



acontis technologies GmbH

SOFTWARE

EC-Master

Python Programming Interface

V3.3

AT3305

Edition: April 2, 2026

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

© Copyright **acontis technologies GmbH**

Neither this document nor excerpts therefrom may be reproduced, transmitted, or conveyed to third parties by any means whatever without the express permission of the publisher. At the time of publication, the functions described in this document and those implemented in the corresponding hardware and/or software were carefully verified; nonetheless, for technical reasons, it cannot be guaranteed that no discrepancies exist. This document will be regularly examined so that corrections can be made in subsequent editions. Note: Although a product may include undocumented features, such features are not considered to be part of the product, and their functionality is therefore not subject to any form of support or guarantee.

Contents

1	Introduction	4
1.1	Requirements	4
1.2	Architecture	4
2	Programmers Guide	6
2.1	Sample Scripts	6
2.2	Sample Code	6
2.3	Wrapper	6
2.3.1	Modules	6
2.3.2	Return code vs. exception handling	7
2.3.3	API with “out” or “ref” parameters	7
2.4	Supported IDEs	8
2.4.1	Python Shell IDLE	8
2.4.2	Visual Studio 2019	9
2.4.3	Visual Studio Code	13
3	Application programming interface, reference	17
3.1	CEcMasterPython	17
3.2	CEcWrapperPython	18
3.3	Types	124
3.4	Error Codes	214
4	Examples	258
4.1	CoeSdoUpload	258
4.2	CoeSdoDownload	258
4.3	ReadSlaveEEPROM	259
4.4	FoeFileDownload	259
5	FAQ	260
	Python Module Index	261

1 Introduction

The Python Wrapper provides a Python interface to use EC-Master, EC-Monitor, EC-Simulator and RAS Client/Server.

1.1 Requirements

Python v3.7 and above

- Python Pause. Required for ticked timing with `pause.until(...)` to lower JobTask's drift, e.g. for Distributed Clocks

```
$ pip install pause
```

- PyQt5 (v5.15.1). Only required to run the GUI demo

```
$ pip install pyqt5
```

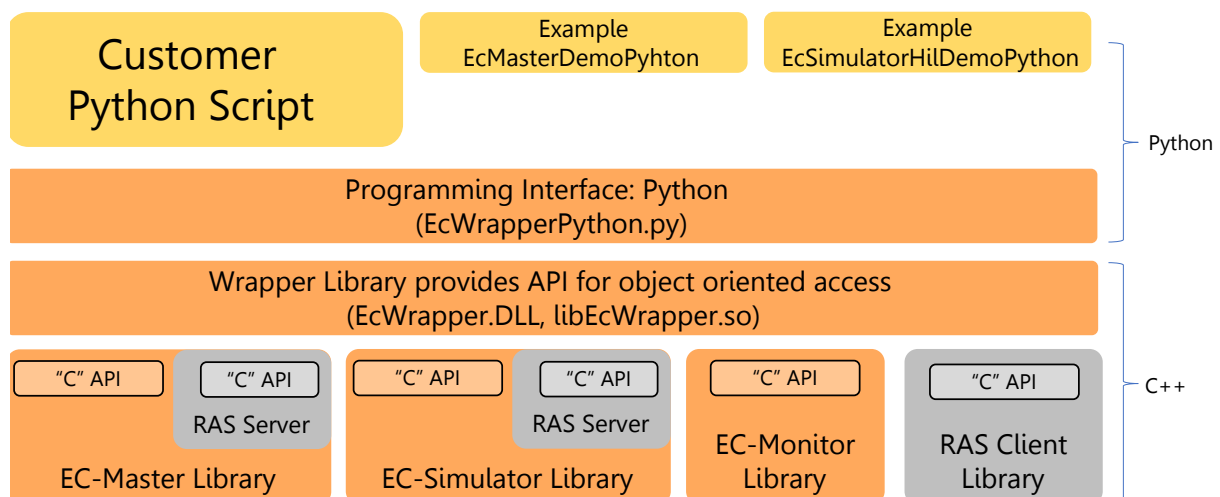
Windows (x86/x64)

- Microsoft Windows 7 and above
- Microsoft Visual C++ 2010 Runtime

Linux (x86/x64/ARM)

- Ubuntu 12.04 and above

1.2 Architecture



The architecture contains 4 basic layers:

Customer Python Script or our examples (EcMasterDemoPython, ...)

- Demo application, written in Python

Programming Interface (EcWrapperPython)

- Provides an object oriented API written in Python

Wrapper Library (EcWrapper)

- Native wrapper library, which provides API for object oriented access

Native Libraries

- EC-Master Library
- EC-Monitor Library
- EC-Simulator Library
- RAS Client Library

2 Programmers Guide

2.1 Sample Scripts

There are currently 2 scripts available:

EcMasterDemoPython.bat

Starts the console demo application

EcMasterDemoPythonInteractive.bat

Starts the interactive demo application

The scripts will start the demo application. The interactive demo application waits for user input where the user can enter the following commands:

```
# Write variable
demo.processImage.variables.Slave_1005__EL2008_.Channel_1.Output.set (1)

# Read variable
demo.processImage.variables.Slave_1005__EL2008_.Channel_1.Output.get ()

# Print properties of variable
demo.processImage.variables.Slave_1005__EL2008_.Channel_1.Output.dmp ()

# Stop the demo:
demo.stopDemo ()
```

2.2 Sample Code

The Python demo application contains of 3 modules:

EcDemoApp.py:

Console demo application

EcDemoAppGui.py:

Gui demo application, based on Qt5

EcDemoAppInteractive.py:

Interactive demo application

2.3 Wrapper

2.3.1 Modules

The Python Wrapper contains of 5 modules:

EcWrapperPython.py

```
class CEcWrapperPython
```

EC-Wrapper base class

```
class CEcMasterPython
```

EC-Master

```
class CEcMasterRasServerPython
```

RAS Server for EC-Master

```

class CEcMasterMbxGatewayClientPython
    Mailbox Gateway Client for EC-Master

class CEcMasterMbxGatewayServerPython
    Mailbox Gateway Server for EC-Master

class CEcSimulatorPython
    EC-Simulator

class CEcSimulatorRasServerPython
    RAS Server for EC-Simulator

class CEcRasClientPython
    RAS Client for EC-Master / EC-Simulator

```

EcMotionPython.py

```

class CEcMotionPython
    EC-Motion interface

```

EcWrapperPythonTypes.py

Python types

EcWrapper.py

CPython interface (internal)

EcWrapperTypes.py

CPython types (internal)

2.3.2 Return code vs. exception handling

The most of all API functions returns a return code for error handling. This behaviour can be changed to throw an exception in error case by simply setting:

```
CEcWrapperPython.EnableExceptionHandling = True # default is False
```

2.3.3 API with “out” or “ref” parameters

The Python Wrapper API is based on C# code. C# supports `out` and `ref` keywords for parameters. This is not supported in Python and is solved by simply submitting `CEcWrapperPythonOutParam` or `CEcWrapperPythonRefParam` to those functions:

```

# This function has an "out" parameter "out_oSbStatus"
def GetScanBusStatus(self, out_oSbStatus):
    # ...
    return

# Create "out" parameter
out_oStatus = CEcWrapperPythonOutParam()
# Call function
pythonWrapper.GetScanBusStatus(out_oStatus)
# Get the "out" parameter value
oSbStatus = out_oStatus.value
# Now, the "oSbStatus" object can be used
print(oSbStatus.dwResultCode)

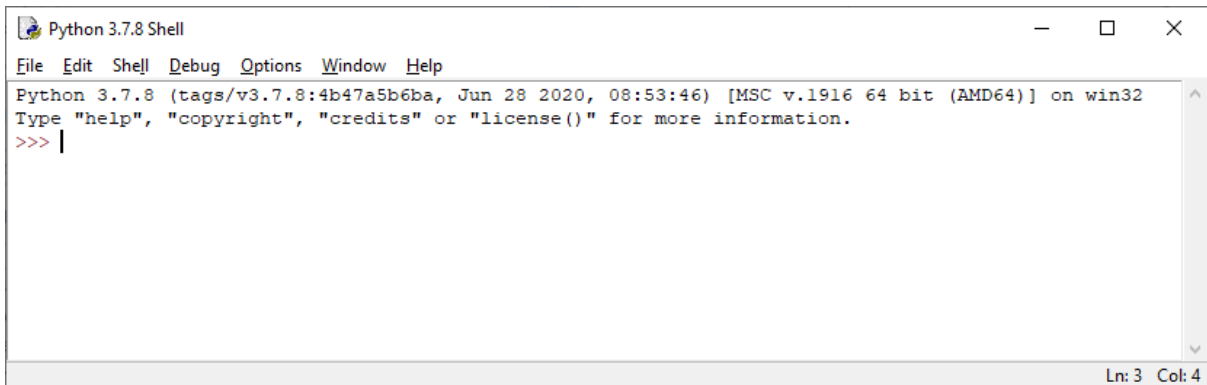
```

2.4 Supported IDEs

2.4.1 Python Shell IDLE

This is the default IDE.

It can be started from Windows Start Menu or by calling `C:/Python/Lib/idlelib/idle.py`:



```
Python 3.7.8 Shell
File Edit Shell Debug Options Window Help
Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
Ln: 3 Col: 4
```

In this shell, the user can simply copy&paste the sample code from: **Examples/EcMasterDemoPython/EcDemoAppInteractive.py**

```
exec("""
import os
import sys
INSTALLDIR = "C:/Program
Files/acontis_technologies/EC-Master-Windows-x86_64Bit/"
os.environ["PATH"] += os.pathsep + INSTALLDIR + "Bin/Windows/x64"
sys.path.append(INSTALLDIR + "Sources/EcWrapperPython")
sys.path.append(INSTALLDIR + "Examples/EcMasterDemoPython")
from EcDemoApp import \*
demo = EcMasterDemoPython()
demo.pAppParams.tRunMode = RunMode.Master
demo.pAppParams.dwBusCycleTimeUsec = 4000
demo.pAppParams.szENIFilename = "ENI.xml"
demo.pAppParams.szLinkLayer = "winpcap 127.0.0.0 1"
demo.pAppParams.nVerbose = 3
demo.startDemo()
print("EcMasterDemoPython is running.")
print("Type demo.help() for interactive help.")
""")
```

