acontis technologies GmbH

SOFTWARE

EC-EAP

EtherCAT® Automation Protocol Stack

Version 0.8

Edition: 2015-03-10

Content - compact

Content

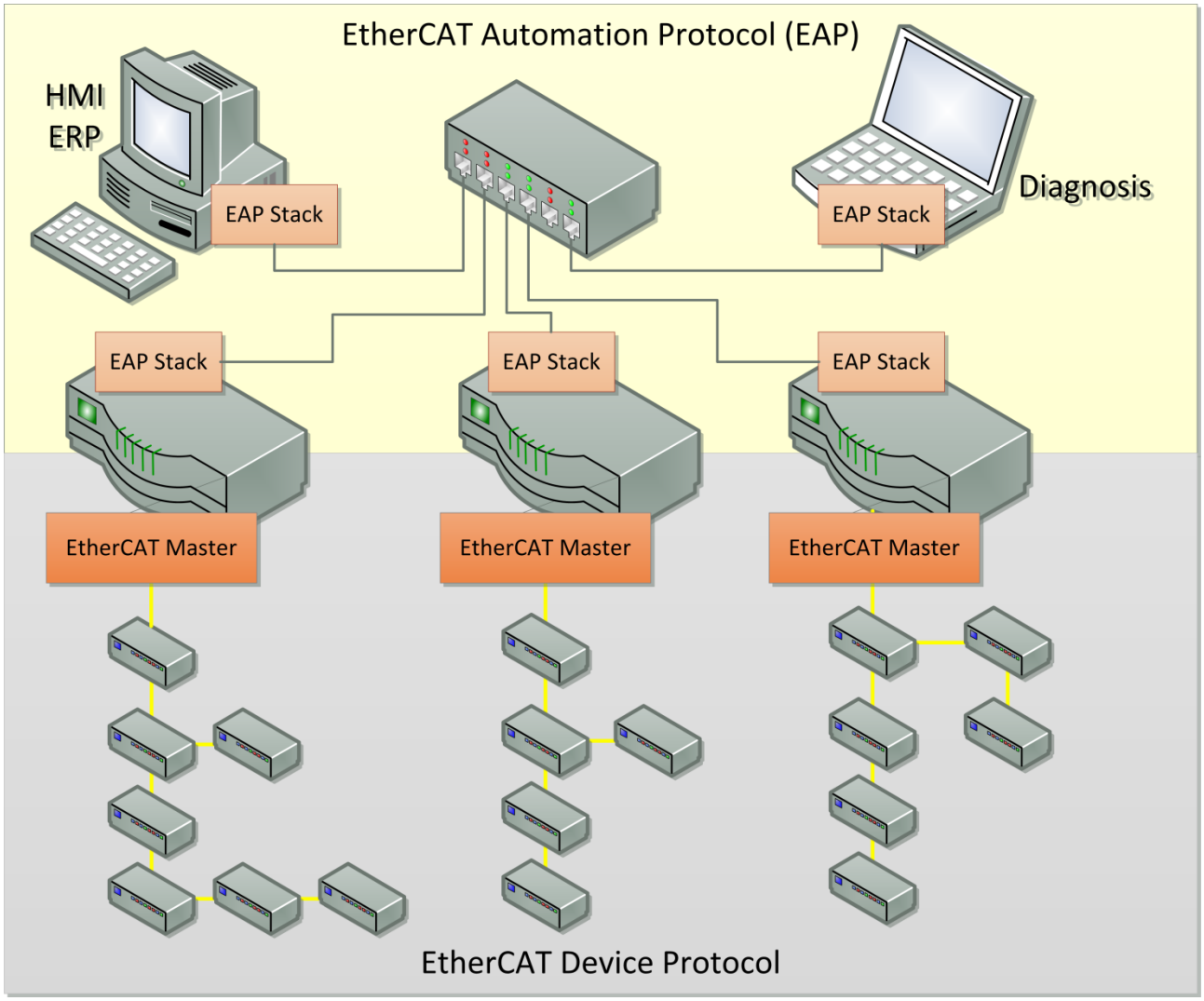
1 Introduction

1.1 EtherCAT Automation Protocol

The EtherCAT Device Protocol is used for EtherCAT segments in most cases with high real-time requirements. The EtherCAT Automation Protocol (EAP) specifies communication services and protocols for Master devices and Ethernet-based devices connected with a standard Ethernet network. These networks are used for example to connect machine parts or separate machines within plant automation. The timing constraints are weaker but a cyclic communication in the range of milliseconds is required as well.

The EAP can be used for different use-cases:

- Master-Master communication
- Configuration tool interface
- Monitoring
- General protocol for data exchange between devices



1.2 EC-EAP Features

The implementation is based on the EtherCAT Technology Group Standard “EtherCAT Automation Protocol ETG.1005 S (R) V1.0.0”. This document is available in the member area on the ETG website www.ethercat.org.

General functions

- Support for EtherCAT Device Configuration (EDC) file created by Beckhoff TwinCAT EAP Configurator
- EAP State Machine
- Performance measurement API
- Callback functions

Process Data Communication (Type 4)

- Process Data Frame Types
  - Via Ethernet (EtherType = 0x88A4)
  - Via UDP/IP (UDP Port = 0x88A4)
- Operation Modes
  - Pushed Data Exchange
  - Polled Data Exchange

EAP Object Dictionary (Sub-profile 1000)

- Process Variables
- Process Data
- Process Data Frames

1.3 Terms and Definitions

EtherCAT Device Protocol
EtherCAT for machine level control networks, which utilizes the special functional principle of "processing on the fly" as defined in [2] … [5]; EtherCAT Type 1 is used

EtherCAT Automation Protocol
EtherCAT for control systems and configuration interfaces as defined in this standard.

Process Variable (PV)
IO data of a device that can be mapped in to PDOs
(Tx-Variables 0x6000…0x6FFF, Rx-Variables 0x7000…0x7FFF)

Process Data Object (PDO)
Mapping of Process Variable(s) to process data
(RxPDO 0x1600…0x17FF, TxPDO 0x1A00…0x1BFF)

Process Data (PD)
Configuration of the PDO
(Tx Process Data 0xD000…0xDFFF, Rx Process Data 0xE000…0xEFFF)

Process Data Frame
Set of Process Data. Can be a Pushed Frame or a Polled Frame.
Corresponds to a SyncManager Area in the EtherCAT Device Protocol

AoE NetID
Unique 6 Byte Identifier. Usually extension of (4 Byte) IP-Address.

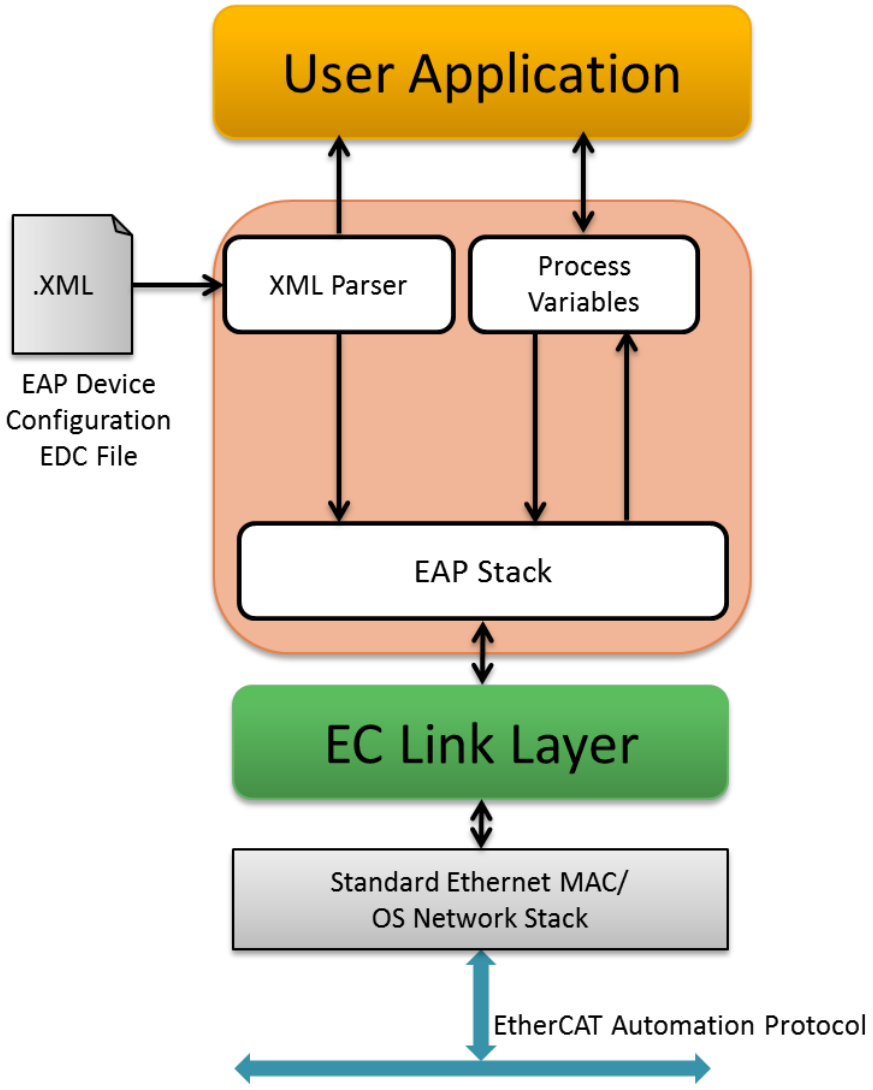
AoE Port
Number of an AoE device connected to an AoE Router.

