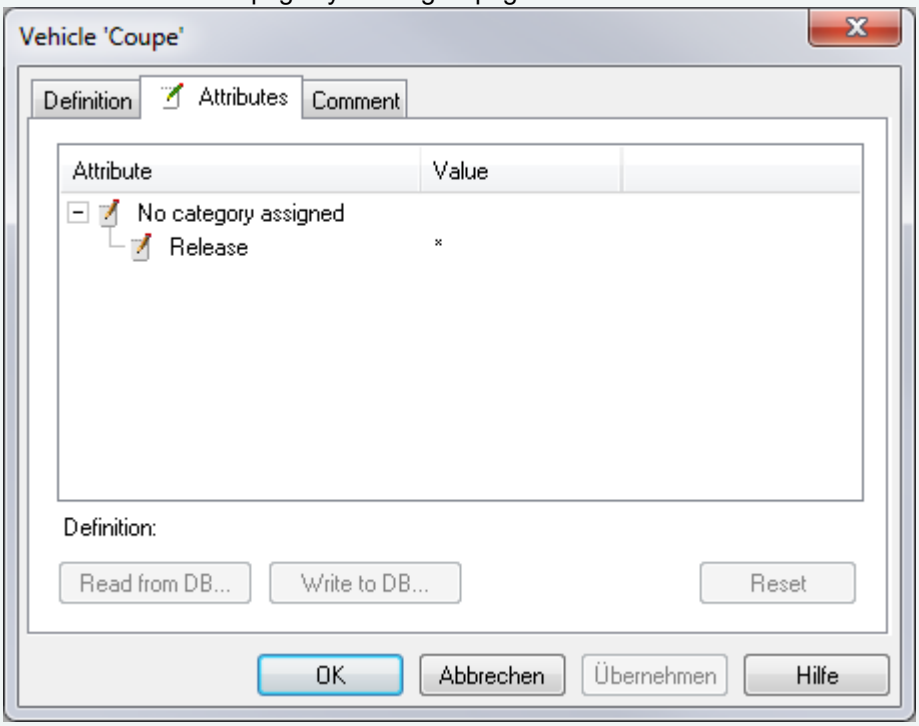
Modify Values of User-Defined Attributes

Like system parameters, the values of user-defined attributes can also be modified at any time.

Proceed as follows to modify the value of an object's user-defined attribute:

Solution 1: Modifying the Values in the Object Dialog

1. Select the object whose user-defined attribute value you wish to modify.
2. Choose the **Edit|Edit** command.
   The Object dialog is opened.
3. Activate the Attribute page by clicking its page.



4. Activate the user-defined attribute whose value you wish to change by clicking the appropriate table cell.
   The corresponding table line is highlighted in color (Default: Blue).
5. Click the value of the attribute.
6. A frame appears around the cell and if applicable the available attribute values are shown.
7. Enter the desired value or select it from the list.
8. Press the **<Esc>** key if you wish to **abort** the procedure, i.e. you wish to leave the value unchanged.
   Press the **<Enter>** key if you wish to **accept** the changed value.
9. Click **[OK]**.
   The changed value of the object's user-defined attribute appears in the table on the right side of the Overview Window.

Solution 2: Modifying the Values in the Table in the Overview or List Window

1. Click the table cell containing the object's user-defined attribute to be modified.
   The corresponding line in the table is highlighted in color (Default: Blue).
   The activated cell is shaded with a different color (Default: Light gray).
2. Activate the editing mode for the activated cell by pressing the **<F2>** key or clicking the cell

again.
A frame appears around the cell and it is shaded in color (Default: Blue); if applicable a list of possible values is shown.

3. Change the attribute value by entering a different value or selecting a different option from the list shown.
4. Press the **<Esc>** key if you wish to **abort** the procedure, i.e. you wish to leave the parameter value unchanged.
   Press the **<Enter>** key if you wish to **accept** the changed value.

3.10 Perform Consistency Check

Task

Execute an automatic consistency check for the CAN database Tutorial.mdc or Tutorial.dbc with CANdb++ and correct any inconsistencies that are found.