... and the demo is running.

```

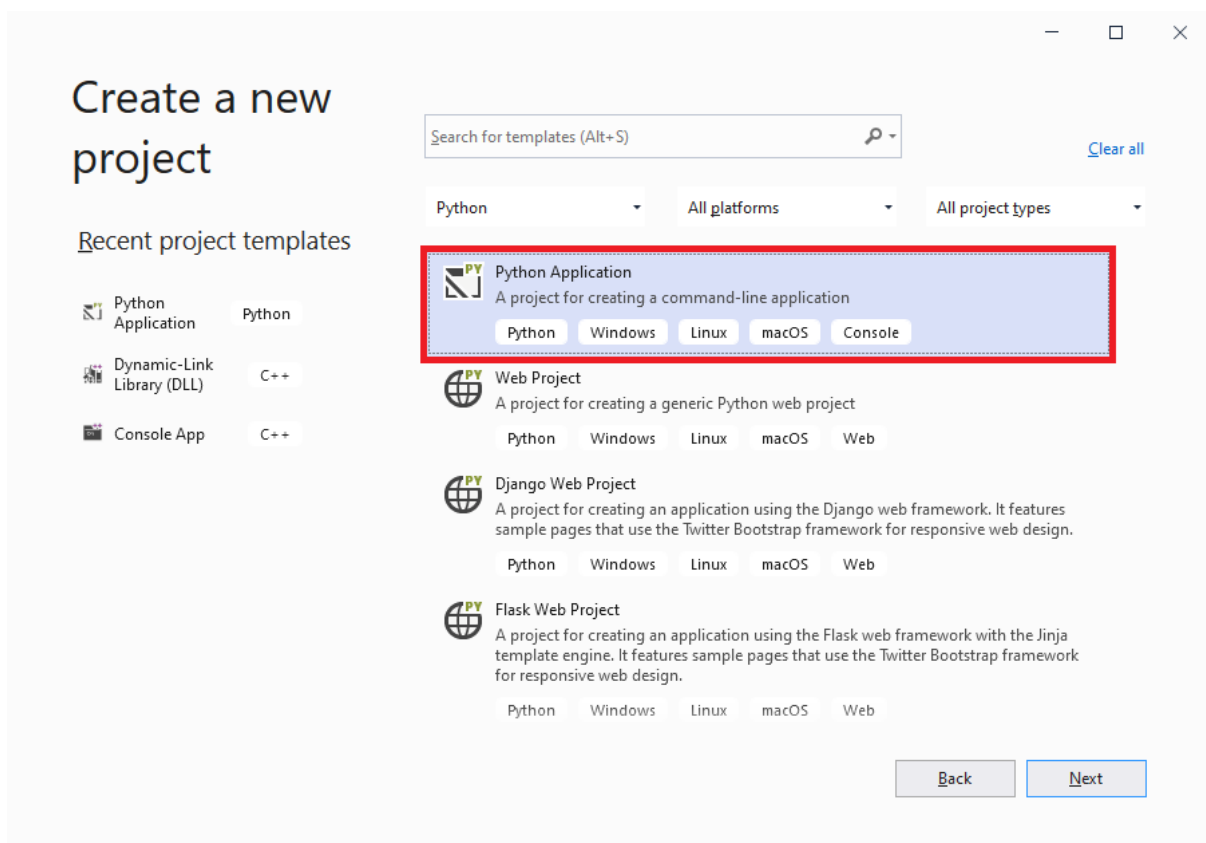
Python 3.7.8 Shell
File Edit Shell Debug Options Window Help
Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> exec("""
import os
import sys
INSTALLDIR = "C:/Temp/EC-Master-Windows-x86_64Bit/"
os.environ["PATH"] += os.pathsep + INSTALLDIR + "Bin/Windows/x64"
sys.path.append(INSTALLDIR + "Sources/ECWrapperPython")
sys.path.append(INSTALLDIR + "Examples/EcMasterDemoPython")
from EcDemoApp import *
demo = EcMasterDemoPython()
demo.pAppParams.tRunMode = RunMode.Master
demo.pAppParams.dwBusCycleTimeUsec = 4000
demo.pAppParams.szENIFilename = "d:/project.xml"
demo.pAppParams.szLinkLayer = "winpcap 172.20.143.181 1"
demo.pAppParams.nVerbose = 1
demo.startDemo()
print("EcMasterDemoPython is running.")
print("Type demo.help() for interactive help.")
""")

EtherCAT network adapter MAC: 64-70-02-04-D9-A3
EcMasterDemoPython is running.
Type demo.help() for interactive help.
>>> |
Ln: 25 Col: 4

```

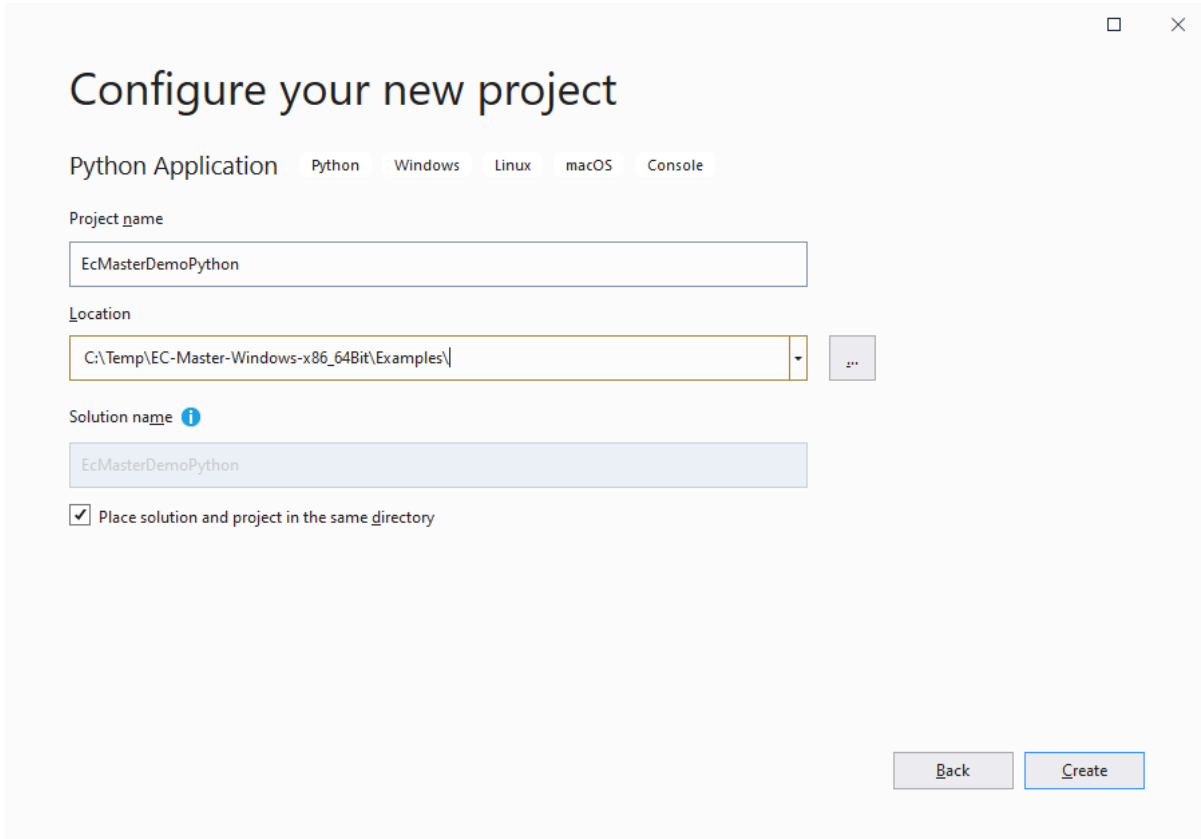
2.4.2 Visual Studio 2019

Create a new project:



Configure the project:

- Replace the generated file `EcMasterDemoPython.py` with the existing `EcDemoApp.py`.



Configure your new project


Python Application Python Windows Linux macOS Console

Project name

EcMasterDemoPython

Location

C:\Temp\EC-Master-Windows-x86_64Bit\Examples\

Solution name 

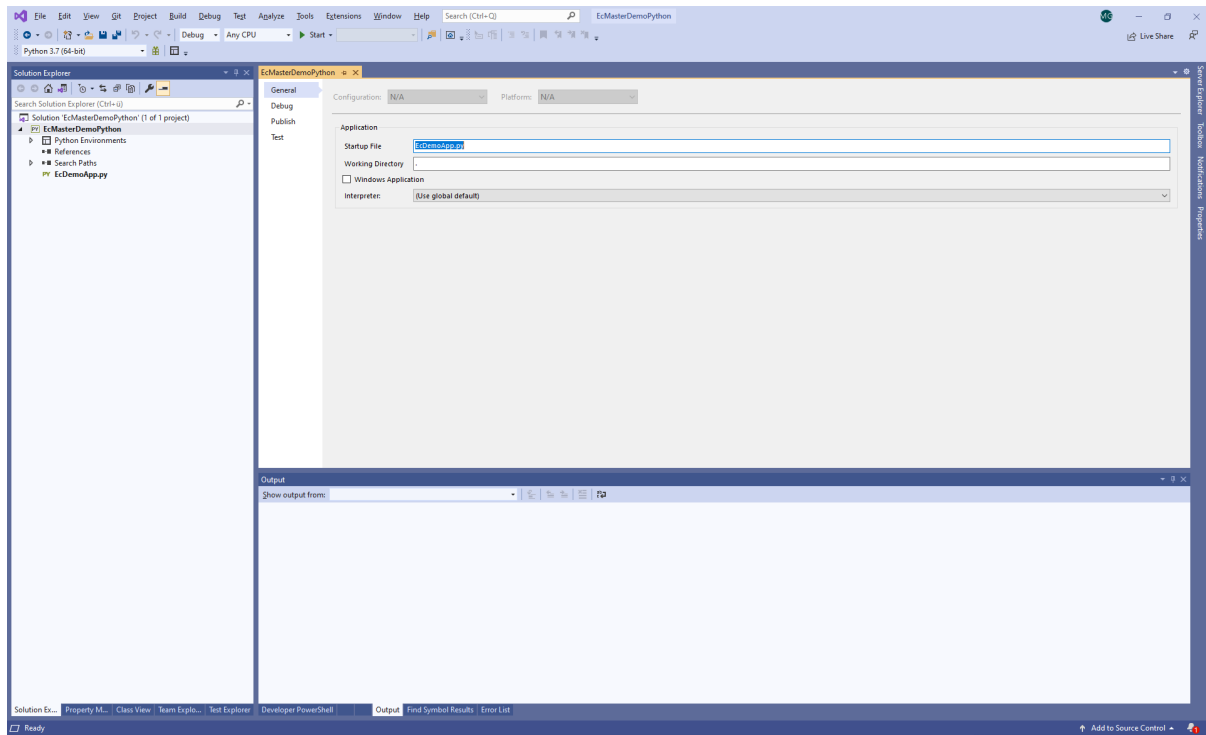
EcMasterDemoPython

Place solution and project in the same directory

Back Create

Configure project *General* settings:

- Startup File: `EcDemoApp.py`



Configure project *Debug* settings:

- Search Paths:

