# Software service tool for electric vehicle system

Glenn Bergqvist, Danaher Motion Särö AB

**The paper describes the work developing a service tool to be used together with CANopen-based system. It presents the thoughts and requirements before and during the development process. The service tool has the purpose to handle design, production and service/maintenance functions in an electrical vehicle. This includes functions like diagnostics, parameter setup, reports and software download.**

**A presentation how it has been implemented to enable all the function for both advanced users and as a user-friendly tool for non-experienced CANopen-users is also included.**

**Figure 1: Man and machine**

## 1 Introduction

A problem with putting new CANopen systems out on the market can be the fact that it is a traditional application where old mechanical, electric or hydraulic equipment has been exchanged to a more or less complex distributed system. Both design engineers and especially service and maintenance people have no or limited knowledge of CAN and distributed systems.

Requests like “Take a CAN-log!”, “Increase parameter 0x2400 Sub Index 3 to 500!”, “Update software to revision 5!”, “Check the error log!” can be incomprehensible or impossible for service or maintenance persons.

Development of the service tool software had the target to simplify the work and make it better suitable for these users. Even with an adapted tool it can also be a struggle to have companies/organizations to take the step to use a PC as a tool for service and maintenance.

## 2 What features do the users want to have?

Putting the requirements together shows that the users isn't primarily interested in the fact that it is a CANopen system but instead requires to efficiently service his application/system.

This together with other requirements like language support, easy distribution, upgradeable with more service applications and access levels builds the requirement base.

## 3 Who is the user?

The range of users can be from people with good knowledge regarding CANopen systems to users with none or low knowledge. The tool needs to be scaleable to support as many user profiles as possible and also for the different stages in the lifecycle of the vehicle/system. From design, production to service and maintenance of the vehicle.

## 4 Language support in software and created application

To be usable in different countries there is also a need to support different languages both in the software itself and in the created service application.

## 5 Connection of the tool to vehicle/system

The service tool needs to verify that it is connected to the right vehicle/system and to secure that all actions are handled in a safe way. The solution to this is to describe the system and all CANopen nodes available in the system.

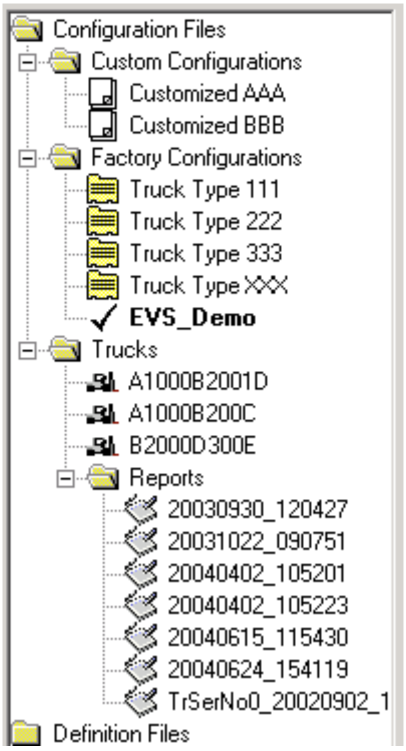
The fact that some systems are ‘flexible’ and sometimes have optional nodes and also can have variants of products needs to be supported.

Identification of the nodes is done with standard CANopen entries and additions:

- Device Type (0x1000:0)
- Software identification

It is also needed to have some possibilities to mask this information to accept minor updates of the connected device.

As the tool supports multiple service applications there are functions to administrate these in a tree view.



```
Configuration Files
⊟ Custom Configurations
    Customized AAA
    Customized BBB
⊟ Factory Configurations
    Truck Type 111
    Truck Type 222
    Truck Type 333
    Truck Type XXX
    ✓ EVS_Demo
⊟ Trucks
    A1000B2001D
    A1000B200C
    B2000D300E
  ⊟ Reports
      20030930_120427
      20031022_090751
      20040402_105201
      20040402_105223
      20040615_115430
      20040624_154119
      TrSerNo0_20020902_1
Definition Files
```

**Figure 2: Tree view of configurations**

## 6 Communication interface

Requirements for different interfaces, connectors also add variation in the usage of the tool. Basic connection with a standard CAN-interface, most common USB to CAN, is needed as a first level.

For cost sensitive applications this interface can be an issue and solutions like direct RS232 or USB to the target system is wanted to minimize cost. With such a solution you normally do not get all the other benefits possible when you are connected to the CAN bus accessing all nodes, taking CAN-traffic logs etc.