AoE Address
Combination of AoE NetID and AoE Port.

2 Getting Started

2.1 Architecture

2.1.1 Component model



The EC-EAP Stack is implemented in C++ and can be easily ported to any embedded OS platforms using an appropriate C++ compiler. The API interfaces are C language interfaces, thus the EAP stack can be used in ANSI-C as well as in C++ environments.

The EAP Stack is divided into the following modules:

- Core
  In the core module the cyclic process data communication is performed. Among others in EAP stack state machine in executed. The EAP stack is configured using a XML file whose format is defined by the schema EapDeviceConfig.xsd. EC-EAP contains an OS independent XML parser.
- Ethernet Link Layer
  This layer exchanges Ethernet frames between EAP devices. If hard real-time requirements exist, this layer has to be optimized for the network adapter card in use.
- OS Layer
  All OS dependent system calls are encapsulated in a small OS layer. Most functions are that easy that they can be implemented using simple C macros.

2.1.2 Configuration

The EC-EAP stack has to know about the process data and the frames configuration in order to exchange data with other EAP devices. This configuration is determined in a configuration file which has to be available in the **EAP Device Configuration** (EDC) format. An EDC file can be prepared and exported i.e. in Beckhoff TwinCAT System Manager.

2.1.3 Process Data Communication

2.1.3.1 Operation Modes

The EAP stack's main task is to exchange process variables between other devices supporting the EtherCAT Automation Protocol. According to the configuration the process data are exchanged either in the operation mode **Pushed Data Exchange** or **Polled Data Exchange**.

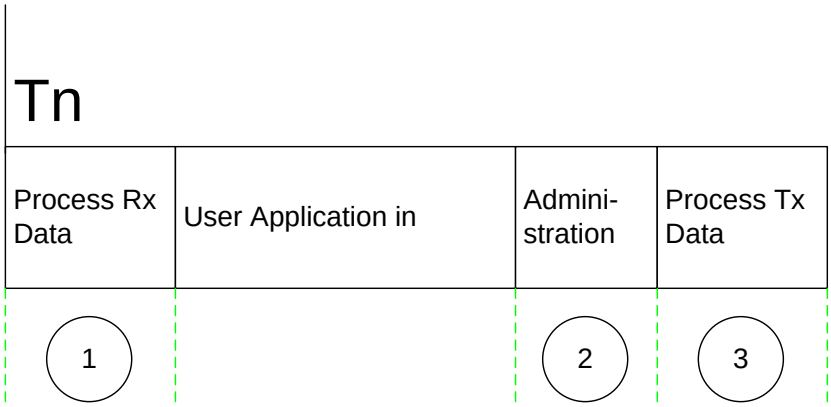
In Pushed Data Exchange mode each communication device sends one or several Process Data Frames with its Transmit PDs. These PDs should correspond to the Receive PDs of one or several communication partner in the network.

In Polled Data Exchange mode the Poll Client sends a PD with the output information to the Polled Server; the server responses with its input information in another PD.

2.1.3.2 Synchronization with user application

The EC-EAP stack operation is fully controlled by the user's application. Thus the user application is responsible for synchronization of the process data between the EC-EAP stack and the application itself. There are no internal tasks running inside the EC-EAP stack.

The timing of the necessary APIs within the "JobTask" is described in the following diagram.



Steps the application has to perform:

1. Process Rx Data: eapJobProcessRxData()
   Process all received EAP frames (e.g. read new input process data).
2. Administration: eapJobOnTimer()
   Cycle counters for the variables will be decremented, check link status, etc.
3. Process Tx Data: eapJobProcessTxData()
   The required process variables are assembled into a process data frame and are transmitted via the link layer.

2.1.3.3 Accessing process data in user application

For accessing the process variables the EC-EAP stack provides two different API sets:

- eapVarReadNoLock() and eapVarWriteNoLock() for accessing variables inside the "JobTask":
  There is no synchronization with the APIs eapJobXXX() implemented.

- eapVarRead() and eapVarWrite() for accessing variables from a different task:
  Accessing the variable data is synchronized with the APIs eapJobXXX() to prevent from inconsistent data.

Performance optimization hint:

If the user application has to update a lot of variables from a different task it's recommend to use this method:

- eapVarLock(): Protects variable access through eapJobXXX()
- zero to multiple eapVarWriteNoLock()
- zero to multiple eapVarReadNoLock()
- eapVarUnlock(): Enables variable access through eapJobXXX()

2.2 Application framework and example application

2.2.1 Overview

The example application EcEapDemo will handle the following tasks:

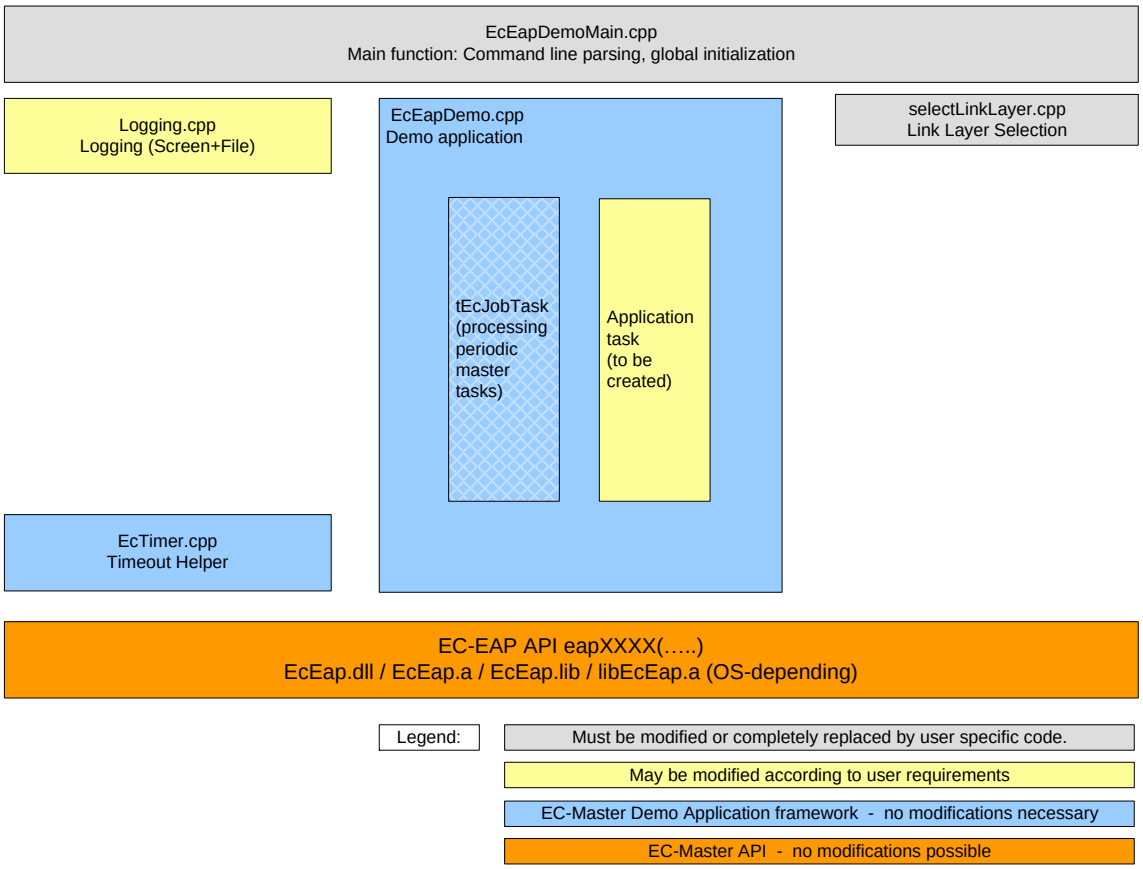
- Showing basic EC-EAP communication
- EC-EAP initialization
- ``Out of the box`` solution for Windows
- The output messages of the demo application will be printed on the console as well as in some files. The following log files will be created (the command line parameter '–log eap' must be specified):
  - error0.log application error messages (logged via LogError function)
  - eap0.log all messages

2.2.2 File reference

The EC-EAP stack application consists of the following files:

EcEapDemoMain.cpp	Entry point for the different operating systems and parsing of command line parameters
EcEapDemo.cpp	Initialize, start and terminate the EC-EAP (function EcEapDemo())
EcEapDemoConfig.h	Contains basic static configuration parameters
selectLinkLayer.cpp	Common Functions which abstract the command line parsing into Link Layer parameters
Logging.cpp	Message logging functions
EcTimer.cpp	Start and monitor timeouts

The following picture gives an overview of delivered files and their responsibility.



Picture 1: Files overview

2.2.3 Lifecycle

This chapter gives brief information about the starting and stopping of the EC-EAP.