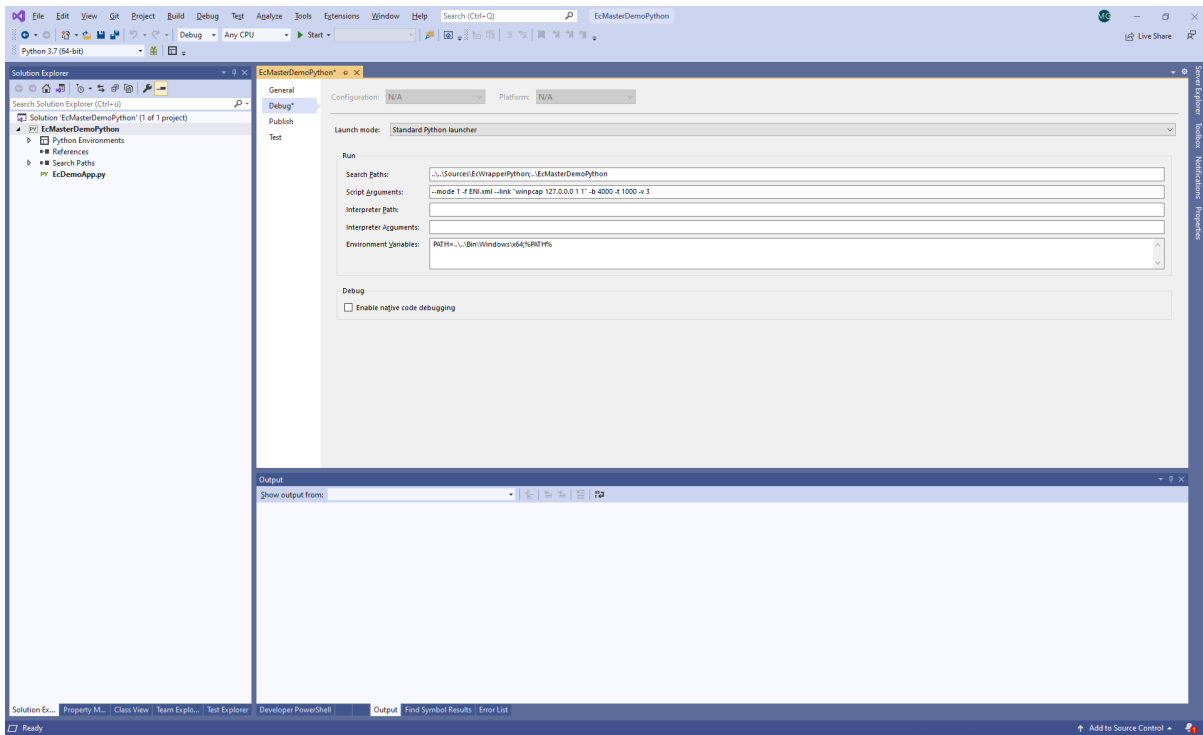
```
../../Sources/EcWrapperPython;../../EcMasterDemoPython
```

- Script Arguments:

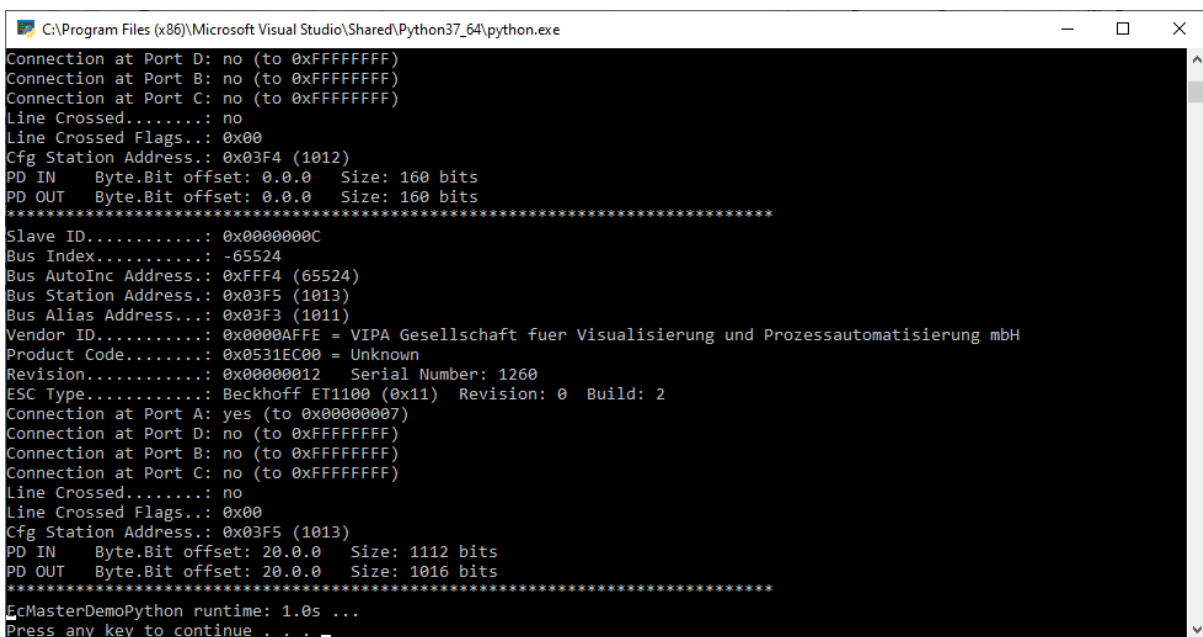
```
-mode 1 -file ENI.xml -link "winpcap 127.0.0.0 1 1" -cycletime 4000 -time 1000_
↔-lvl 3
```

- Environment Variables:

```
PATH=../../Bin/Windows/x64;%PATH%
```

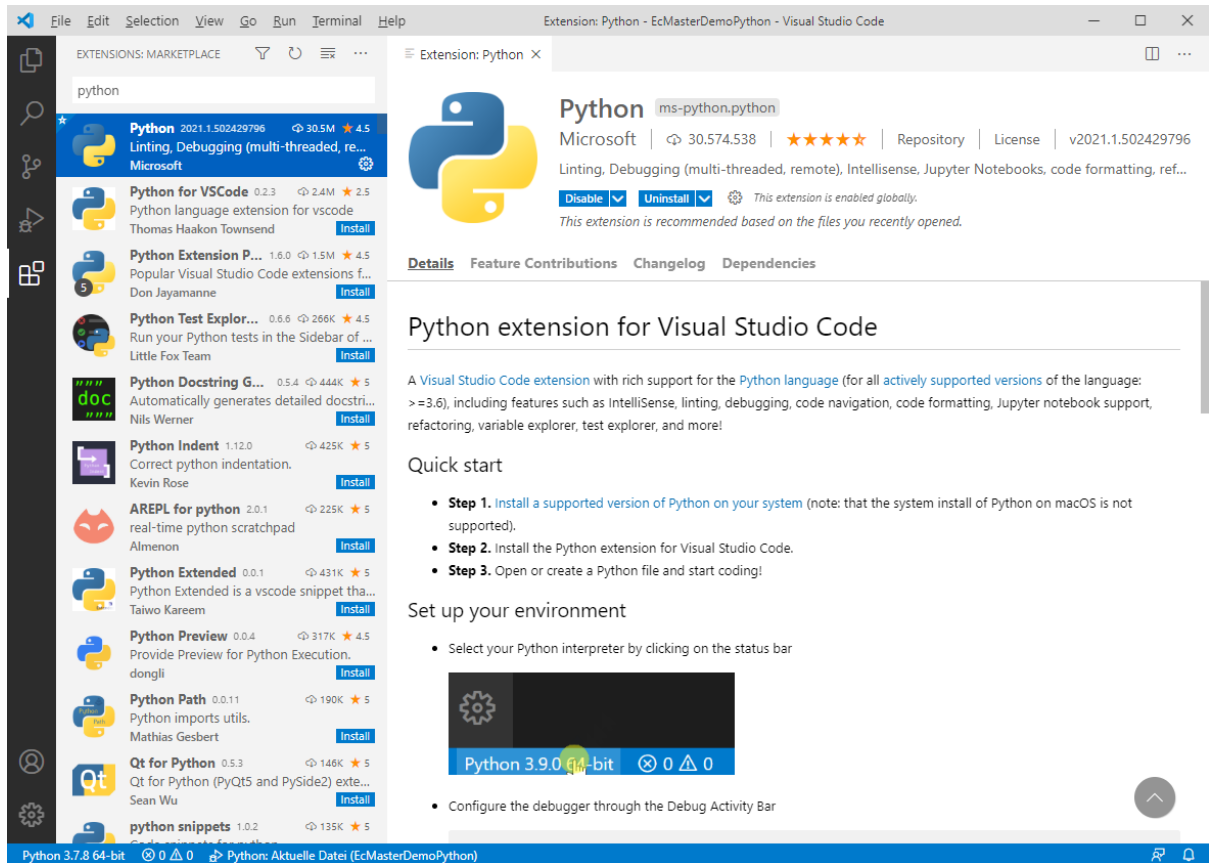


Press *Start* and the demo is running:



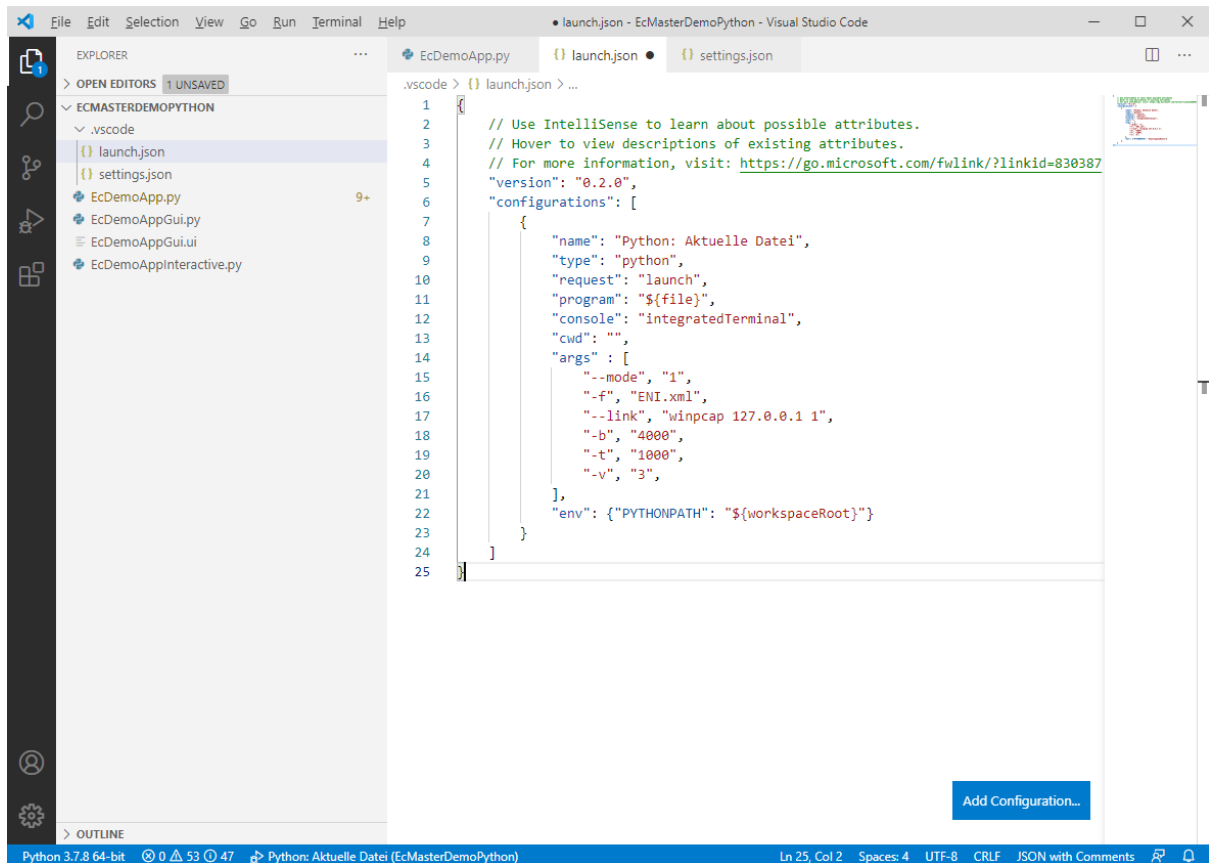
2.4.3 Visual Studio Code

Install python extension by open extension tab and enter *python*:



Open folder **Examples/EcMasterDemoPython** and configure the **launch.json**:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Python: Aktuelle Datei",
      "type": "python",
      "request": "launch",
      "program": "${file}",
      "console": "integratedTerminal",
      "cwd": "",
      "args": [
        "-mode", "1",
        "-file", "ENI.xml",
        "-link", "winpcap 127.0.0.1 1",
        "-cycleTime", "4000",
        "-time", "1000",
        "-lvl", "3",
      ],
      "env": { "PYTHONPATH": "${workspaceRoot}" }
    }
  ]
}
```

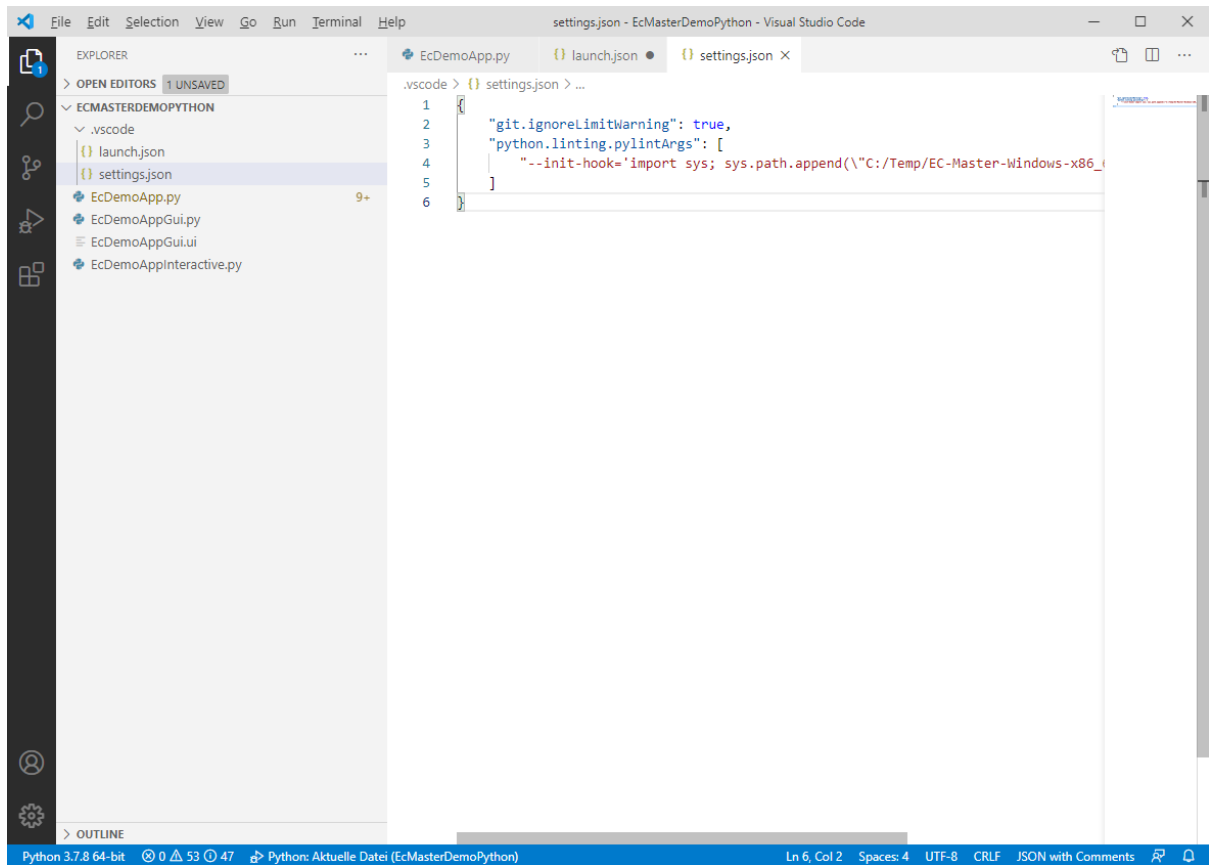


Configure linter in `settings.json`:

```

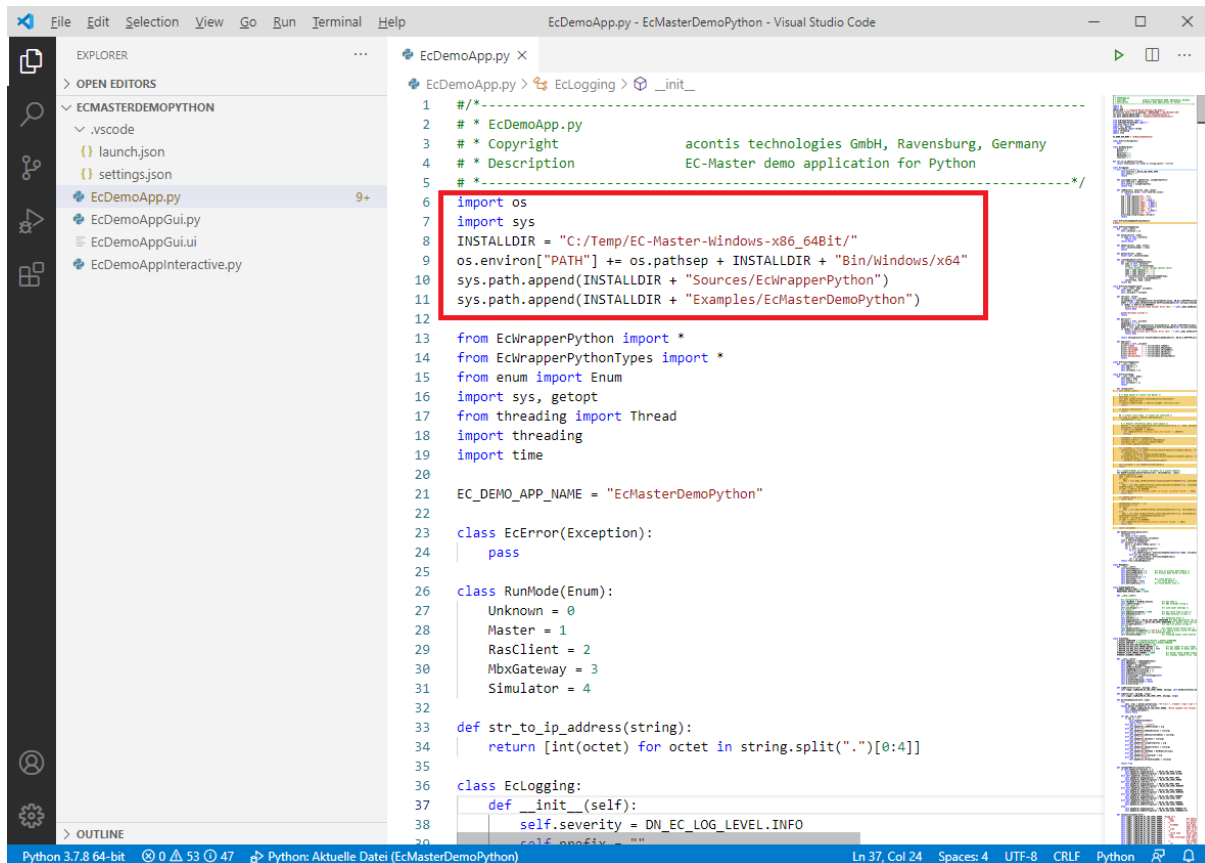
{
  "git.ignoreLimitWarning": true,
  "python.linting.pylintArgs": [
    "--init-hook='import sys; sys.path.append(\"C:/Temp/EC-Master-
↵Windows-x86_64Bit/Sources/EcWrapperPython\")' "
  ]
}

```

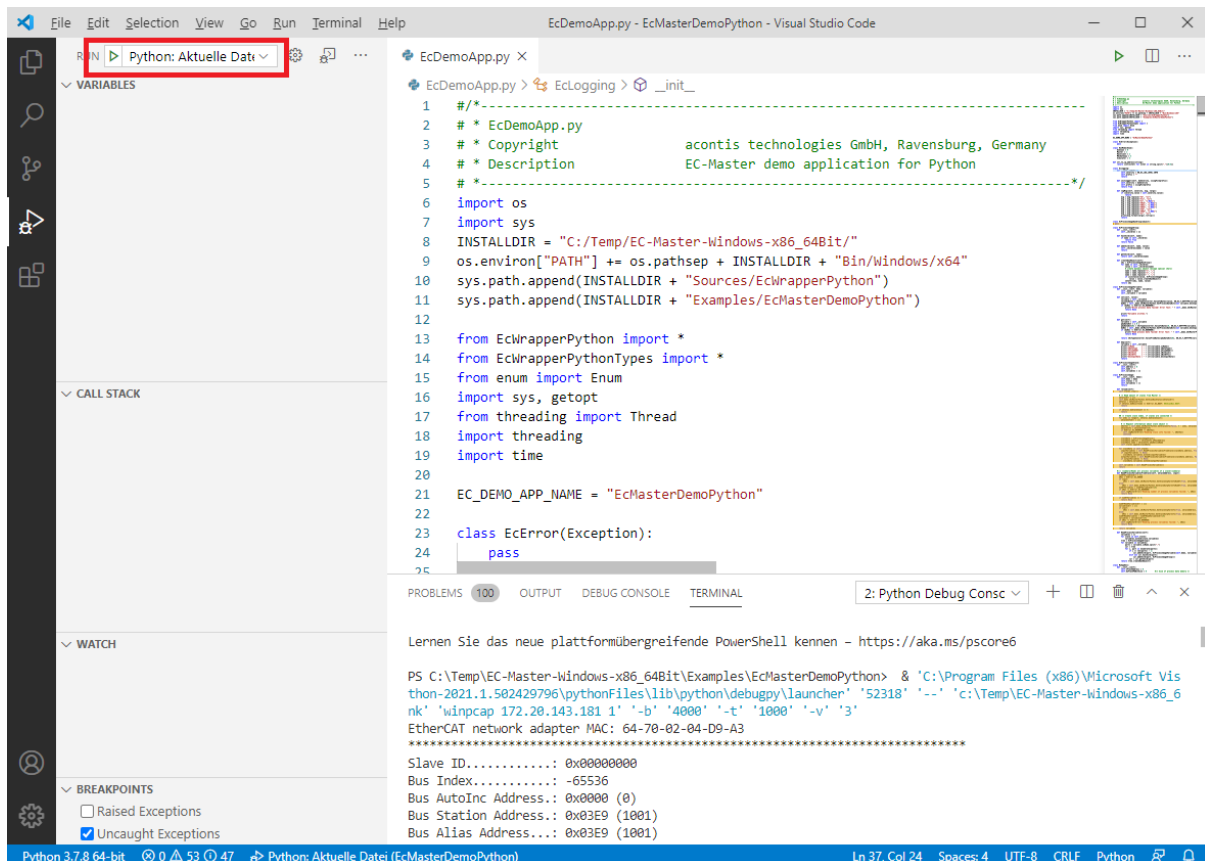


Open `EcDemoApp.py` and the following lines to set environment:

```
import os  
import sys  
INSTALLDIR = "C:/Temp/EC-Master-Windows-x86_64Bit/"  
os.environ["PATH"] += os.pathsep + INSTALLDIR + "Bin/Windows/x64"  
sys.path.append(INSTALLDIR + "Sources/EcWrapperPython")  
sys.path.append(INSTALLDIR + "Examples/EcMasterDemoPython")
```



Start debugging and the demo output will be written into the terminal:



3 Application programming interface, reference

Function prototypes, definitions etc. of the API can be found in `EcWrapperPython.py` and `EcWrapperPythonTypes.py`

3.1 CEcMasterPython

class `EcWrapperPython.CEcMasterPython`

Bases: `CEcWrapperPython`

CEcMaster for Python

Close ()

Deinitialize EcMaster

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

DoOpen (*oParms, bStartTimingTask*)

Initialize EcMaster

Parameters

- **oParms** – Parameter definitions
- **bStartTimingTask** – Start the EcWrapper provided timing task

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

static Open (*oParms, bStartTimingTask*)

Initialize EcMaster and return instance

Parameters

- **oParms** – Parameter definitions
- **bStartTimingTask** – Start the EcWrapper provided timing task

Returns

Instance on success, otherwise Null.

3.2 CEcWrapperPython

class EcWrapperPython.CEcWrapperPython

ActiveSlaveEEPROM (

bFixedAddressing: bool,
wSlaveAddress: int,
out_pbSlavePDIAccessActive: bool,
dwTimeout: int

) → *ECError*

Check whether EEPROM is marked access active by Slave PDI application.

This function may not be called from within the JobTask's context.

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **pbSlavePDIAccessActive** – Pointer to Boolean value: EC_TRUE: EEPROM active by PDI application, EC_FALSE: EEPROM not active
- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time. The timeout value must not be set to EC_NOWAIT.
- **out_pbSlavePDIAccessActive** (*bool*) –

Returns

EC_E_NOERROR or error code

Return type

ECError

ActiveSlaveEEPROMReq (

dwClientId: int,
dwTferId: int,
bFixedAddressing: bool,
wSlaveAddress: int,
out_pbSlavePDIAccessActive: bool,
dwTimeout: int

) → *ECError*

Requests EEPROM is marked access active by Slave PDI application check and returns immediately.

An EC_NOTIFY_EEPROM_OPERATION is given on completion or timeout. This function may be called from within the JobTask's context.

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **dwTferId** (*int*) – Transfer ID. The application can set this ID to identify the transfer. It will be passed back to the application within EC_T_EEPROM_OPERATION_NOTIFY_DESC.
- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **pbSlavePDIAccessActive** – Pointer to Boolean value: EC_TRUE: EEPROM active by PDI application, EC_FALSE: EEPROM not active. Must be valid until operation complete.

- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time. The timeout value must not be set to EC_NOWAIT.
- **out_pbSlavePDIAccessActive** (*bool*) –

Returns

EC_E_NOERROR or error code

Return type

EError

AddDbgMsg (*strDbgMsg: str*)

Adds a debug message to the print message queue

Parameters**strDbgMsg** (*str*) – debug message**AdsAdapterStart** (*poStartParams: DN_EC_T_ADS_ADAPTER_START_PARMS,**dwTimeout: int*

) → EError

Starts the ADS Pass Through Server

- #EC_E_NOERROR if successful
- #EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- #EC_E_INVALIDPARAM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer
- #EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- #EC_E_ADS_IS_RUNNING if the ADS server is running

Parameters

- **poStartParams** (DN_EC_T_ADS_ADAPTER_START_PARMS) – Pass through server start parameter
- **dwTimeout** (*int*) – Timeout

Return type

EError

Returns:

AdsAdapterStop (*dwTimeout: int*) → EError

Stops the ADS Pass Through Server

Parameters**dwTimeout** (*int*) – Timeout**Returns**

EC_E_NOERROR or error code

Return type

EError

AoeGetSlaveNetId (*dwSlaveId: int, out_poAoeNetId: DN_EC_T_AOE_NETID*) → EError

Retrieve the NetID of a specific EtherCAT device

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **poAoeNetId** – AoE NetID of the corresponding slave
- **out_poAoeNetId** (DN_EC_T_AOE_NETID) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer, or dwTimeout is EC_NOWAIT
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if slave has no mailbox support
- EC_E_ADS_IS_RUNNING if ADS server is running

Return type

[ESError](#)

AoeRead (

dwSlaveId: int,
ref_poTargetNetId: DN_EC_T_AOE_NETID,
wTargetPort: int,
dwIndexGroup: int,
dwIndexOffset: int,
dwDataLen: int,
pbyData: List[int],
out_pdwDataOutLen: int,
out_pdwErrorCode: int,
out_pdwCmdResult: int,
dwTimeout: int

) → [ESError](#)

Execute an AoE mailbox read request to an EtherCAT slave device.

This function may not be called from within the JobTask's context.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **poTargetNetId** – Target NetID
- **wTargetPort** (*int*) – Target port
- **dwIndexGroup** (*int*) – AoE read command index group
- **dwIndexOffset** (*int*) – AoE read command index offset
- **dwDataLen** (*int*) – Buffer length [bytes]
- **pbyData** (*List[int]*) – Buffer receiving transferred data
- **pdwDataOutLen** – Number of bytes read from the target device
- **pdwErrorCode** – AoE response error code
- **pdwCmdResult** – AoE read command result code
- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time.
- **ref_poTargetNetId** (DN_EC_T_AOE_NETID) –
- **out_pdwDataOutLen** (*int*) –
- **out_pdwErrorCode** (*int*) –
- **out_pdwCmdResult** (*int*) –

Returns

- EC_E_NOERROR on success
- EC_E_AOE_VENDOR_SPECIFIC if the AoE device has responded with a user defined error code
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive

Return type

[EError](#)

AoeReadReq (

pMbxTfer: object,
dwSlaveId: int,
poTargetNetId: DN_EC_T_AOE_NETID,
wTargetPort: int,
dwIndexGroup: int,
dwIndexOffset: int,
dwTimeout: int

) → [EError](#)

Execute a non-blocking AoE mailbox read request to an EtherCAT slave device.

If the function returns EC_E_AOE_DEVICE_XXXX (well known device errors) or EC_E_AOE_VENDOR_SPECIFIC the original ADS error can be retrieved from the mailbox transfer object at EC_T_MBXTFER.MbxData.AoE_Response. A unique transfer ID must be written into EC_T_MBXTFER.dwTferId.

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object
- **dwSlaveId** (*int*) – Slave ID
- **poTargetNetId** (DN_EC_T_AOE_NETID) – Target NetID
- **wTargetPort** (*int*) – Target port
- **dwIndexGroup** (*int*) – AoE read command index group
- **dwIndexOffset** (*int*) – AoE read command index offset
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR on success
- EC_E_AOE_VENDOR_SPECIFIC if the AoE device has responded with a user defined error code
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive

Return type

[EError](#)

```

AoeReadWrite (
    dwSlaveId: int,
    ref_poTargetNetId: DN_EC_T_AOE_NETID,
    wTargetPort: int,
    dwIndexGroup: int,
    dwIndexOffset: int,
    dwReadDataLen: int,
    dwWriteDataLen: int,
    pbyData: List[int],
    out_pdwDataOutLen: int,
    out_pdwErrorCode: int,
    out_pdwCmdResult: int,
    dwTimeout: int
) → ECError

```

Execute an AoE mailbox read/write request to an EtherCAT slave device.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **poTargetNetId** – Target NetID
- **wTargetPort** (*int*) – Target port
- **dwIndexGroup** (*int*) – AoE read/write command index group
- **dwIndexOffset** (*int*) – AoE read/write command index offset
- **dwReadDataLen** (*int*) – Number of bytes to read from the target device
- **dwWriteDataLen** (*int*) – Number of bytes to write to the target device
- **pbyData** (*List[int]*) – [in, out] Buffer containing and receiving transferred data
- **pdwDataOutLen** – Number of bytes read from the target device
- **pdwErrorCode** – Pointer to AoE response error code
- **pdwCmdResult** – Pointer to AoE write command result code
- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time. EC_NOWAIT is not valid.
- **ref_poTargetNetId** (DN_EC_T_AOE_NETID) –
- **out_pdwDataOutLen** (*int*) –
- **out_pdwErrorCode** (*int*) –
- **out_pdwCmdResult** (*int*) –

Returns

- EC_E_NOERROR on success
- EC_E_AOE_VENDOR_SPECIFIC if the AoE device has responded with a user defined error code
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive

Return type

ECError

```
AoeWrite (  
    dwSlaveId: int,  
    ref_poTargetNetId: DN_EC_T_AOE_NETID,  
    wTargetPort: int,  
    dwIndexGroup: int,  
    dwIndexOffset: int,  
    dwDataLen: int,  
    pbyData: List[int],  
    out_pdwErrorCode: int,  
    out_pdwCmdResult: int,  
    dwTimeout: int  
) → ECErrror  
Execute an AoE mailbox write request to an EtherCAT slave device.
```

This function may not be called from within the JobTask's context.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **poTargetNetId** – Target NetID
- **wTargetPort** (*int*) – Target port
- **dwIndexGroup** (*int*) – AoE write command index group
- **dwIndexOffset** (*int*) – AoE write command index offset
- **dwDataLen** (*int*) – Buffer length [bytes]
- **pbyData** (*List[int]*) – Buffer containing transfered data
- **pdwErrorCode** – Pointer to AoE response error code
- **pdwCmdResult** – Pointer to AoE write command result code
- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time.
- **ref_poTargetNetId** (DN_EC_T_AOE_NETID) –
- **out_pdwErrorCode** (*int*) –
- **out_pdwCmdResult** (*int*) –

Returns

- EC_E_NOERROR on success
- EC_E_AOE_VENDOR_SPECIFIC if the AoE device has responded with a user defined error code
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive

Return type

[ECErrror](#)