Wireless or remote connection is also wanted for some applications. Short distance connection with Bluetooth/WiFi and long distance, remote, with GSM or over Internet is possible. This adds other requirements regarding safety and limitations caused by bandwidth.

## 7 XML file system to support the system

To meet the requirements for this functionality a XML-based file system has been developed to handle all information needed to be available for the service tool. It is open and easy to upgrade with new information. Any CANopen-device can be added and supported. A system for signing the files to prevent manipulation of contents has been added.

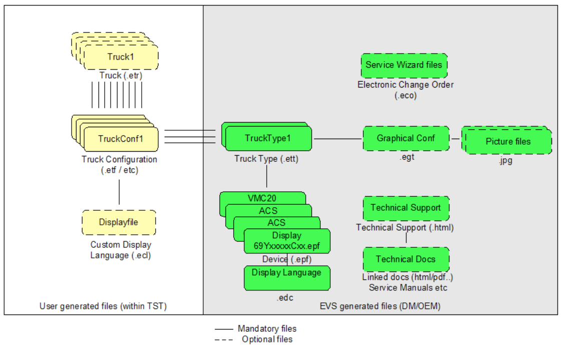


**Figure 3: XML-file system implemented**

Most important files are the Product file (epf) and the System description file (ett).

## 8 Product file

For each product (device) a file is created which includes following information. The

Product file is created by the designer of the device/product.

Product:

- Name of the product
- Software article number
- Checksum information
- Timestamp information

Connection:

- Communication speeds supported by the product (including default)
- Node number information

A big part of the product file is normally the object dictionary which includes:

- Symbolic name
- Object type (variable or array)
- Index and SubIndex
- DataType and AccessType
- Minimum, maximum, default value
- Scaling information (factor and offset)
- Unit information
- Bitfield information (to describe symbols with less then 8 bits)
- PublicName (user friendly name of the symbol)
- Visibility (operator/service/OEM/Internal)
- MaxArraySize
- Bitfield information (to describe symbols with less then 8 bits)

In the product file information regarding compatibility can also be described and is used by the tool to secure downloading of compatible software.

The last part of the product file includes the software in hex-format to enable downloading software to the product.

For some special products as displays (HMI) a separate file can be used to store all string information etc needed for the display to support multiple languages.

## 9 System description file

To describe the vehicle/system an xml-file called TruckType is used. Information inside describes which products (nodes) is included. This file is created by the designer of the service application.

TruckType:

- Name of the vehicle
- Article number
- Baud rate

Devices:

- Node ID
- DeviceName
- Article number
- Article number match (to make it possible to have a mask as a rule to access the vehicle with the tool)
- Type code (DeviceType)
- Type code mask (to make it possible to have a mask as a rule to access the vehicle with the tool)
- SDO RX and TX information about COB-ID to use
- Information if device is optional or mandatory
- Public name. A user friendly name showed in the software. Included for all strings showed in the software there is the possibility to add a language code attribute to support different languages