Basically the operation of the EC-EAP is wrapped between the functions

- eapInit()
- eapConfigure()
- eapSetState()

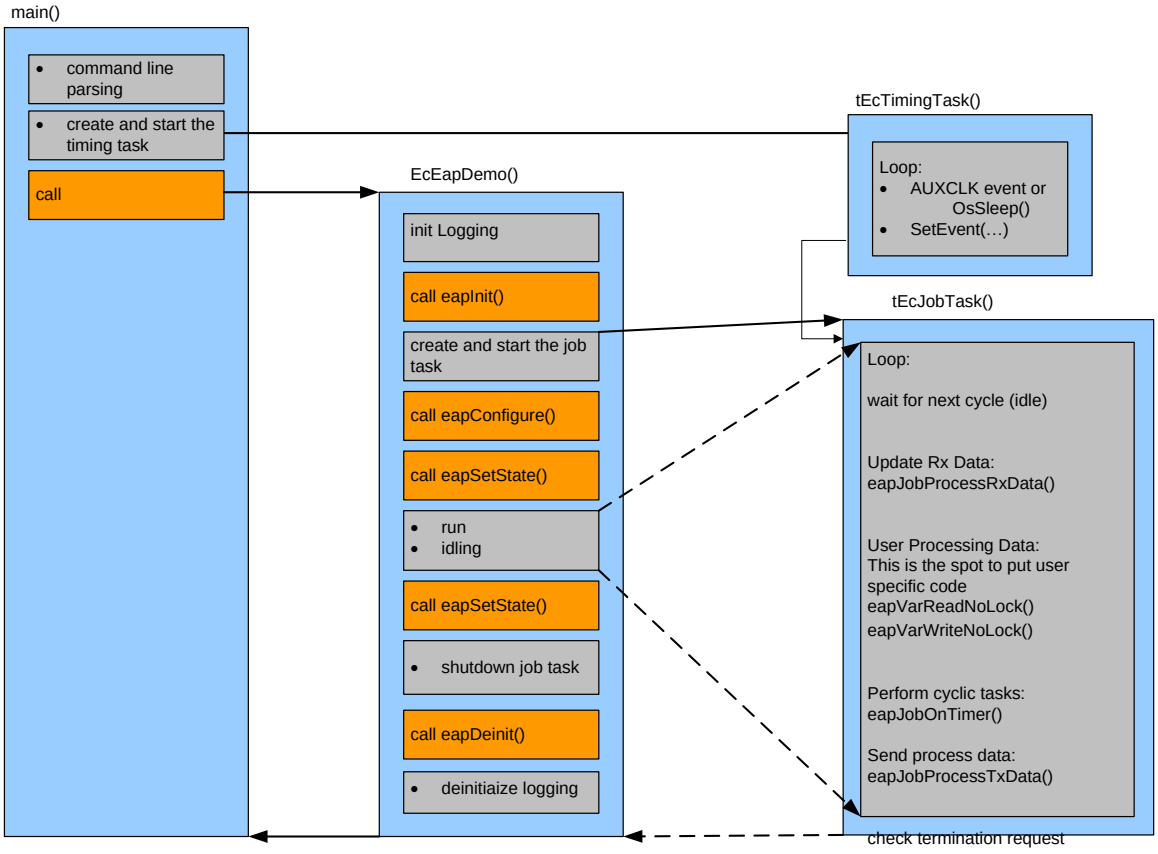
and

- eapDeinit()

With the former two functions the EC-EAP is prepared for operation and started. Also during this preparation there is a thread set up and started, which does all the cyclic duties of the EC-EAP.

With the last function, the EC-EAP is stopped and memory is cleaned up.

An overview of the complete lifecycle is given in Picture.



Picture 2: Demo application lifecycle

Here a closer description of the functions:

main()	Simply the wrapper to start the demo; probably dependent on the operating system. Implement command line parsing for individual parameter setting here.
EcEapDemo()	Demo application. The function takes care for starting and stopping the EC-EAP and all related tasks. In between there is a spot where the function does nothing (idling). During this idling all relevant work is done by the ``tEcJobTask()``, described below.
tEcJobTask()	Thread which does the necessary periodic work. Application specific manipulation of the process image, which must be synchronous with the network cycle, can be put in myAppWorkPd.
eapSetState()	EC-EAP API function: Startup the EC-EAP and switch the bus to the different states from INIT to OPERATIONAL.
eapInit()	EC-EAP API function: Prepare the EAP stack for operation and set operational parameters, e.g. used link layer, buffer sizes … .
eapDeinit()	EC-EAP API function: Clean up.
eapConfigure()	EC-EAP API function: Tell EC-EAP stack about its XML file configuration.

Remark: During cyclic operation, which is the desired OPERATIONAL state, the main work is done in the `` tEcJobTask()``. The originating process, in case of the demo this is ``EcEapDemo()``, is doing nothing (idling).

2.2.4 File logging

There are several functions which log information into files. In case the VxWorks netDrv driver is used (e.g. when accessing the PC hard disk via FTP) the content of these files is stored in memory. These files will be closed only when the application terminates or when the specified number of messages to be stored in a single file is exceeded. There are no means to limit the amount of memory needed for those files. If you extend the duration of the demo and the number of messages to be stored in the log file is too high it may occur that there is no free memory available and thus the system to crash.
Using the InitLogging() function the application can determine the rollover parameter when a new log file shall be created.
The file logging for VxWorks is disabled by default. Setting the global variable bLogFileEnb to a value of 1 the file logging is enabled.

2.2.5 Debug messages

Every time the EC-EAP detects an error it calls the OS-Layer function OsDbgMsg(). The demo application registers a hook OsDbgMsgHook() to be called by OsDbgMsg(). The hook will store those messages into a log file.

2.3 Setting up and run example EcEapDemo

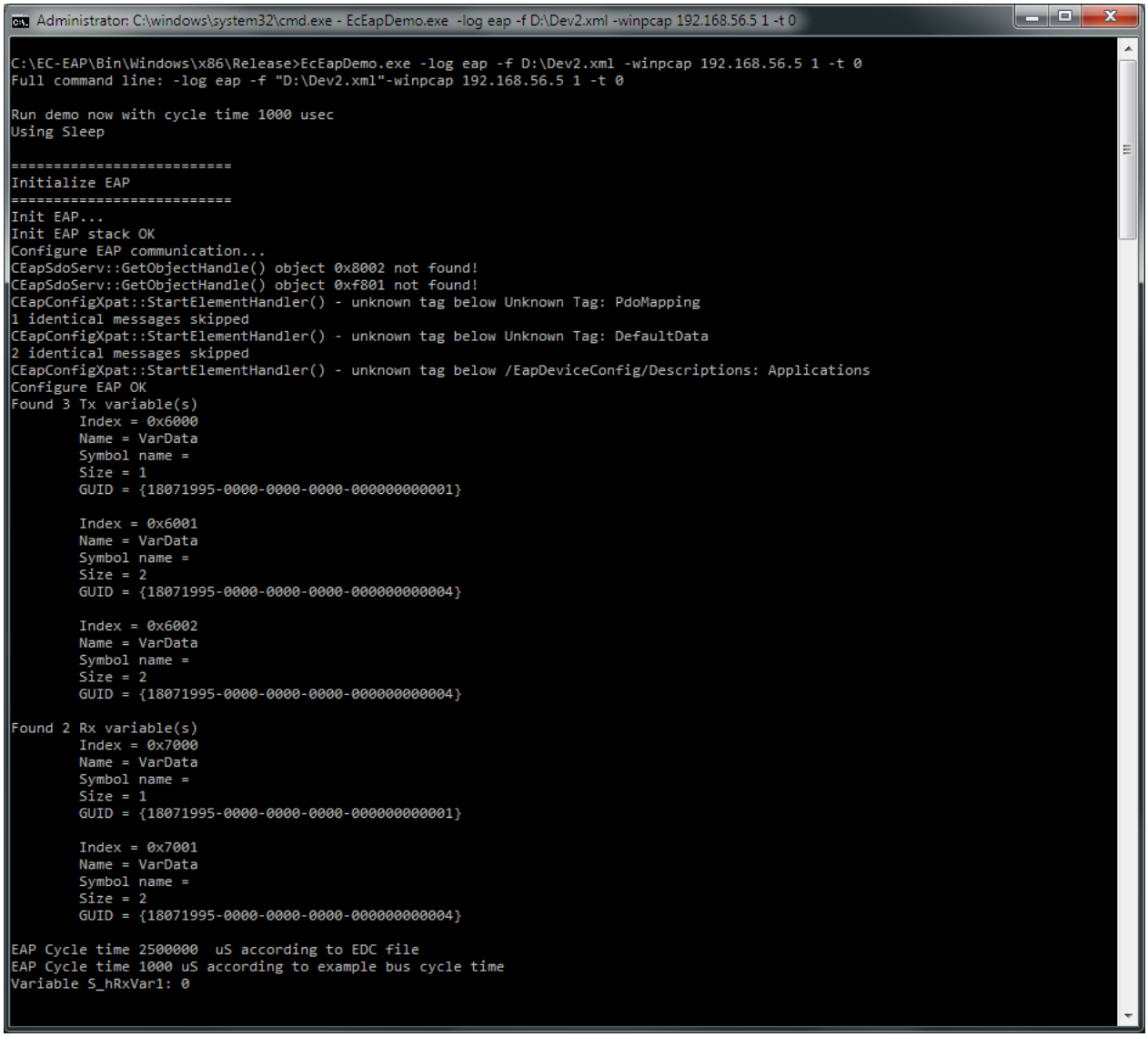
2.3.1 On Microsoft Windows 7/8

Step 1: Windows configuration
See the section Operating system configuration for how to prepare the operating system

Step 2: Determine the network interface
Using the command line option the network interface card used by the example application can be determined. For example the option `-winpcap 192.168.110.11` will be using the network adapter card with the IP address 192.168.110.11.