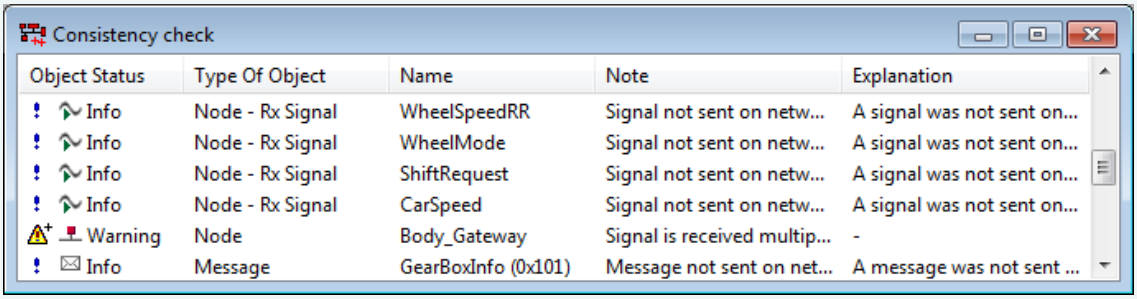
Perform Consistency Check

To determine whether the objects of a CAN database and their interrelationships are consistent with one another, an automatic consistency check can be performed with CANdb++.

Proceed as follows to perform a consistency check:

1. Choose the **File|Consistency check** command to start the automatic consistency check.
2. The results of the consistency check are displayed in a **Consistency Check** Window.

- If CANdb++ does not find any inconsistencies in the CAN database, then there will be no entries in the Consistency Check Window.
- However, if inconsistencies are found in the CAN database they will be listed in the Consistency Check Window.



Note

When the Consistency Check Window is open it is updated automatically with each modification of the CAN database.

4 API

4.1 INI Files

Location of INI Files

The CANdb++ installation directory contains the file **vcustom.ini**. In this file you can make the following settings for CANdb++ 3.1:

- Section **[UserSelection]**, item **[SettingsFolder]**:
  - The storage location for the configuration file **CANdb.ini** is defined there.
    Default value: **C:\ProgramData\Vector\CANdb++ 3.1**
- Section **[UserSelection]**, item **[UserFolder]**:
  - The storage location for the example databases (subdirectory **Data**) and for the database templates (subdirectory **Templates**) is defined there.
    Default value: **C:\Users\Public\Documents\Vector\CANdb++ 3.1**

To make changes to options it is possible to edit the INI files with any ASCII editor.

Note

You should exit CANdb++ before editing the INI files.

The changed options in the INI files do not take effect until CANdb++ is restarted.

4.1.1 CANdb.ini

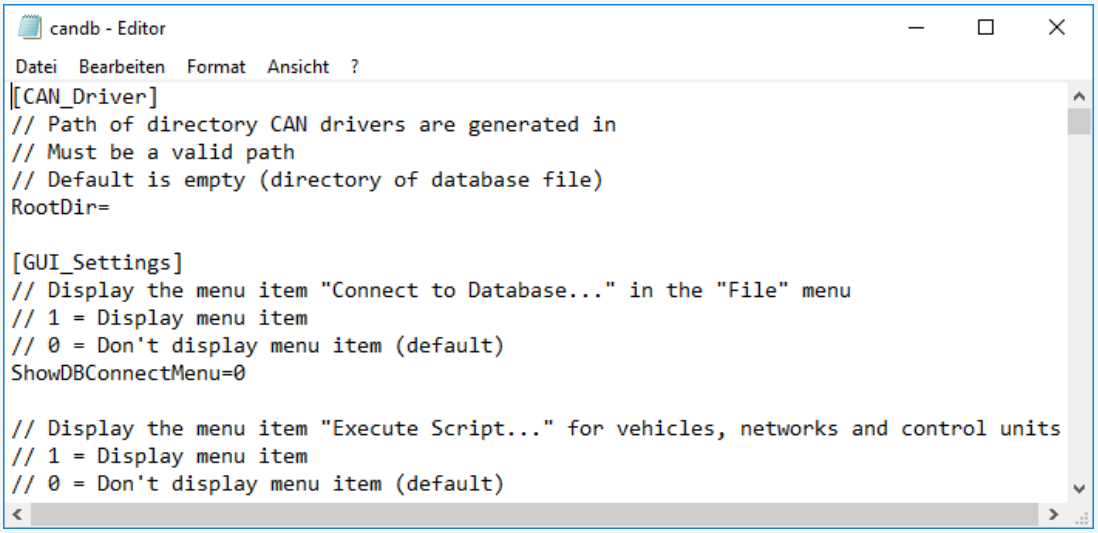
The following options can be configured for CANdb++ in CANdb.ini:

- Language of the menus and dialogs
  (section `[Language]`, line `"Country"`)
- Root directory for generated CAN drivers
  (section `[CAN_Driver]`, line `"RootDir="`)
- Display format for attributes of type Integer
  (section `[GUI_Settings]`, line `"DisplayIntAttrsAlwaysDecimal="`)
- Format to display signals with motorola byte ordering
  (section `[Format]`, line `"MotorolaFormat="`)
- Indexing of bits in layout dialog of messages
  (section `[Format]`, line `"UseInvertedLayoutIndexing="`)

Step by Step Procedure

Proceed as follows to make changes to these options:

1. Open the CANdb.ini file with an ASCII editor.



2. If you want to set the root directory for generated CAN drivers, enter the desired root directory in the line `RootDir=` of the section `[CAN_Driver]`.
   When generating CAN drivers, sub-directories are created automatically in the root directory. The generated CAN drivers are stored in these sub-directories.
3. Store the file CANdb.ini and close the editor.
   If CANdb++ was opened during editing, you have to exit CANdb++ and open it again to work with the new settings.

5 Glossary

B

BusType
Contains the bus type or the network protocol. In the CANdb++ default setting("PropsFromUserAttrs=1" and "AttrBusType=BusType" in CANdb.ini), this value is mapped to the system attribute 'protocol' of a network.

C

CHM
Help file (Microsoft Compiled HTML Help)

CSV
Text file with Comma/Character Separated Values

D

DBC
CANdb network file (Data Base for CAN)

DLL
Run-time library (Dynamic Link Library)

E

EXE
Executable program

G

GenMsgCycleTime
Contains the cycle time of the message or the relation node-Tx message. With the default settings in CANdb++ ("PropsFromUserAttrs=1" and "AttrMsgCycleTime=GenMsgCycleTime" in CANdb.ini) this value is mapped to the system attribute cycle time. In the default setting, therefore, the system attribute cycle time of a message (e.g. on the "Message - Definition" page) cannot be processed directly; instead, it is determined via the user-defined attribute.

GenMsgSendType
Describes the transmitter type of the message or the relation node-Tx message. The available values are user or OEM-specific. With the default settings in CANdb++ ("PropsFromUserAttrs=1" and "AttrMsgSendType=GenMsgSendType" in CANdb.ini) this value is mapped to the system attribute transmitter type (Tx method). In the default setting, therefore, the system attribute transmitter type of a message (e.g. on the "Message - Definition" page) cannot be processed directly; instead, it is determined via the user-defined attribute.

GenSigInactiveValue
Describes the value of a signal for which the signal is in the inactive state. Signals that are sent with the transmitter type (GenSigSendType) 'IfActive' are sent as soon as the signal assumes a value not equal to the inactive

value. Signals that are sent with the transmitter type 'IfActiveWithRepetion' are normally sent cyclically. However, if the signal value assumes the inactive state, the signal must not be sent.

GenSigSendType
The GenSigSendType attribute specifies the send type of a signal.

GenSigStartValue
Describes the start value or initial value of a signal.

I

INI
File with configuration settings.

M

MDC
CANdb++ CAN database

N

NmStationAddress
A node that uses the OSEK NM (network management) must have a NM-ID that is unambiguous on the network (within a vehicle, the NM-ID does not have to be unambiguous). This NM-ID is managed in the attribute NMStationAdress. In the CANdb++ default setting("PropsFromUserAttrs=1" and "AttrBusType=NMStationAddress" in CANdb.ini), this value is mapped to the system attribute 'address' of a node.

T

TXT
Text file

6 Index

More Information

- News
- Products
- Demo Software
- Support
- Training Classes
- Addresses

www.vector.com