```
AoeWriteControl (
    dwSlaveId: int,
    poTargetNetId: DN_EC_T_AOE_NETID,
    wTargetPort: int,
    wAoEState: int,
    wDeviceState: int,
    dwDataLen: int,
    pbyData: List[int],
    ref_pdwErrorCode: int,
    ref_pdwCmdResult: int,
    dwTimeout: int
) → ECErr
Execute an AoE mailbox write control request to an EtherCAT slave device.
```

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **poTargetNetId** (DN_EC_T_AOE_NETID) – Target NetID. The Target NetID of a AoE slave device can be retrieved by `ecatAoeGetSlaveNetId()`.
- **wTargetPort** (*int*) – Target port
- **wAoEState** (*int*) – AoE state
- **wDeviceState** (*int*) – Device specific state
- **dwDataLen** (*int*) – Buffer length [bytes]
- **pbyData** (*List[int]*) – Buffer containing transfered data
- **pdwErrorCode** – Pointer to AoE response error code
- **pdwCmdResult** – Pointer to AoE write command result code
- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time. EC_NOWAIT is not valid.
- **ref_pdwErrorCode** (*int*) –
- **ref_pdwCmdResult** (*int*) –

Returns

- EC_E_NOERROR on success
- EC_E_AOE_VENDOR_SPECIFIC if the AoE device has responded with a user defined error code
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive

Return type

ECErr

```
AoeWriteReq (
    pMbxTfer: object,
    dwSlaveId: int,
    poTargetNetId: DN_EC_T_AOE_NETID,
    wTargetPort: int,
    dwIndexGroup: int,
    dwIndexOffset: int,
    dwTimeout: int
) → ECErr
Execute a non-blocking AoE mailbox write request to an EtherCAT slave device.
```

If the function returns `EC_E_AOE_DEVICE_XXXX` (well known device errors) or `EC_E_AOE_VENDOR_SPECIFIC` the original ADS error can be retrieved from the mailbox transfer object at `EC_T_MBXTFER.MbxData.AoE_Response`. A unique transfer ID must be written into `EC_T_MBXTFER.dwTferId`.

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object
- **dwSlaveId** (*int*) – Slave ID
- **poTargetNetId** (`DN_EC_T_AOE_NETID`) – Target NetID
- **wTargetPort** (*int*) – Target port
- **dwIndexGroup** (*int*) – AoE write command index group
- **dwIndexOffset** (*int*) – AoE write command index offset
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- `EC_E_NOERROR` on success
- `EC_E_AOE_VENDOR_SPECIFIC` if the AoE device has responded with a user defined error code
- `EC_E_MASTER_RED_STATE_INACTIVE` if Master Redundancy is configured and master is inactive

Return type

[ECError](#)

AssignSlaveEEPROM (

bFixedAddressing: bool,
wSlaveAddress: int,
bSlavePDIAccessEnable: bool,
bForceAssign: bool,
dwTimeout: int

) → [ECError](#)

Set EEPROM Assignment to PDI or EtherCAT Master.

This function may not be called from within the JobTask's context.

Parameters

- **bFixedAddressing** (*bool*) – `EC_TRUE`: use station address, `EC_FALSE`: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **bSlavePDIAccessEnable** (*bool*) – `EC_TRUE`: EEPROM assigned to slave PDI application,
- **bForceAssign** (*bool*) – Force Assignment of EEPROM (only for Ecat Master Assignment)
- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time. The timeout value must not be set to `EC_NOWAIT`.

Returns

`EC_E_NOERROR` or error code

Return type

[ECError](#)

AssignSlaveEEPROMReq (

dwClientId: int,
dwTferId: int,
bFixedAddressing: bool,
wSlaveAddress: int,
bSlavePDIAccessEnable: bool,
bForceAssign: bool,
dwTimeout: int

) → *ECErr*

Requests EEPROM Assignment to PDI or EtherCAT Master operation and returns immediately

EC_NOTIFY_EEPROM_OPERATION is given on completion or timeout. This function may be called from within the JobTask's context.

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **dwTferId** (*int*) – Transfer ID. The application can set this ID to identify the transfer. It will be passed back to the application within EC_T_EEPROM_OPERATION_NOTIFY_DESC.
- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **bSlavePDIAccessEnable** (*bool*) – EC_TRUE: EEPROM assigned to slave PDI application, EC_FALSE: EEPROM assigned to EC-Master
- **bForceAssign** (*bool*) – Force Assignment of EEPROM (only for ECat Master Assignment)
- **dwTimeout** (*int*) – Timeout [ms]

Returns

EC_E_NOERROR or error code

Return type

ECErr

BadConnectionsDetect (*bRefreshSlaveStatistics*: bool, *dwTimeout*: int) → *ECErr*

Detects bad connections

Analyzes the slave ESC error counters: - Invalid Frame Counter (0x0300), - RX Error Counter (0x0301), - Lost Link Counter (0x0310),

whether there is a problem in the area PHY - connector - cable - connector - PHY. If one of the above error counters shows a value not equal to zero, an EC_NOTIFY_BAD_CONNECTION is generated, which contains the exact position of the faulty connection.

Parameters

- **bRefreshSlaveStatistics** (*bool*) –
- **dwTimeout** (*int*) – Timeout [ms]. not be EC_NOWAIT!

Returns

EC_E_NOERROR or error code

Return type

ECErr

BlockNode (*pMismatch*: DN_EC_T_SB_MISMATCH_DESC, *dwTimeout*: int) → *ECErr*

Blocks a slave on a specific port

If an invalid slave node is connected, which causes the bus topology scan to fail, the previous port, where this node is connected to can be shut down with this call. This allows a Hot-Connect system to not be disturbed, if unknown nodes are connected. If this function will be executed on a Hot-Connect member (a slave which is part of a hot connect group) the complete hot connect group will be excluded from the bus. This function may only be called from within the JobTask's context with parameter `dwTimeout` set to `EC_NOWAIT`.

Parameters

- **pMismatch** (`DN_EC_T_SB_MISMATCH_DESC`) – Pointer to `EC_T_SB_MISMATCH_DESC` carrying mismatch descriptor.
- **dwTimeout** (`int`) – Timeout [ms]. The function will block at most for this time.

Returns

`EC_E_NOERROR` or error code

Return type

`ECError`

ClearSlaveStatistics (`dwSlaveId: int`) → `ECError`

Clears all error registers of a slave.

Parameters

dwSlaveId (`int`) – Slave Id, `INVALID_SLAVE_ID` clears all slaves

Returns

`EC_E_NOERROR` or error code

Return type

`ECError`

ClntQueueRawCmd (

dwClntId: int,
wInvokeId: int,
byCmd: int,
dwMemoryAddress: int,
pbData: List[int],
wLen: int

) → `ECError`

Transfers a raw EtherCAT command to one or multiple slaves.

Using this function it is possible to exchange data between the master and the slaves. When the response to the queued frame is received, the notification `EC_NOTIFY_RAWCMD_DONE` is given for the appropriate client. This function queues a single EtherCAT command. Queued raw commands will be sent after sending cyclic process data values. If a timeout occurs the corresponding frame will be sent again, the timeout value and retry counter can be set using the master configuration parameters `EC_T_INIT_MASTER_PARAMS.dwEcatCmdTimeout` and `EC_T_INIT_MASTER_PARAMS.dwEcatCmdMaxRetries`. Using auto increment addressing (APRD, APWR, APRW) may lead to unexpected results in case the selected slave does not increment the working counter. In such cases the EtherCAT command would be handled by the slave directly behind the selected one. This function may not be called from within the JobTask's context.

Parameters

- **dwClntId** (`int`) – Client ID to be notified (0 if all registered clients shall be notified)
- **wInvokeId** (`int`) – Invoke ID to reassign the results to the sent CMD
- **byCmd** (`int`) – EtherCAT command
- **dwMemoryAddress** (`int`) – Slave memory address, depending on the command to be sent this is either a physical or a logical address

- **pbyData** (*List[int]*) – [in, out] Buffer containing or receiving transferred data. In case a read-only command is queued (e.g. APRD) this pointer should be set to a value of EC_NULL.
- **wLen** (*int*) – Number of bytes to transfer

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceId is out of range or the command is not supported
- EC_E_BUSY if the master or the corresponding slave is currently changing its operational state
- EC_E_INVALIDSIZE if the size of the complete command does not fit into a single Ethernet frame. The maximum amount of data to transfer must not exceed 1486 bytes.

Return type

[ECErrors](#)

```
ClntSendRawMbx (
    dwClntId: int,
    pbyMbxCmd: List[int],
    dwMbxCmdLen: int,
    dwTimeout: int
) → ECErrors
Send a raw mailbox command
```

Parameters

- **dwClntId** (*int*) – Client ID
- **pbyMbxCmd** (*List[int]*) – Buffer containing the raw mailbox command starting with mailbox header
- **dwMbxCmdLen** (*int*) – Length of pbyMbxCmd buffer
- **dwTimeout** (*int*) – Timeout [ms]

Returns

EC_E_NOERROR or error code

Return type

[ECErrors](#)

```
CoeGetEntryDesc (
    dwSlaveId: int,
    wIndex: int,
    bySubIndex: int,
    byValueInfoType: int,
    out_oEntryDesc: DN_EC_T_COE_ENTRYDESC,
    dwTimeout: int
) → ECErrors
Determines the description of a specific object entry
```

Parameters

- **dwSlaveId** (*int*) – slave ID
- **wIndex** (*int*) – Object index
- **bySubIndex** (*int*) – Object subindex

- **byValueInfoType** (*int*) – Bit mask to define which information to determine. The bit values are defined as follows: EC_COE_ENTRY_ObjAccess - Object access rights EC_COE_ENTRY_ObjCategory - Object category EC_COE_ENTRY_PdoMapping - Information if the object is PDO mappable EC_COE_ENTRY_UnitType - Unit EC_COE_ENTRY_DefaultValue - Default value EC_COE_ENTRY_MinValue - Minimum value EC_COE_ENTRY_MaxValue - Maximum value
- **out_oEntryDesc** (DN_EC_T_COE_ENTRYDESC) – out Description of a specific object entry
- **dwTimeout** (*int*) – Timeout

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

CoeGetEntryDescReq (

pMbxTfer: object,
dwSlaveId: int,
wObIndex: int,
byObSubIndex: int,
byValueInfo: int,
dwTimeout: int

) → *ECError*

Initiates retrieval of the description of a specific object entry and returns immediately.

A unique transfer ID must be written into EC_T_MBXTFER.dwTferId. EC_NOTIFY_MBOXRCV is given on completion.

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object
- **dwSlaveId** (*int*) – Slave ID
- **wObIndex** (*int*) – Object index
- **byObSubIndex** (*int*) – Object sub-index
- **byValueInfo** (*int*) – The value info bit mask includes which elements shall be in the response. See EC_COE_ENTRY_VALUEINFO “Value info flags” for available values.
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer, or dwTimeout is EC_NOWAIT
- EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if slave has no mailbox support
- EC_E_INVALID_SLAVE_STATE if slave is in an invalid state for mailbox transfer
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive

- EC_E_ADS_IS_RUNNING if ADS server is running
- EC_ERROR_CODES_COE “CoE SDO error code”

Return type

ECErr