Step 3: Copy all of the example application files into one directory
The application EcEapDemo.exe together with the EC-EAP DLL EcEap.dll, the link-layer DLL emllPcap.dll and the configuration EDC file (for example the file eapconfig2.xml) have to be copied into one directory.

Step 4: Run the example application
The file EcEapDemo.exe has to be executed. The file name of the configuration file has to be given as a command line parameter as well as the appropriate link layer settings.
Example (starting the application with command line parameter `-f D:\Dev2.xml -winpcap 192.168.56.5 1 -t 0 -log eap`):



2.3.3 Command line parameters

EcEapDemo <Link Layer> [-f configFileName] [-t time] [-b time] [-v level] [-a affinity] [-perf] [-sp [port]]

e.g. EcEapDemo -winpcap 192.168.157.2 1 -f eapconf2.xml -t 0 -v 3

2.3.3.1 Link Layer

Using one of the following Link Layer options, the demo application will dynamically load the network driver for the specified network adapter card (e.g. –i8254x 1 1 for the Intel Pro/1000 network card). The EC-EAP stack then will use the appropirate network driver to access the Ethernet adapter.

- ***–winpcap <ipAddress> <mode>***
  <ipAddress>: IP address of network adapter card, e.g. 192.168.157.2
  <mode>: Mode 0 = Interrupt mode, 1= Polling mode
  Hardware: Hardware independent
  This parameter is only available for Windows.

2.3.3.2 Configuration file name

- ***–f <configFileName>***
  Determines which EAP configuration file to use

2.3.3.3 Further command line parameters

- ***–t <time>***
  <time>: Time in msec, 0 = forever (default = 30000)
  Specifies the duration the demo application runs. When the time expires the demo application exits completely.

- ***–b <cycle time>***
  <cycle time>: Bus cycle time in µsec
  Specifies the bus cycle time. Defaults to 1000µs (1ms).

- ***–v <level>***
  <level>: Verbosity level: 0=off (default), 1..n=more messages
  The verbosity level specifies how much console output messages will be generated by the demo application. A high verbosity level leads to more messages.

- ***–a <affinity>***
  <affinity>: 0 = first CPU, 1 = second, ...
  The CPU affinity specifies which CPU the demo application ought to use.

- ***–perf***
  Enable job measurement for all jobs (e.g. for the job JOB_ProcessAllRxFrames). It will be measured maximum and average time in micro seconds.

2.4 Evaluation version

Restriction of the evaluation version is:
Sending Ethernet frames terminates after 4 hours. Afterwards a timeout error will occur.

3 Programmer's Guide

3.1 Software Development Kit (SDK)

The EC-EAP development kit is needed to write applications based on the EAP stack. The EAP stack is shipped as a library which is linked together with the application.

The following components are supplied together with the SDK:

- Documentation (….\Doc)
- EC-EAP Software Development Kit (…\SDK) containing libraries and header files to build C/C++-applications.
  - ….\Bin: Executables containing the EAP stack
  - ….\SDK\INC: Header files to be included with the application
  - ….\SDK\LIB: Libraries to be linked with the application
  - ….\Sources\Common: Shared .cpp-files
- Examples (…\Examples)
  - …\Examples\EcEapDemo: Contains demo project source and headerfiles, shows the usage of the SDK
  - …\Workspace\WindowsVS2010\EcEapDemo: Contains the project file (.vcxproj) for the demo project
- TwinCAT Project (…\TwinCAT Demo Project) contains packed TwinCAT project used together with the EcEapDemo project

3.2 Accessing variables in application

The data, exchanged between EAP devices in every cycle, are stored in the process data area. Each process variable (data block) must be accessed via a handle wich has to be obtained prior using eapGetHandleByIndex() or eapGetHandleByName(). Afterwards the process variables can be written or read with or without memory lock depending on application logic. Moreover the memory lock can be issued or released separately before resp. after variable data access.

Pseudo code for memory locked access (internal memory lock issued and released during access in eapVarRead() or eapVarWrite())

```
eapGetHandleByIndex() or eapGetHandleByName(…)
…
eapVarRead() or eapVarWrite()
…
```

Pseudo code for memory locked access (external memory lock)

```
eapGetHandleByIndex() or eapGetHandleByName(…)
…
eapVarLock()
…
eapVarReadNoLock() or eapVarWriteNoLock()
…
eapVarUnlock()
…
```

Pseudo code for unlocked access

```
eapGetHandleByIndex() or eapGetHandleByName(…)
…
eapVarReadNoLock() or eapVarWriteNoLock()
…
```

3.4 Application Programming Interface (API) reference

Function prototypes, definitions etc. of the API can be found in the header file EcEap.h which is the main header file to include when using EC-EAP.

3.4.1 Generic API return status values

Most of the functions and also some notifications will return an error status value to indicate whether a function was successfully executed or not.
Some of the return status values have a generic meaning unspecific to the called API function.

EC_E_NOERROR	The function was successfully executed.
EC_E_NOTSUPPORTED	Unsupported feature or functionality.
EC_E_NOMEMORY	Not enough memory or frame buffer resources available.
EC_E_INVALIDPARM	Invalid or inconsistent parameters.
EC_E_TIMEOUT	Timeout error.
EC_E_LINK_DISCONNECTED	Link cable not connected.
EAP_E_CFGFILENOTFOUND	The configuration file (EDC) not found
EAP_E_WRONGSTATE	The operation requested (a function call) can not be performed in the actual EAP state (e.i. the frames can be transmitted only in OPERATIONAL state).
EAP_E_INITTRANSMITDATA	The trasmit data can not be initialized.

The EC_E_BUSY return status value indicates that a previously requested operation is still in progress. For example if the EAP stack is requested to enter the OPERATIONAL state the next request from the API will return this status value unless the OPERATIONAL state is entered.

3.4.2 General functions

3.4.2.1 eapInit

Initializes and starts the EAP stack. Gives back a stack handle, this handle will used in further functions calls.

```
EC_T_DWORD eapInit (
    EAP_T_INITPARMS          oParms,
    EC_T_HANDLE*             hPort
);
```

Parameters

oParms
[in] Initialization parameters structure (see also EAP_T_INITPARMS)

hPort
[out] EAP stack handle

Return

EC_E_NOERROR or error code

Comment

The EAP stack will be initialized by calling this function. This function has to be called prior to calling any other EAP functions.

EAP_T_INITPARMS

```
typedef struct _EAP_T_INITPARMS
{
   EAP_T_PROTOCOL_TYPE      eProtocolType;
   EC_T_WORD                wSocket;
   EC_T_LINK_PARMS*         poLinkParms;
} EAP_T_INITPARMS, *EAP_PT_INITPARMS;
```

eProtocolType
[in] Protocol type, possible values are:
eEapProtocol_Ethernet, EtherType 0x88A4
eEapProtocol_UDP, UPD Port 0x88A4
eEapProtocol_TCP, TCP Socket 0x88A4, not supported for now

wSocket
[in] Socket nummer, if omitted 0x88A4 will be used.

pvLinkParms
[in] Pointer to the link layer parameters (e.g. to determine which network adapter to use).The parameters are highly depending on the network adapter used by the link layer. More information can be found in chapter Link Layer selection and initialization.

3.4.2.2 eapDeinit

Terminates the EC-EAP stack and releases all resources.

```
EC_T_DWORD ecatDeinit(
        EC_T_HANDLE hPort,
);
```

Parameters

hPort
[in] EAP stack handle

Return

EC_E_NOERROR or error code

Comment
Stops the EC-EAP stack and releases all resources.
This function may not be called from within the JobTask's context.

3.4.2.3 eapConfigure

Configure the EC-EAP stack.

```
EC_T_DWORD eapConfigure(
        EC_T_HANDLE hPort,
        EAP_T_CONFIGPARM_DESC oConfig
);
```

Parameters
hPort
[in] EAP stack handle
oConfig
[in] Parameter description structure

EAP_T_CONFIGPARM_DESC

```
typedef struct _EAP_T_CONFIGPARM_DESC
{
   EAP_T_CNF_TYPE      eCnfType;
   EC_T_BYTE*          pbyData;
   EC_T_DWORD          dwLen;
} EAP_T_CONFIGPARM_DESC, *EAP_PT_CONFIGPARM_DESC;
```

eCnfType
[in] Type of configuration data, depending on this enum the data field is interpreted differently, possible values are:
eEapProtocol_Filename,a EDC (XML) file
pbyData
[in] Configuration data.
dwLen
[in] Legth of the configuration data in bytes.

Return
EC_E_NOERROR or error code

Comment
This function has to be called after *eapInit*.It reads the configuration file (EDC) and initiliazes internal storage and structures. This function may not be called from within the JobTask's context.