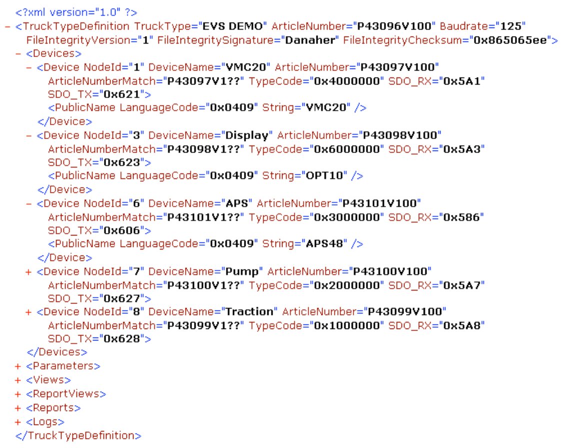
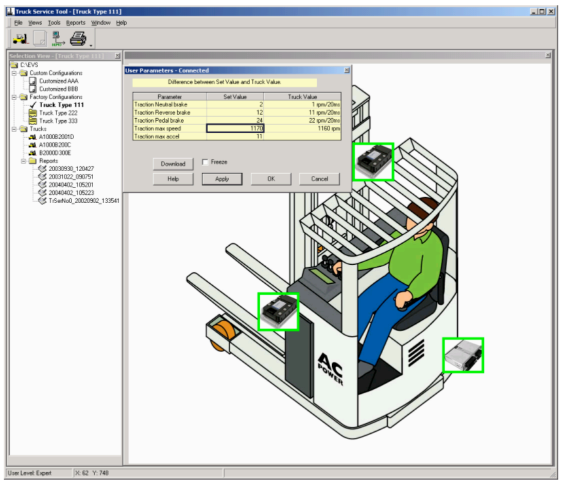

```
  <?xml version="1.0" ?>
- <TruckTypeDefinition TruckType="EVS DEMO" ArticleNumber="P43096V100" Baudrate="125"
    FileIntegrityVersion="1" FileIntegritySignature="Danaher" FileIntegrityChecksum="0x865065ee">
  - <Devices>
    - <Device NodeId="1" DeviceName="VMC20" ArticleNumber="P43097V100"
        ArticleNumberMatch="P43097V1??" TypeCode="0x4000000" SDO_RX="0x5A1"
        SDO_TX="0x621">
        <PublicName LanguageCode="0x0409" String="VMC20" />
      </Device>
    - <Device NodeId="3" DeviceName="Display" ArticleNumber="P43098V100"
        ArticleNumberMatch="P43098V1??" TypeCode="0x6000000" SDO_RX="0x5A3"
        SDO_TX="0x623">
        <PublicName LanguageCode="0x0409" String="OPT10" />
      </Device>
    - <Device NodeId="6" DeviceName="APS" ArticleNumber="P43101V100"
        ArticleNumberMatch="P43101V1??" TypeCode="0x3000000" SDO_RX="0x586"
        SDO_TX="0x606">
        <PublicName LanguageCode="0x0409" String="APS48" />
      </Device>
    + <Device NodeId="7" DeviceName="Pump" ArticleNumber="P43100V100"
        ArticleNumberMatch="P43100V1??" TypeCode="0x2000000" SDO_RX="0x5A7"
        SDO_TX="0x627">
    + <Device NodeId="8" DeviceName="Traction" ArticleNumber="P43099V100"
        ArticleNumberMatch="P43099V1??" TypeCode="0x1000000" SDO_RX="0x5A8"
        SDO_TX="0x628">
    </Devices>
  + <Parameters>
  + <Views>
  + <ReportViews>
  + <Reports>
  + <Logs>
  </TruckTypeDefinition>
```

**Figure 4: System description file (ett)**

## 10 Views

The configurable user interface which is normally unique for each vehicle type is described as views. They can be of different sorts for diagnostics (read only) or setup for writing information to the vehicle. Other types of views are logs for reading array-information and reports useable to make printouts.



Difference between Set Value and Truck Value.

| Parameter | Set Value | Truck Value |
|---|---|---|
| Traction Neutral brake | 2 | 1 rpm/20ms |
| Traction Reverse brake | 12 | 11 rpm/20ms |
| Traction Pedal brake | 24 | 22 rpm/20ms |
| Traction max speed | 1170 | 1160 rpm |
| Traction max accel | 11 | |

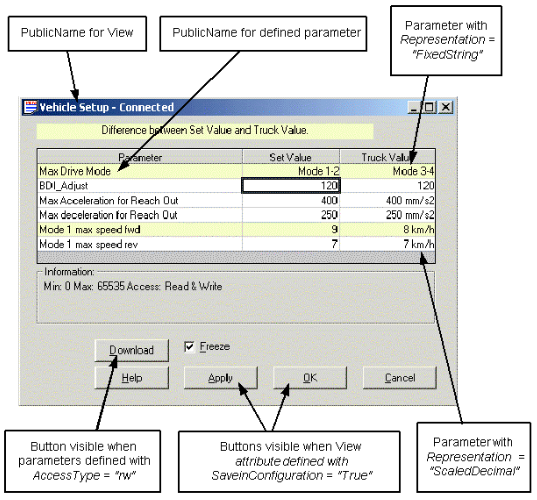
**Figure 5: User interface**

Also here language code attribute can be used to fully support different countries.

- View name
- Visibility (access level)
- Save in configuration enable (for setup views where value wants to be stored on the PC)
- Update interval (how often SDO-question is sent in ms)
- Public name (with language code support)

Each view has contents of a number of parameters. Description of each of them includes:

- Symbolic address (describing which node and object)
- Access type (rw, wo, const)
- Representation. Describes in which format information shall be shown (Decimal, ScaledDecimal, Hexa-decimal, String, FixedString, ....)