```

CoeGetODList (
    dwSlaveId: int,
    eListType: DN_EC_T_COE_ODLIST_TYPE,
    out_pList: List[int],
    dwLen: int,
    out_pdwOutLen: int,
    dwTimeout: int
) → ECErr
    Gets a list of object Ids that are available in a slave.

```

Parameters

- **dwSlaveId** (*int*) – slave ID
- **eListType** (*DN_EC_T_COE_ODLIST_TYPE*) – Object list type
eODListType_ALL: Get all objects
eODListType_RxPdoMap: Get PDO mappable objects
eODListType_TxPdoMap: Get objects that can be changed
eODListType_StoredFRapl: Get objects that are stored for a device replacement
eODListType_StartupParm: Get objects that can be used as startup parameter
- **out_pList** (*List [int]*) – out Filled OD list
- **dwLen** (*int*) – Length of list
- **out_pdwOutLen** (*int*) – out Length of output list
- **dwTimeout** (*int*) – Timeout

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

```

CoeGetODListReq (
    pMbxTfer: object,
    dwSlaveId: int,
    eListType: DN_EC_T_COE_ODLIST_TYPE,
    dwTimeout: int
) → ECErr
    Initiates retrieval of a list of object IDs that have so far been transferred to a slave and received by the EC-Monitor and returns immediately.

```

!(EC_MONITOR)

Initiates retrieval of a list of object IDs that are available in a slave and returns immediately.

A unique transfer ID must be written into *EC_T_MBXTFER.dwTferId*. *EC_NOTIFY_MBOXRCV* is given on completion.

The mailbox transfer object will receive the slave response containing the list type followed by the list itself. Therefore the buffer must be 2 bytes bigger than the expected list size.

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer
- **dwSlaveId** (*int*) – Slave ID
- **eListType** (*DN_EC_T_COE_ODLIST_TYPE*) – Which object types shall be transferred

- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer, or dwTimeout is EC_NOWAIT
- EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if slave has no mailbox support
- EC_E_INVALID_SLAVE_STATE if slave is in an invalid state for mailbox transfer
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if ADS server is running
- EC_ERROR_CODES_COE “CoE SDO error code”

Return type

ECErr

```
CoeGetObjectDesc (
    dwSlaveId: int,
    wIndex: int,
    out_oObjDesc: DN_EC_T_COE_OBDESC,
    dwTimeout: int
) → ECErr
Determines the description of a specific object
```

Parameters

- **dwSlaveId** (*int*) – slave ID
- **wIndex** (*int*) – Object index
- **out_oObjDesc** (DN_EC_T_COE_OBDESC) – out Description of a specific object
- **dwTimeout** (*int*) – Timeout

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

```
CoeGetObjectDescReq (
    pMbxTfer: object,
    dwSlaveId: int,
    wObIndex: int,
    dwTimeout: int
) → ECErr
Initiates retrieval of the description of a specific object and returns immediately.
```

A unique transfer ID must be written into EC_T_MBXTFER.dwTferId. EC_NOTIFY_MBOXRCV is given on completion.

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object
- **dwSlaveId** (*int*) – Slave ID
- **wObIndex** (*int*) – Object index
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer, or dwTimeout is EC_NOWAIT
- EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if slave has no mailbox support
- EC_E_INVALID_SLAVE_STATE if slave is in an invalid state for mailbox transfer
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if ADS server is running
- EC_ERROR_CODES_COE “CoE SDO error code”

Return type

[ECErrors](#)

CoeProfileGetChannelInfo (

bStationAddress: bool,
wSlaveAddress: int,
dwChannel: int,
pInfo: DN_EC_T_PROFILE_CHANNEL_INFO

) → [ECErrors](#)

Return information about a configured CoE profile channel from the ENI file

Parameters

- **bStationAddress** (*bool*) –
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **dwChannel** (*int*) – Channel
- **pInfo** (*DN_EC_T_PROFILE_CHANNEL_INFO*) – Channel info

Returns

EC_E_NOERROR or error code

Return type

[ECErrors](#)

CoeRxPdoTfer (*pMbxTfer: object, dwSlaveId: int, dwNumber: int, dwTimeout: int*) → [ECErrors](#)

Initiate CoE RxPDO transfer

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object
- **dwSlaveId** (*int*) – Slave ID

- **dwNumber** (*int*) – PDO number
- **dwTimeout** (*int*) – Timeout [ms]

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

CoeSdoDownload (

dwSlaveId: int,
wObIndex: int,
byObSubIndex: int,
pbData: List[int],
dwDataLen: int,
dwTimeout: int,
dwFlags: int

) → [ESError](#)

Initiates a CoE SDO download to an EtherCAT slave device and returns immediately.

The length of the data to be downloaded must be set in `EC_T_MBXTFER.dwDataLen`. A unique transfer ID must be written into `EC_T_MBXTFER.dwTferId`. `EC_NOTIFY_MBOXRCV` is given on completion.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **wObIndex** (*int*) – Object index
- **byObSubIndex** (*int*) – Object sub index. If Complete Access only 0 or 1 allowed.
- **pbData** (*List[int]*) –
- **dwDataLen** (*int*) –
- **dwTimeout** (*int*) – Timeout [ms]
- **dwFlags** (*int*) – Mailbox Flags. Bit 0: set if Complete Access (`EC_MAILBOX_FLAG_SDO_COMPLETE`).

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if `dwInstanceID` is out of range, the input pointer is `EC_NULL` or contains `EC_NULL` pointer, or `dwTimeout` is `EC_NOWAIT`
- EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- EC_E_SLAVE_NOT_PRESENT if the slave is not present
- EC_E_NOTFOUND if no slave matching `dwSlaveId` can be found
- EC_E_NO_MBX_SUPPORT if the slave has no mailbox support
- EC_E_INVALID_SLAVE_STATE if the slave is in an invalid state for mailbox transfer
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if the ADS server is running
- EC_ERROR_CODES_COE “CoE SDO error code”

Return type

EError

CoESdoDownloadReq (

pMbxTfer: object,
dwSlaveId: int,
wObIndex: int,
byObSubIndex: int,
dwTimeout: int,
dwFlags: int

) → EError

Initiates a CoE SDO download to an EtherCAT slave device and returns immediately.

The length of the data to be downloaded must be set in EC_T_MBXTFER.dwDataLen. A unique transfer ID must be written into EC_T_MBXTFER.dwTferId. EC_NOTIFY_MBOXRCV is given on completion.

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object
- **dwSlaveId** (*int*) – Slave ID
- **wObIndex** (*int*) – Object index
- **byObSubIndex** (*int*) – Object sub index. If Complete Access only 0 or 1 allowed.
- **dwTimeout** (*int*) – Timeout [ms]
- **dwFlags** (*int*) – Mailbox Flags. Bit 0: set if Complete Access (EC_MAILBOX_FLAG_SDO_COMPLETE).

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARG if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer, or dwTimeout is EC_NOWAIT
- EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- EC_E_SLAVE_NOT_PRESENT if the slave is not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if the slave has no mailbox support
- EC_E_INVALID_SLAVE_STATE if the slave is in an invalid state for mailbox transfer
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if the ADS server is running
- EC_ERROR_CODES_COE “CoE SDO error code”

Return type

EError

```

CoeSdoUpload (
    dwSlaveId: int,
    wObIndex: int,
    byObSubIndex: int,
    pbyData: List[int],
    dwDataLen: int,
    out_pdwOutDataLen: int,
    dwTimeout: int,
    dwFlags: int

```

) → *ECErr*

Initiates a CoE SDO upload from an EtherCAT slave device to the master and returns immediately

A unique transfer ID must be written into EC_T_MBXTFER.dwTferId. EC_NOTIFY_MBOXRCV is given on completion.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **wObIndex** (*int*) – Object index
- **byObSubIndex** (*int*) – Object sub index. If Complete Access only 0 or 1 allowed.
- **pbyData** (*List[int]*) –
- **dwDataLen** (*int*) –
- **pdwOutDataLen** –
- **dwTimeout** (*int*) – Timeout [ms]
- **dwFlags** (*int*) – Mailbox Flags. Bit 0: set if Complete Access (EC_MAILBOX_FLAG_SDO_COMPLETE).
- **out_pdwOutDataLen** (*int*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer, or dwTimeout is EC_NOWAIT
- EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if slave has no mailbox support
- EC_E_INVALID_SLAVE_STATE if slave is in an invalid state for mailbox transfer
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if the ADS server is running
- EC_ERROR_CODES_COE “CoE SDO error code”

Return type

ECErr

```

CoeSdoUploadReq (
    pMbxTfer: object,
    dwSlaveId: int,
    wObIndex: int,
    byObSubIndex: int,
    dwTimeout: int,
    dwFlags: int
) → ECErr

```

Initiates a CoE SDO upload from an EtherCAT slave device to the master and returns immediately

A unique transfer ID must be written into EC_T_MBXTFER.dwTferId. EC_NOTIFY_MBOXRCV is given on completion.

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object created with emMbxTferCreate
- **dwSlaveId** (*int*) – Slave ID
- **wObIndex** (*int*) – Object index
- **byObSubIndex** (*int*) – Object sub index. If Complete Access only 0 or 1 allowed.
- **dwTimeout** (*int*) – Timeout [ms]
- **dwFlags** (*int*) – Mailbox Flags. Bit 0: set if Complete Access (EC_MAILBOX_FLAG_SDO_COMPLETE).

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceId is out of range, the input pointer is EC_NULL or contains EC_NULL pointer, or dwTimeout is EC_NOWAIT
- EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if slave has no mailbox support
- EC_E_INVALID_SLAVE_STATE if slave is in an invalid state for mailbox transfer
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if the ADS server is running
- EC_ERROR_CODES_COE “CoE SDO error code”

Return type

ECErr