3.4.2.4 eapSetState

The EC-EAP stack (state machine of it) will be set into the requested state.

```
EC_T_DWORD eapSetState(
    EC_T_HANDLE     hPort,
    EC_T_STATE      eNewState,
    EC_T_DWORD      dwTimeout
);
```

Parameters
hPort
[in] EAP stack handle
eNewState
[in] Requested State.

eEapState_INIT EC-EAP state Init
eEapState_PREOP EC-EAP state pre-operational
eEapState_SAFEOP EC-EAP state safe operational
eEapState_OP EC-EAP state operational

dwTimeout
[in] Timeout in msec. This function will block until the requested state is reached or the timeout elapsed. If the timeout value is set to EC_NOWAIT the function will return immediately.

Return
EC_E_NOERROR or error code. If the last requested state was not reached, EC_E_BUSY will be returned and the request has to be repeated at a later time.

Comment
If the function is called with EC_NOWAIT, the client may wait for reaching the requested state using the notification callback (EC_NOTIFY_STATECHANGED)
This function may not be called from within the JobTask's context.

3.4.2.5 eapGetState
The current EC-EAP state is returned.

```
EC_T_STATE eapGetState(
    EC_T_HANDLE hPort,
    EAP_T_STATE* eState
);
```

Parameters
hPort
[in] EAP stack handle
eState
[in] Actual state.

Return
Current EC-EAP state (*eEapState_INIT*, *eEapState_PREOP*, *eEapState_SAFEOP* or *eEapState_OP*).

Comment
The actual EC-EAP state will be written into eState variable, therefore the memory has to be allocated prior function call.

3.4.2.6 eapGetVersion
Gets the EC-EAP software version as EC_T_DWORD.

```
ATECAT_API EC_T_DWORD ecatGetVersion(
        EC_T_DWORD* pdwVersion
);
```

Parameters
pdwVersion
[out] Pointer to EC_T_DWORD to carry out EC-EAP version.

Return
Always *EC_E_NOERROR*

3.4.2.7 eapGetCycleTime
Gets actual EC-EAP cycle time in µS.

ATECAT_API EC_T_DWORD eapGetCycleTime(
EC_T_HANDLE hPort,
EC_T_DWORD* dwCycleTime
);

Parameters
hPort
[in] EAP stack handle
dwCycleTime
[in] Actual cycle time in µS.

Return
Current EC-EAP cycle time in µS.

Comment
The actual EC-EAP state will be written into dwCycleTime variable, therefore the memory has to be allocated prior function call.

3.4.2.8 eapCalculateCycleTime
Calculates actual EC-EAP cycle time in µS.

ATECAT_API EC_T_DWORD eapCalculateCycleTime(
EC_T_HANDLE hPort,
EC_T_DWORD* dwCycleTime
);

Parameters
hPort
[in] EAP stack handle
dwCycleTime
[in] Actual cycle time in µS.

Return
Calculated EC-EAP cycle time in µS.

Comment
The calculated cycle time as greatest common divider; takes in account all timings of EC-EAP. The calculated time is optimal.

3.4.2.9 eapSetCycleTime
Sets a new EC-EAP cycle time in µS.

ATECAT_API EC_T_DWORD eapSetCycleTime(
EC_T_HANDLE hPort,
EC_T_DWORD dwCycleTime);

Parameters
hPort
[in] EAP stack handle
dwCycleTime
[in] New cycle time in µS.

Return
Sets a new EC-EAP cycle time in µS.

Comment
-

3.4.3 Process Data Communication

3.4.3.1 eapJobProcessRxData

Process all input (receive) frames: checks whether frames were received, filters received frames and stores the data into internal process data image.

```
EC_T_DWORD eapJobProcessRxData(
    EC_T_HANDLE hPort
);
```

Parameters

hPort

[in] EAP stack handle

Return

EC_E_NOERROR or error code

Important notes:

In order to maintain proper data communication it is important to call this function once each cycle preferably with cycle time returned by eapCalculateCycleTime(). An internal process data memory lock will be issued during the whole call.

Comment

The content of receive process varaibles will be changed after this function call and can be obtained via eapVarRead() and eapVarReadNoLock().

3.4.3.2 eapJobProcessTxData

Process all output (transmit) frames: checks whether frames must be sent, forms correct ethernet frames and sends them over selected network interface.

```
EC_T_DWORD eapJobProcessTxData(
    EC_T_HANDLE hPort
);
```

Parameters

hPort

[in] EAP stack handle

Return

EC_E_NOERROR or error code

Important notes:

In order to maintain proper data communication it is important to call this function once each cycle preferably with cycle time returned by eapCalculateCycleTime(). An internal process data memory lock will be issued during the whole call.

Comment

The content of transmit process varaibles can be changed by calling eapVarWrite() and eapVarWriteNoLock().

3.4.3.3 eapJobOnTimer

Performs cyclical tasks.

```
EC_T_DWORD eapJobOnTimer(
```

```
    EC_T_HANDLE hPort
);
```

Parameters
hPort
[in] EAP stack handle

Return
EC_E_NOERROR or error code

Important note:
In order to maintain proper data communication it is important to call this function once each cycle preferably with cycle time returned by eapCalculateCycleTime().

Comment
This function has to be call before eapJobProcessTxData() in order to select right process data to be sent.

3.4.3.4 eapGetNumOfTxVars

Gives back an amount of configured transmit variables (0x6nnn objects).

```
EC_T_DWORD ecatGetNumOfTxVars (
    EC_T_HANDLE hPort,
    EC_T_WORD* wNumOfVars
);
```

Parameters
hPort
[in] EAP stack handle
wNumOfVars
[out] The amount of transmit variables.

Return
EC_E_NOERROR or error code

Comment
The memory for *wNumOfVars* must be allocated prior function call. The value obtained can be used afterwards in allocating memory for eapGetTxVarList().

3.4.3.5 eapGetTxVarList

Get the list of configured transmit variables (0x6nnn objects).

```
EC_T_DWORD ecatGetTxVarList (
    EC_T_HANDLE hPort,
    EC_T_WORD* aIndexes
);
```

Parameters
hPort
[in] EAP stack handle
aIndexes
[out] Array with object's indexes of all configured transmit variables.

Return
EC_E_NOERROR or error code

Comment
The memory for *aIndexes* must be allocated prior function call. The amount of transmit variables can be obtained by calling eapGetNumOfTxVars().

3.4.3.6 eapGetNumOfRxVars

Gives back an amount of configured receive variables (0x7nnn objects).

```
EC_T_DWORD ecatGetNumOfRxVars (
    EC_T_HANDLE hPort,
    EC_T_WORD* wNumOfVars
);
```

Parameters

hPort
[in] EAP stack handle
wNumOfVars
[out] The amount of receive variables.

Return

EC_E_NOERROR or error code

Comment

The memory for *wNumOfVars* must be allocated prior function call. The value obtained can be used afterwards in allocating memory for eapGetRxVarList().

3.4.3.7 eapGetRxVarList

Get the list of configured receive variables (0x7nnn objects).

```
EC_T_DWORD ecatGetRxVarList (
    EC_T_HANDLE hPort,
    EC_T_WORD* aIndexes
);
```

Parameters

hPort
[in] EAP stack handle
aIndexes
[out] Array with object's indexes of all configured receive variables.

Return

EC_E_NOERROR or error code

Comment

The memory for *aIndexes* must be allocated prior function call. The amount of receive variables can be obtained by calling eapGetNumOfRxVars().

3.4.3.8 eapGetHandleByIndex

Gets the handle of a EAP process variable with given object index.

```
EC_T_DWORD eapGetHandleByIndex(
        EC_T_HANDLE hPort,
        EAP_VAR_HANDLE* hVar,
        EC_T_WORD wIndex
);
```

Parameters

hPort
[in] EAP stack handle

hVar
[out] Handle of a variable
wIndex
[in] Index of a corresponding EAP object in range from 0x600 to 0x6FFF or from 0x7000 to 0x7FFF

Return
EC_E_NOERROR or error code

Comment
The memory for *hVar* must be allocated prior function call. The handle returned can be used by further calls like eapVarRead() or eapVarWrite().

3.4.3.9 eapGetHandleByName
Gets the handle of a EAP process variable with given name.

```
EC_T_DWORD eapGetHandleByName(
        EC_T_HANDLE hPort,
        EAP_VAR_HANDLE* hVar,
        EC_T_CHAR *szName
);
```

Parameters
hPort
[in] EAP stack handle
hVar
[out] Handle of a variable
szName
[in] Name of a variable

Return
EC_E_NOERROR or error code

Comment
The memory for *hVar* has to be allocated prior function call. The handle returned can be used by further calls like eapVarRead() or eapVarWrite(). If more than one variable with given name exists, the first one with smallest object index will be taken.

3.4.3.10 eapGetVarInfo
Gets the process variable information.

```
EC_T_DWORD ecatGetVarInfo(
        EC_T_HANDLE hPort,
        EAP_VAR_HANDLE hVar,
        EAP_T_VAR_INFO* pVarInfo
) ;
```

Parameters
hPort
[in] EAP stack handle
hVar
[in] Handle of a variable
pVarInfo
[out] Information about variable

EC_T_ VAR_INFO

```
typedef struct _EAP_T_VAR_INFO
{
   EC_T_WORD         wSize;
   EC_T_WORD         wIndex;
   EC_T_CHAR         szName[256];
   EC_T_CHAR         szSymbolName[256];
   EC_T_GUID         GUID;
} EAP_T_VAR_INFO, *EAP_PT_VAR_INFO;
```

wSize
[out] Size of the variable in bytes
wIndex
[out] Index of the corresponding object.
szName
[out] Name of the variable.
szSymbolName
[out] Symbol name of the variable.
GUID
[out] Type of the variable as GUID.