PublicName for View
PublicName for defined parameter
Parameter with *Representation = "FixedString"*
Vehicle Setup - Connected
Information:
Min: 0 Max: 65535 Access: Read & Write
Download
Freeze
Help
Apply
OK
Cancel
Button visible when parameters defined with *AccessType = "rw"*
Buttons visible when View *attribute defined with SaveinConfiguration = "True"*
Parameter with *Representation = "ScaledDecimal"*

Difference between Set Value and Truck Value.

| Parameter | Set Value | Truck Value |
|---|---|---|
| Max Drive Mode | Mode 1-2 | Mode 3-4 |
| BDI_Adjust | 120 | 120 |
| Max Acceleration for Reach Out | 400 | 400 mm/s2 |
| Max deceleration for Reach Out | 250 | 250 mm/s2 |
| Mode 1 max speed fwd | 9 | 8 km/h |
| Mode 1 max speed rev | 7 | 7 km/h |

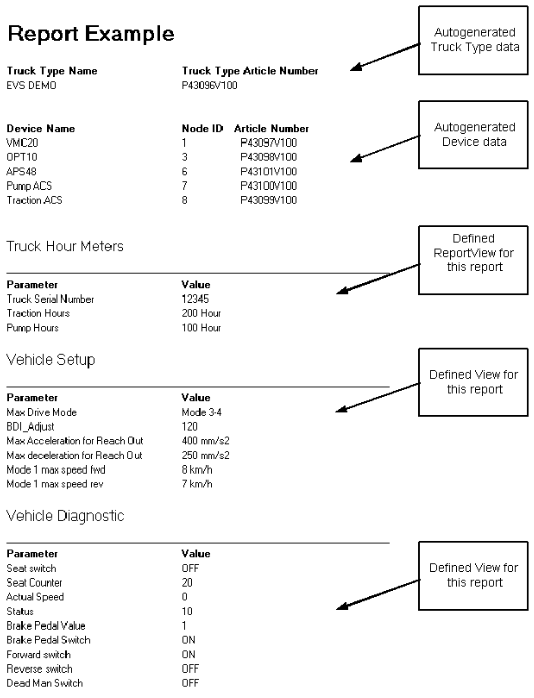
**Figure 6: View example**

Depending if the view is for setup (read/write) or diagnostics (read) the number of columns differ. To indicate difference between stored value in the PC and the read value from the vehicle the background color is changed to easy see that a parameter doesn't have original value.

## 11 Reports

A number of views can be gathered together to form a report. This is used to make printable reports or stored for later retrieval.

Typical use is service reports or the possibility to document settings in a vehicle from production.


Autogenerated Truck Type data
Autogenerated Device data
Defined ReportView for this report
Defined View for this report
Defined View for this report

**Report Example**

| Truck Type Name | Truck Type Article Number |
|---|---|
| EVS DEMO | P43096V100 |

| Device Name | Node ID | Article Number |
|---|---|---|
| VMC20 | 1 | P43097V100 |
| OPT10 | 3 | P43098V100 |
| APS48 | 6 | P43101V100 |
| Pump ACS | 7 | P43100V100 |
| Traction ACS | 8 | P43099V100 |

Truck Hour Meters

| Parameter | Value |
|---|---|
| Truck Serial Number | 12345 |
| Traction Hours | 200 Hour |
| Pump Hours | 100 Hour |

Vehicle Setup

| Parameter | Value |
|---|---|
| Max Drive Mode | Mode 3-4 |
| BDI_Adjust | 120 |
| Max Acceleration for Reach Out | 400 mm/s2 |
| Max deceleration for Reach Out | 250 mm/s2 |
| Mode 1 max speed fwd | 8 km/h |
| Mode 1 max speed rev | 7 km/h |

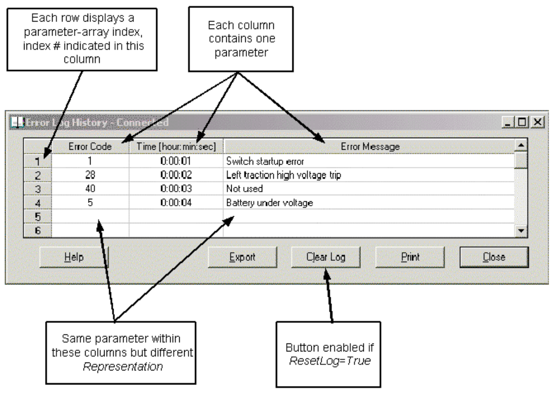
Vehicle Diagnostic

| Parameter | Value |
|---|---|
| Seat switch | OFF |
| Seat Counter | 20 |
| Actual Speed | 0 |
| Status | 10 |
| Brake Pedal Value | 1 |
| Brake Pedal Switch | ON |
| Forward switch | ON |
| Reverse switch | OFF |
| Dead Man Switch | OFF |

**Figure 7: Report example**

## 12 Logs

Another form of view is a log which is used to read array information. This is normally used to read typical error logs etc.