Return
EC_E_NOERROR or error code

Comment
The memory for *pVarInfo* has to be allocated prior function call.

3.4.3.11 eapVarRead
Gets the content of an EAP process variable.

```
EC_T_DWORD eapVarRead(
        EC_T_HANDLE hPort,
        EAP_VAR_HANDLE hVar,
        EC_T_PVOID pData,
        EC_T_DWORD* dwDataLen
);
```

Parameters
hPort
[in] EAP stack handle
hVar
[in] Handle of a variable
pData
[out] Buffer
dwDataLen
[in,out] Buffer size (in), bytes proceeded (out)

Return
EC_E_NOERROR or error code

Comment
This function copies the content of an EAP process variable into buffer *pData*, its memory has to be allocated prior. The parameter *dwDataLen* as input contains the buffer size, as output gives back the real amount of data written into buffer, usually both values are the same.
Important. An internal memory lock will be issued and released during memory access.

3.4.3.12 eapVarWrite
Sets the content of an EAP process variable.

```
EC_T_DWORD  eapVarWrite(
        EC_T_HANDLE hPort,
        EAP_VAR_HANDLE hVar,
        EC_T_PVOID pData,
        EC_T_DWORD* dwDataLen
);
```

Parameters

hPort
[in] EAP stack handle
hVar
[in] Handle of a variable
pData
[in] Buffer
dwDataLen
[in,out] Buffer size (in), bytes proceeded (out)

Return

EC_E_NOERROR or error code

Comment

This function copies the content of the input buffer into an EAP process variable. The parameter *dwDataLen* as input contains the buffer size, as output gives back the real amount of data written into the process variable.
Important. An internal memory lock will be issued and released during memory access.

3.4.3.13 eapVarReadNoLock

Gets the content of an EAP process variable, no memory lock will be issued.

```
EC_T_DWORD eapVarReadNoLock(
        EC_T_HANDLE hPort,
        EAP_VAR_HANDLE hVar,
        EC_T_PVOID pData,
        EC_T_DWORD* dwDataLen
);
```

Parameters

hPort
[in] EAP stack handle
hVar
[in] Handle of a variable
pData
[out] Buffer
dwDataLen
[in,out] Buffer size (in), bytes proceeded (out)

Return

EC_E_NOERROR or error code

Comment

This function copies the content of an EAP process variable into buffer *pData*, its memory has to be allocated prior. The parameter *dwDataLen* as input contains the buffer size, as output gives back the real amount of data written into buffer, usually both values are the same.
Important. There will be no internal memory issued/released during memory access. To protect the memory against unexpected changes in a multithreading environment an external memory lock has to be issued/released, please see API functions eapVarLock() and eapVarUnlock().

3.4.3.14 eapVarWriteNoLock

Sets the content of an EAP process variable, no memory lock will be issued.

```
EC_T_DWORD  eapVarWriteNoLock(
        EC_T_HANDLE hPort,
        EAP_VAR_HANDLE hVar,
        EC_T_PVOID pData,
        EC_T_DWORD* dwDataLen
);
```

Parameters

hPort
[in] EAP stack handle

hVar
[in] Handle of a variable

pData
[in] Buffer

dwDataLen
[in,out] Buffer size (in), bytes proceeded (out)

Return

EC_E_NOERROR or error code

Comment

This function copies the content of the input buffer into an EAP process variable. The parameter *dwDataLen* as input contains the buffer size, as output gives back the real amount of data written into the process variable.
Important. There will be no internal memory issued/released during memory access. To protect the memory against unexpected changes in a multithreading environment an external memory lock has to be issued/released, please see API functions eapVarLock() and eapVarUnlock().

3.4.3.15 eapVarLock

Issues a memory lock during further reading/writing operations.

```
EC_T_DWORD  eapVarLock(
        EC_T_HANDLE hPort
);
```

Parameters

hPort
[in] EAP stack handle

Return

EC_E_NOERROR or error code

Comment

The memory lock can be used in order to protect memory against unexpected changes especially in multithreading evironements.
Important. Memory lock will also prevent the EC-EAP core code from accessing memory, thus it can lead to possible data loss. Please, keep memory locking as short as possible.
To release locked memory the API function eapVarUnlock() will be used.

3.4.3.16 eapVarUnLock

Releases last memory lock issued by eapVarLock().

```
EC_T_DWORD  eapVarUnLock(
        EC_T_HANDLE hPort
);
```

Parameters
hPort
[in] EAP stack handle

Return
EC_E_NOERROR or error code

Comment
The memory lock can be used in order to protect memory against unexpected changes especially in multithreading evironements.
Important. Memory lock will also prevent the EC-EAP core code from accessing memory, thus it can lead to possible data loss. Please, keep memory locking as short as possible.

3.4.3.17 eapVarRegisterCallback
Register a callback function to be called each time the variable given changes its content.

```
EC_T_DWORD  eapVarRegisterCallback(
        EC_T_HANDLE hPort,
        EAP_VAR_HANDLE hVar,
        EAP_T_PFVAR_CB fnCallback,
        EC_T_VOID* pvContext
);
```

Parameters
hPort
[in] EAP stack handle
hVar
[in] Handle of a variable
fnCallback
[in] Callback function
pvContext
[in] Data to be transferred

```
typedef EC_T_VOID (*EAP_T_PFVAR_CB)(EC_T_VOID* Context, EC_T_HANDLE hPort,
        EAP_VAR_HANDLE hVar);
```

Return
EC_E_NOERROR or error code

Comment
A callback function will be called each time whenever the content of the EAP variable will be changed. The parameter *pvContext* will be transferred each time the callback function will be invoked. To unregister a callback function the parameter fnCallback has to be set to EC_NULL .

3.4.4 Mailbox Communication

Currently not supported.

3.4.5 Diagnosis und error detection

3.4.5.1 Introduction

Return values of API functions can be used as the main error diagnosis source. The API functions return specific EAP error codes (prefix EAP_E_....) and common error codes (i.e. with prefix EC_E_...) as well. For more detailed errors description and explaination see Error Groups and Error Codes in Appendix.

3.4.5.2 eapGetText

Return text tokens by ID from EC-EAP stack.

EC_T_CHAR* ecatGetText(EC_T_WORD wTextId);

Parameters

wTextId

[in] Text enumeration ID

Return

EC-EAP stack stored text. To find the subordinate texts see source code in files EcError.h and EcEap.h.

3.4.5.3 ecatPerfMeasInit

Initialize performance measurement.

```
EC_T_VOID ecatPerfMeasInit(
    EC_T_TSC_MEAS_DESC*   pTscMeasDesc,
    EC_T_UINT64           dwlFreqSet,
    EC_T_DWORD            dwNumMeas,
    EC_T_FNMESSAGE        pfnMessage
);
```

Parameters

pTscMeasDesc
[in,out] measurement descriptor
dwlFreqSet
[in] TSC frequency, 0: auto-calibrate
dwNumMeas
[in] number of elements to be allocated in in pTscMeasDesc->aTscTime
pfnMessage
[in] Reserved. Set to EC_NULL.

Return

-

3.4.5.4 ecatPerfMeasDeinit

De-initialize performance measurement.

```
EC_T_VOID ecatPerfMeasDeinit(
    EC_T_TSC_MEAS_DESC*   pTscMeasDesc
);
```

Parameters

pTscMeasDesc
[in,out] measurement descriptor

Return

-

3.4.5.5 ecatPerfMeasEnable

Enable performance measurement.

```
EC_T_VOID ecatPerfMeasEnable(
    EC_T_TSC_MEAS_DESC*   pTscMeasDesc
);
```

Parameters

pTscMeasDesc
[in,out] measurement descriptor

Return

-

3.4.5.6 ecatPerfMeasDisable

Disable performance measurement.

```
EC_T_VOID ecatPerfMeasDisable(
    EC_T_TSC_MEAS_DESC*    pTscMeasDesc
);
```

Parameters

pTscMeasDesc
[in,out] measurement descriptor

Return

-

3.4.5.7 ecatPerfMeasStart

Start measurement.

```
EC_T_VOID ecatPerfMeasStart(
    EC_T_TSC_MEAS_DESC*    pTscMeasDesc,
    EC_T_DWORD             dwIndex
);
```

Parameters

pTscMeasDesc
[in,out] measurement descriptor
dwIndex
[in] measurement index

Return

-

3.4.5.8 ecatPerfMeasEnd

Stop measurement.

```
EC_T_TSC_TIME* ecatPerfMeasEnd(
    EC_T_TSC_MEAS_DESC*    pTscMeasDesc,
    EC_T_DWORD             dwIndex
);

typedef struct _EC_T_TSC_TIME
{
   EC_T_UINT64     qwStart;          /* start time */
   EC_T_UINT64     qwEnd;            /* end time */
   EC_T_DWORD      dwCurr;           /* [1/10 usec] */
   EC_T_DWORD      dwMax;            /* [1/10 usec] */
   EC_T_DWORD      dwAvg;            /* [1/100 usec] */
   EC_T_BOOL       bMeasReset;       /* EC_TRUE if measurement values shall be reset */
   EC_T_INT        nIntLevel;        /* for interrupt lockout handling */
} EC_T_TSC_TIME;
```

Parameters

pTscMeasDesc
[in,out] measurement descriptor
dwIndex
[in] measurement index

Return

Pointer to corresponding time descriptor.

3.4.5.9 ecatPerfMeasReset

Request measurement reset. Reset is done within ecatPerfMeasEnd().

```
EC_T_VOID ecatPerfMeasReset(
    EC_T_TSC_MEAS_DESC*   pTscMeasDesc,
    EC_T_DWORD            dwIndex
);
```

Parameters

pTscMeasDesc
[in,out] measurement descriptor
dwIndex
[in] measurement index, 0xFFFFFFFF: all indexes

Return

-

3.4.5.10 ecatPerfMeasShow

Log current performance measurement values using OsDbgMsg.

```
EC_T_VOID ecatPerfMeasShow(
    EC_T_TSC_MEAS_DESC*   pTscMeasDesc,
    EC_T_DWORD            dwIndex,
    EC_T_CHAR**           aszMeasCaption
);
```

Parameters

pTscMeasDesc
[in] measurement descriptor
dwIndex
[in] measurement index, 0xFFFFFFFF: all indexes
aszMeasCaption
[in] captions as array of zero terminated strings.

Return

-

4 Software Integration

4.1 Operating system configuration

The operating system used has to be prepared first for usage with the EC-EAP stack. The main task is to include the appropriate network adapter support in the operating system. Only network adapters which support at least 100 MBit are supported.

4.1.1 Microsoft Windows 7/8

4.1.1.1 WinPCap link layer

Using Windows XP it is recommended to use a separate network adapter to connect EtherCAT devices. If the main network adapter is used for both EtherCAT devices and the local area network there may be a main impact on the local area network operation.
The network adapter card used by EtherCAT has to be set to a fixed private IP address, e.g. 192.168.x.y.

4.1.1.2 Optimized link layers

To use the optimized link layers under Windows, it is necessary to install the EcatDrv driver included in the optimized link layer delivery.

4.1.3 Compiler settings

4.1.3.1 General

For all operating systems the same principal rules to generate the example applications can be used.

4.1.3.1.1 Include search path

The header files are located in the following two directories:

a) `<InstallPath>\SDK\INC\<OS>\<ARCH>` (where <OS> is a placeholder for the operating system and <ARCH> for the architecture if different archtitectures are supported)

b) `<InstallPath>\SDK\INC`

c) `<InstallPath>\Sources\Common`

4.1.3.1.2 Preprocessor macro

The demo applicatons are the same for all operating systems. The appropriate pre-processor macro has to be set for the operating system (for example `VXWORKS`, `LINUX`, `__INTIME__`, `__TKERNEL`,...).
For big endian systems the macro `EC_BIG_ENDIAN` has to be defined.

4.1.3.1.3 Libraries

The libraries located in `<InstallPath>\SDK\LIB\<OS>\<ARCH>` have to be added (<OS> is a placeholder for the operating system used and <ARCH> for the architecture if different archtitectures are supported).

4.1.3.2 Windows settings

The following settings are necessary to build the example application for Windows XP:

- Preprocessor definitions:
  `WIN32` (implicitely defined by VC 6.0)

- Library path of components:
  `<InstallPath>/SDK/LIB/Windows/x86`

- Include path:
  `<InstallPath>/SDK/INC/Windows`
  `<InstallPath>/SDK/INC`
  `<InstallPath>/Sources/Common`

The following pre-processor settings are required for the compiler:

- If a debug build shall be made the pre-processor macro DEBUG has to be defined (e.g. –DDEBUG for the gnu compiler)
- For **Windows XP** support WIN32 has to be defined
- For **Windows CE** support WIN32 and UNDER_CE have to be defined

Link Layer Binaries: the following include search paths have to be set

- Path to the SDK header files
- Path to the private header files
- Path to the link layer header files
- Path to the link layer sources

EAP Stack Binaries: the following include search paths have to be set

- Path to the SDK header files
- Path to the private header files
- Path to the OS layer header files

5 Appendix

5.1 Error Groups

Nr.	Group	Abbreviation	Description
1	Application Error	APP	Error within application, running the master. e.g.. API Function call with invalid parameters
2	EtherCAT network information file problem	ENI	Master configuration XML file mismatches slave configuration on bus. e.g.. Bus Topology Scan cannot detect all slaves configured within network information file.
3	Master parameter configuration	CFG	Master configuration parameters erroreous. e.g.. mailbox command queue not large enough
4	Bus/Slave Error	SLV	Slave error e.g.. Working Counter Error
5	Link Layer	LLA	LinkLayer errror (network interface driver). e.g.. Intel Pro/1000 NIC could not be found.
6	Remote API	RAP	Remote API error. e.g. Connection to Remote API server is not possible from client-
7	Internal software error	ISW	Master internal error e.g. Master state machine in undefined state.
8	DC Master Sync	DCM	DC slave and host time synchronization.
9	Pass-Through-Server	PTS	Initialisation/De-Initialisation errors
10	EAP stack	EAP	Errors from EC-EAP stack

5.2 Error Codes

5.2.1 Generic Error Codes

Code / Define	Text	Group	Possible error cause
0x00000000 EC_E_NOERROR	No Error	n. a.	Function call successful.
0x98110001 EC_E_NOTSUPPORTED	Feature not supported	APP	Function or property not available
0x98110002 EC_E_INVALIDINDEX	Invalid Index	APP	CoE: invalid SDO index.
0x98110003 EC_E_INVALIDOFFSET	Invalid Offset	ISW	Invalid offset, while accessing Process Data Image
0x98110005 EC_E_INVALIDSIZE	Invalid Size	APP	Invalid size - while accessing Process Data Image - while storing data
0x98110006 EC_E_INVALIDDATA	Invalid Data	ISW	Multiple error sources
0x98110007 EC_E_NOTREADY	Not ready	ISW	Multiple error sources
0x98110008 EC_E_BUSY	Busy	APP	Stack is busy currently and not available to process the API request. The function may be called again later.
0x98110009 EC_E_ACYC_FRM_FREEQ_EMPTY	Cannot queue acyclic ecat command	ISW	Acyclic command queue is full. Possible solution: Increase of configuration value *dwMaxQueuedEthFrames*
0x9811000A EC_E_NOMEMORY	No Memory left	CFG	Not enough allocatable memory available (memory full / corrupted).
0x9811000B EC_E_INVALIDPARM	Invalid Parameter	APP	API function called with erroneous parameter set.
0x9811000C EC_E_NOTFOUND	Not Found	APP	Network Information File not found or API called with invalid SlaveID.
0x9811000E EC_E_INVALIDSTATE	Invalid State	ISW	Multiple error sources
0x9811000F EC_E_TIMER_LIST_FULL	Cannot add slave to timer list	ISW	Slave timer list full.
0x98110010 EC_E_TIMEOUT	Timeout		Multiple error sources.

Code / Define	Text	Group	Possible error cause
0x98110011 EC_E_OPENFAILED	Open Failed	ISW	Multiple error sources.
0x98110012 EC_E_SENDFAILED	Send Failed	LLA	Transmit of frame failed.
0x98110013 EC_E_INSERTMAILBOX	Insert Mailbox error	CFG	Mailbox command couldn't be stored to internal command queue. Possible solution: Increas of configurationvalue *dwMaxQueuedCoeCmds*
0x98110014 EC_E_INVALIDCMD	Invalid Command	ISW	Unknown mailbox command code.
0x98110015 EC_E_UNKNOWN_MBX_PROTOCOL	Unknown Mailbox Protocol Command	ISW	Unknown Mailbox protocol or mailbox command with unknown protocol association.
0x98110016 EC_E_ACCESSDENIED	Access Denied	ISW	Master internal software error:
0x9811001A EC_E_PRODKEY_INVALID	Product Key Invalid	CFG	Application is using evaluation version of stack, which stops operation after 30 minutes.
0x9811001B EC_E_WRONG_FORMAT	Wrong configuration format	ENI	Network information file is empty or malformed.
0x9811001C EC_E_FEATURE_DISABLED	Feature disabled	APP	Application tried to perform a missing or disabled API function.
0x9811001E EC_E_BUSCONFIG_MISMATCH	Bus Config Mismatch	ENI	Network information file and currently connected bus topology does not match.
0x9811001F EC_E_CONFIGDATAREAD	Error reading config file	ENI	Network information file could not be read.
0x98110021 EC_E_XML_CYCCMDS_MISSING	Cyclic commands are missing	ENI	Network information file does not contain cyclic commands.
0x98110022 EC_E_XML_ALSTATUS_READ_MISSING	AL_STATUS register read missing in XML file for at least one state	ENI	Read of AL Status register is missing in cyclic part of given network information file.
0x98110023 EC_E_MCSM_FATAL_ERROR	Fatal internal McSm	ISW	Master control state machine is in an undefined state.
0x98110024 EC_E_SLAVE_ERROR	Slave error	SLV	Cannot address slave (no station address configured or slave absent).
0x98110025 EC_E_FRAME_LOST	Frame lost, IDX mismatch	SLV	An EtherCAT frame was lost on bus segment, means the response was not received. In case this error shows frequently a problem with the wiring could be the cause.