Each row displays a parameter-array index, index # indicated in this column
Each column contains one parameter
Error Log History - Connected
Help
Export
Clear Log
Print
Close
Same parameter within these columns but different *Representation*
Button enabled if *ResetLog=True*

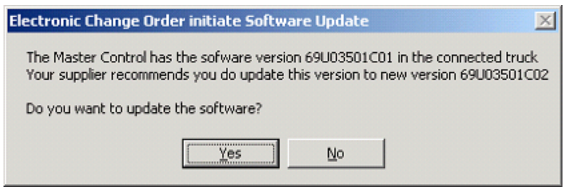
| | Error Code | Time [hour:min:sec] | Error Message |
|---|---|---|---|
| 1 | 1 | 0:00:01 | Switch startup error |
| 2 | 28 | 0:00:02 | Left traction high voltage trip |
| 3 | 40 | 0:00:03 | Not used |
| 4 | 5 | 0:00:04 | Battery under voltage |
| 5 | | | |
| 6 | | | |

**Figure 8: Log example (array information)**

The translation of a value to a understandable string can be very useful to read error log information to both present cause of error and also provide the user with possible actions.

## 13 Service wizard

To simplify some functions even more a way of using electronic change orders is implemented. It makes it possible to remove some manual actions from the user checking software versions for updates etc. This is done by writing a script.


Electronic Change Order initiate Software Update

The Master Control has the sofware version 69U03501C01 in the connected truck
Your supplier recommends you do update this version to new version 69U03501C02

Do you want to update the software?

**Figure 9: Electronic change order example**

Some functions in the service tool is available through the script engine to make maintenance work automated.

## 14 Flexibility for different users

To support functions for simple to more advanced users the software is handled with options to mainly avoid the software to be to complex for simple usage.

There is also a safety issue where for example a service engineer shall not have the possibility to upgrade software and not freely browse and edit the full object dictionary. This can be needed for design use.

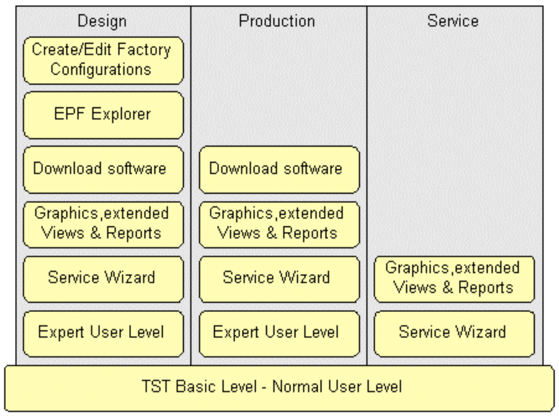

Design
Production
Service
Create/Edit Factory Configurations
EPF Explorer
Download software
Graphics,extended Views & Reports
Service Wizard
Expert User Level
Download software
Graphics,extended Views & Reports
Service Wizard
Expert User Level
Graphics,extended Views & Reports
Service Wizard
TST Basic Level - Normal User Level

**Figure 10: Possible user functions**

For the more experienced CANopen user an EPF Explorer-function is available. With this function the full object dictionary can be browsed. Reading and writing of parameters can be done without limitations used in other parts of the tool.

## 15 File integrity

To protect against any manipulation of the xml-files used by the tool a checksum/identification feature has been integrated. A file that has been manipulated will not be valid and the service tool will ignore the file and give an error message.

## 16 Distribution

An important feature is the way the software and the service application itself can be distributed.

Tools in the software for import of new service applications is included to make updates fast and efficient.

CD-distribution is simple as the installation process includes functions to include installation of software, drivers for different CAN-interfaces, service application (XML-files) and also possibility to add additional documentation as service manuals, technical support screens etc for the vehicle/system.

Customization with logos for end-user organization etc is also available.

## 17 Service application development

In parallel with the service tool a design tool has been developed to simplify the work creating the service application.

Using the product files as input the tool generates all needed files to have an service application up and running.

## 18 Summary/conclusion

The development and first time of usage of the service tool shows that we have created a flexible and scaleable platform but also that new requirements are added.

It includes better graphical interface to simplify the user interface even more and also more flexibility regarding communication interfaces and standards.

As CANopen is the standard in the bottom the possibility to use the tool for any type of CANopen system is possible.