Code / Define	Text	Group	Possible error cause
0x98110026 EC_E_CMD_MISSING	At least one EtherCAT command is missing in the received frame	SLV	Received EtherCAT frame incomplete.
0x98110028 EC_E_INVALID_DCL_MODE	IOCTL EC_IOCTL_DC_LATCH_REQ_LTIMV ALS invalid in DCL auto read mode	APP	This function cannot be used if DC Latching is running in mode „Auto Read“.
0x98110029 EC_E_AI_ADDRESS	Auto increment address increment mismatch	SLV	Network information file and bus topology doesn’t match any more. Error shows only, if a already recognized slave isn’t present any more.
0x9811002A EC_E_INVALID_SLAVE_STATE	Slave in invalid state, e.g. not in OP (API not callable in this state)	APP	Mailbox commands are not allowed in current slave state.
0x9811002B EC_E_SLAVE_NOT_ADDRESSABLE	Station address lost (or slave missing) - FPRD to AL_STATUS failed	SLV	Slave had a powercycle.
0x9811002C EC_E_CYC_CMDS_OVERFLOW	Too many cyclic commands in XML configuration file	ENI	Error while creating network information file within configuration utility.
0x9811002D EC_E_LINK_DISCONNECTED	Ethernet link cable disconnected	SLV	EtherCAT bus segment not connected to network interface.
0x9811002E EC_E_MASTERCORE_INACCESSIBLE	Master core not accessible	RAP	Connection to remote server was terminated or master instance has been stopped on remote side.
0x9811002F EC_E_COE_MBXSND_WKC_ERROR	COE mbox send: working counter	SLV	CoE mailbox couldn’t be read on slave, slave didn’t read out mailbox since last write.
0x98110030 EC_E_COE_MBXRCV_WKC_ERROR	COE mbox receive: working counter	SLV	CoE Mailbox couldn’t be read from slave.
0x98110031 EC_E_NO_MBX_SUPPORT	No mailbox support	APP	Slave does not support mailbox access.
0x98110032 EC_E_NO_COE_SUPPORT	CoE protocol not supported	ENI	Configuration error or slave information file doesn’t match slave firmware.
0x98110033 EC_E_NO_EOE_SUPPORT	EoE protocol not supported	ENI	Configuration error or slave information file doesn’t match slave firmware.
0x98110034 EC_E_NO_FOE_SUPPORT	FoE protocol not supported	ENI	Configuration error or slave information file doesn’t match slave firmware.
0x98110035 EC_E_NO_SOE_SUPPORT	SoE protocol not supported	ENI	Configuration error or slave information file doesn’t match slave firmware.
0x98110036 EC_E_NO_VOE_SUPPORT	VoE protocol not supported	ENI	Configuration error or slave information file doesn’t match slave firmware.

Code / Define	Text	Group	Possible error cause
0x98110037 EC_E_EVAL_VIOLATION	Configuration violates Evaluation limits	ENI	Network information file contains configuration data for more slaves than allowed, while using evaluation version of stack.
0x98110038 EC_E_EVAL_EXPIRED	Evaluation Time limit reached	CFG	Time limit (30minutes) of evaluation version is reached, hence master stack is stopped.
0x98110070 EC_E_CFGFILENOTFOUND	Master configuration not found	CFG	The path to the master configuration file (XML) was wrong or the file is not available.
0x98110071 EC_E_EEPROMREADERROR	Command error while EEPROM upload	SLV	Could not read from slave EEPROM.
0x98110072 EC_E_EEPROMWRITEERROR	Command error while EEPROM download	SLV	Could not write to slave EEPROM.
0x98110073 EC_E_XML_CYCCMDS_SIZEMISMATCH	Cyclic command wrong size (too long)	CFG	Error while creating a new cyclic command. The size which was defined in the master configuration xml does not match to the size of the process data.
0x98110123 EC_E_VOE_MBX_WKC_ERROR	VoE mailbox send: working counter	SLV	VoE mailbox couldn't be written.
0x98110124 EC_E_EEPROMASSIGNERROR	EEPROM assignment failed	SLV	Assignment of the EEPROM to the slave went wrong.
0x98110125 EC_E_MBX_ERROR_TYPE	Error mailbox received	SLV	Unknown mailbox error code received in mailbox
0x98110126 EC_E_REDLINEBREAK	Redundancy line break	SLV	Cable break between slaves or between master and first slave
0x98110127 EC_E_XML_INVALID_CMD_WITH_RED	Invalid EtherCAT cmd in cyclic frame with redundancy	ENI	BRW commands are not allowed with redundancy. LRW commands with an expected WKC>3 are not allowed with redundancy (Workaround: Use LRD/LWR instead of LRW)
0x98110128 EC_E_XML_PREV_PORT_MISSING	<PreviousPort>-tag is missing!	ENI	If the auto increment address is not the first slave on the bus we check if a previous port tag OR a hot connect tag is available
0x98110130 EC_E_DLSTATUS_IRQ_TOPOCHANGED	Data link (DL) status interrupt because of changed topology	SLV	Handled inside the master
0x98110131 EC_E_PTS_IS_NOT_RUNNING	The Pass Through Server is not running!	PTS	The Pass-Through-Server was tried to be enabled/disabled or stopped without being started.

Code / Define	Text	Group	Possible error cause
0x98110132 EC_E_PTS_IS_RUNNING	The Pass Through Server is running!	PTS	The Pass-Through-Server was started and enabled. But in this state no API call is allowed.
0x98110133 EC_E_PTS_THREAD_CREATE_FAILED	Could not start the Pass Through Server!	PTS	The Pass-Through-Server could not be started.
0x98110134 EC_E_PTS_SOCK_BIND_FAILED	The Pass Through Server could not bind the IP address with a socket!	PTS	Possibly because the IPaddress (and Port) is already in use or the IP-address does not exist.
0x98110135 EC_E_PTS_NOT_ENABLED	The Pass Through Server is running but not enabled	PTS	-
0x98110136 EC_E_PTS_LL_MODE_NOT_SUPPORTED	The LinkLayer mode is not supported by the Pass Through Server!	PTS	The Master is running in interrupt mode but the Pass-Through-Server only supports polling mode.
0x98110137 EC_E_VOE_NO_MBX_RECEIVED	No VoE mailbox received!	SLV	The master has not yet received a VoE mailbox from a specific slave.
0x98110138 EC_E_DC_REF_CLOCK_SYNC_OUT_UNIT_DISABLED	SYNC out unit of reference clock is disabled!	ENI	Slave is selected as Reference clock in ENI file, but slave doesn't have a SYNC unit. Possible a ESI file bug.
0x98110139 EC_E_DC_REF_CLOCK_NOT_FOUND	Reference clock not found!	SLV	May happen if reference clock is removed from network.
0x9811013A EC_E_XML_DC_REF_CLOCK_NOT_FIRST	'Sync Window Monitoring' is active and reference clock is not first slave!	ENI	Configuration error.
0x9811016F EC_E_MAX_BUS_SLAVES_EXCEEDED	Error: Maximum number of bus slave has been exceeded!	CFG	The maximum number of pre-allocated bus slave objects are to small. The maximum number can be adjusted by the master initialization parameter EC_T_INITMASTERPARMS.wMaxBusSlaves.
0x98110170 EC_E_MBXERR_SYNTAX	Mailbox error: Syntax of 6 octet Mailbox header is wrong!	SLV	Slave error mailbox return value: 0x01
0x98110171 EC_E_MBXERR_UNSUPPORTEDPROTOCOL	Mailbox error: The Mailbox protocol is not supported!	SLV	Slave error mailbox return value: 0x02
0x98110172 EC_E_MBXERR_INVALIDCHANNEL	Mailbox error: Field contains wrong value!	SLV	Slave error mailbox return value: 0x03
0x98110173 EC_E_MBXERR_SERVICENOTSUPPORTED	Mailbox error: The mailbox protocol header of the mailbox protocol is wrong!	SLV	Slave error mailbox return value: 0x04

Code / Define	Text	Group	Possible error cause
0x98110174 EC_E_MBXERR_INVALIDHEADER	Mailbox error: The mailbox protocol header of the mailbox protocol is wrong!	SLV	Slave error mailbox return value: 0x05
0x98110175 EC_E_MBXERR_SIZETOOSHORT	Mailbox error: Length of received mailbox data is too short!	SLV	Slave error mailbox return value: 0x06
0x98110176 EC_E_MBXERR_NOMOREMEMORY	Mailbox error: Mailbox protocol can not be processed because of limited ressources!	SLV	Slave error mailbox return value: 0x07
0x98110177 EC_E_MBXERR_INVALIDSIZE	Mailbox error: The length of data is inconsistent!	SLV	Slave error mailbox return value: 0x08
0x98140000 EAP_E_CFGFILENOTFOUND	EDC file not found	EAP	The configuration file can not be located or opened for reading
0x98140001 EAP_E_WRONGSTATE	Wrong EAP state	EAP	The operation required (i.e. function call) can not be performed in actual EAP state
0x98140002 EAP_E_INITTRANSMITDATA	Can not intialize transmit data	EAP	Internal error during initialization of transmit data