```

ConfigAddJunctionRedundancyConnection (
    wHeadStationAddress: int,
    wHeadRedPort: int,
    wTailStationAddress: int,
    wTailRedPort: int
) → ECErr

```

Add a junction redundancy connection

Since there is no mechanism to configure junction redundancy in the ENI, this API allows adding junction redundancy connections which will be validated by the EC-Master stack. It has to be called after em-

ConfigLoad and prior to calling emConfigApply. Calling this API enables junction redundancy support implicitly.

Parameters

- **wHeadStationAddress** (*int*) – Station address of the junction redundancy head slave. Typically this is an EtherCAT junction.
- **wHeadRedPort** (*int*) – Port at head slave to which the junction redundancy cable is connected. Must be ESC_PORT_B.
- **wTailStationAddress** (*int*) – Station address of the junction redundancy tail slave
- **wTailRedPort** (*int*) – Port at tail slave to which the junction redundancy cable is connected. May not be ESC_PORT_A.

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

ConfigApply () → [ESError](#)

Apply the network configuration.

It has to be called after emConfigLoad.

Args:

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

ConfigExcludeSlave (*wStationAddress: int*) → [ESError](#)

Exclude a slave from the network configuration.

It has to be called after emConfigLoad and prior to calling emConfigApply.

Parameters

wStationAddress (*int*) – Station address of the slave to be excluded. A value of 0 excludes all slaves.

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

ConfigExtend (*bResetConfig: bool, dwTimeout: int*) → [ESError](#)

Extends the existing network configuration

This function extends the existing configuration described in the ENI to allow mailbox communication with unexpected slaves. After this function was called, unexpected slaves can reach PREOP state. After the configuration was extended, disconnecting any slave will generate a bus mismatch, because all the slaves are part of the configuration. Recalling this function with bResetConfig set to EC_FALSE will extend the configuration again by any new connected unexpected slaves. The previous extension is not deleted. Calling the function with bResetConfig set to EC_TRUE will reset all the previous extensions.

This function may not be called from within the JobTask's context.

Parameters

- **bResetConfig** (*bool*) – EC_TRUE: Extended configuration will be removed
- **dwTimeout** (*int*) – Timeout [ms]

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

ConfigIncludeSlave (*wStationAddress: int*) → [ESError](#)

Include a slave in the network configuration

Slaves that were previously excluded with `emConfigSlaveExclude` can be added again. It has to be called after `emConfigLoad` and prior to calling `emConfigApply`.

Parameters

wStationAddress (*int*) – Station address of the slave to be included. A value of 0 includes all slaves.

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

ConfigLoad (
eCnfType: DN_EC_T_CNF_TYPE,
pbyCnfData: List[int],
dwCnfDataLen: int
) → [ESError](#)

Load the network configuration.

In combination with `emConfigApply`, this function replaces `emConfigureNetwork` and must be called after the initialization. Among others the EtherCAT topology defined in the given XML configuration file will be stored internally.

A client must not be registered prior to calling this function. Existing client registrations will be dropped.

Parameters

- **eCnfType** (*DN_EC_T_CNF_TYPE*) – Type of configuration data provided
- **pbyCnfData** (*List[int]*) – Configuration data
- **dwCnfDataLen** (*int*) – Length of configuration data in byte

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if `dwInstanceID` is out of range, the input pointer is `EC_NULL` or contains `EC_NULL` pointer
- EC_E_NOMEMORY if some memory cannot be allocated
- EC_E_ADS_IS_RUNNING if the ADS server is running

Return type

[ESError](#)

ConfigSetPreviousPort (
wStationAddress: int,
wStationAddressPrev: int,
wPortPrev: int
) → [ESError](#)

Set previous port information of a slave

It has to be called after `emConfigLoad` and prior to calling `emConfigApply`.

Parameters

- **wStationAddress** (*int*) – Station address of the slave
- **wStationAddressPrev** (*int*) – Previous slave station address
- **wPortPrev** (*int*) – Previous port

Returns

EC_E_NOERROR or error code

Return type

[ECErrors](#)

```
ConfigureNetwork (
    eCnfType: DN_EC_T_CNF_TYPE,
    pbyCnfData: List[int],
    dwCnfDataLen: int
) → ECErrors
Configure the Network.
```

This function must be called after the initialization. Among others the EtherCAT topology defined in the given XML configuration file will be stored internally.

A client must not be registered prior to calling this function. Existing client registrations will be dropped.

Parameters

- **eCnfType** ([DN_EC_T_CNF_TYPE](#)) – Type of configuration data provided
- **pbyCnfData** ([List\[int\]](#)) – Filename / configuration data, or EC_NULL if eCnfType is
- **dwCnfDataLen** (*int*) – Length of configuration data in byte, or zero if eCnfType is

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized or eCnfType is eCnfType_GenPreopENI or eCnfType_GenOpENI and link is disconnected
- EC_E_INVALIDPARAM if dwInstanceID is out of range or pParms is EC_NULL contains some values out of range
- EC_E_LINK_DISCONNECTED if link is disconnected
- EC_E_FEATURE_DISABLED if a configured feature is not included in the license key
- EC_E_NOTSUPPORTED if a configured feature is not supported (e.g not compiled in the library)
- EC_E_CFGFILENOTFOUND if the ENI file cannot be found
- EC_E_WRONG_FORMAT if format errors have been detected in the ENI or EEPROM in case of eCnfType_GenPreopENI or eCnfType_GenOpENI
- EC_E_OEM_SIGNATURE_MISMATCH if the OEM signature in the ENI file doesn't match the used OEM key
- EC_E_ENI_ENCRYPTION_WRONG_VERSION if the ENI encryption version is not supported (e.g. the library is too old)
- EC_E_ENI_ENCRYPTED if the ENI is encrypted and no OEM key has been set
- EC_E_XML_CYCCMDS_MISSING if the ENI doesn't contain cyclic commands

- `EC_E_XML_ALSTATUS_READ_MISSING` if the ENI doesn't contain any read AL status command
- `EC_E_XML_CYCCMDS_SIZEMISMATCH` if the size of the cyclic commands in the ENI mismatch
- `EC_E_XML_INVALID_INP_OFF` if some input offsets in the ENI are invalid
- `EC_E_XML_INVALID_OUT_OFF` if some output offsets in the ENI are invalid
- `EC_E_XML_INVALID_CMD_WITH_RED` if the ENI contains LRW commands and cable redundancy is configured
- `EC_E_XML_PREV_PORT_MISSING` if some previous port information are missing in the ENI
- `EC_E_XML_DC_CYCCMDS_MISSING` if the DC related cyclic commands are missing in the ENI
- `EC_E_XML_AOE_NETID_INVALID` if the ENI contains some invalid NetID

Return type`ECErrror`

```

ControlMotion (
    dwCmd: EC_T_MOTION_CMD,
    pInData: List[int],
    wDataLen: int,
    out_pOutData: List[int],
    wOutLen: int
) → ECErrror
    Controls the motion

```

Parameters

- `dwCmd` (`EC_T_MOTION_CMD`) – Command
- `pInData` (`List[int]`) – Input data
- `wDataLen` (`int`) – Length of in data
- `out_pOutData` (`List[int]`) – Output data
- `wOutLen` (`int`) – Length of output data field

Returns

`EC_E_NOERROR` on success, otherwise an error code.

Return type`ECErrror`

```

static ConvStringFromByteArray (data: List[int]) → str
    Converts byte array to string (e.g. 01 02)

```

Parameters

`data` (`List[int]`) – Data

Returns

Data as string

Return type`str`

```

static ConvStringToByteArray (data: str) → List[int]
    Converts byte array from string (e.g. 01 02)

```

Parameters

`data` (`str`) – Data

Returns

Data as byte array

Return type*List[int]*

```

static ConvValueFromBytes (
    bytes_: List[int],
    type_: DN_EC_T_DEFTYPE,
    out_value: object
) → ECError
    Converts value to bytes

```

Parameters

- **bytes** – Bytes
- **type** – Data type
- **out_value** (*object*) – Value
- **bytes_** (*List[int]*) –
- **type_** (*DN_EC_T_DEFTYPE*) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECError*

```

static ConvValueFromString (type_: DN_EC_T_DEFTYPE, value: str) → object
    Converts value from string

```

Parameters

- **type** – Type
- **value** (*str*) – Value
- **type_** (*DN_EC_T_DEFTYPE*) –

Returns

Value as object

Return type*object*

```

static ConvValueToBytes (
    type_: DN_EC_T_DEFTYPE,
    value: object,
    out_bytes: List[int]
) → ECError
    Converts value to bytes

```

Parameters

- **type** – Data type
- **value** (*object*) – Value
- **out_bytes** (*List[int]*) – Bytes
- **type_** (*DN_EC_T_DEFTYPE*) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECError*

static ConvValueToString (*type_*: DN_EC_T_DEFTYPE, *value*: object) → str

Converts value to string

Parameters

- **type** – Type
- **value** (*object*) – Value
- **type_** (DN_EC_T_DEFTYPE) –

Returns

Value as string

Return type

str

static ConvertMasterStringFromBytes (

byData: List[int],

dwOffset: int,

dwLen: int

) → str

Converts a master string into UTF8 format

Parameters

- **abyData** – Byte array to convert
- **dwOffset** (*int*) – Offset
- **dwLen** (*int*) – Length
- **byData** (*List[int]*) –

Returns

Converted string

Return type

str

static CreateObjectFromBytes (*bytes_*: List[int], *ref_obj*: object) → *ECError*

Creates object from byte array

Parameters

- **bytes** – Object data as byte array
- **ref_obj** (*object*) – Object which should be filled
- **bytes_** (*List[int]*) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECError*

DcConfigure (*pDcConfigure*: DN_EC_T_DC_CONFIGURE) → *ECError*

Configure the distributed clocks.

- Set the DC synchronization settling time ([ms]).
- Set the DC slave limit for the wire or'ed clock deviation value. This value determines whether the slave clocks are synchronized or not.

- Configure the ARMW burst frames to compensate the static deviations of the clock speeds.

Parameters

pDcConfigure (*DN_EC_T_DC_CONFIGURE*) – Configuration parameter a pointer to a structure of type *EC_T_DC_CONFIGURE*

Returns

EC_E_NOERROR or error code

Return type

ECError

DcContDelayCompDisable () → *ECError*

Disable the continuous propagation delay compensation.

Args:

Returns

EC_E_NOERROR or error code

Return type

ECError

DcContDelayCompEnable () → *ECError*

Enable the continuous propagation delay compensation.

Calling this function will generate a propagation delay measurement every 30s. The result of the measurement is used to correct the propagation delay values on the bus.

Args:

Returns

EC_E_NOERROR or error code

Return type

ECError

DcDisable () → *ECError*

Disable DC Support

Args:

Returns

EC_E_NOERROR or error code

Return type

ECError

DcEnable () → *ECError*

Enable DC Support

Args:

Returns

EC_E_NOERROR or error code

Return type

ECError

DcIsEnabled (*out_pbDcIsEnabled: bool*) → *ECError*

Determines if DC is enabled and used.

Parameters

- **pbDcIsEnabled** – *EC_TRUE* if DC is enabled
- **out_pbDcIsEnabled** (*bool*) –

Returns

EC_E_NOERROR or error code

Return type

[EError](#)

DcmConfigure (*pDcmConfig*: [DN_EC_T_DCM_CONFIG](#), *dwInSyncTimeout*: *int*) → [EError](#)

Configure DC master synchronization

Parameters

- **pDcmConfig** ([DN_EC_T_DCM_CONFIG](#)) – Configuration information, a pointer to a structure of type [EC_T_DCM_CONFIG](#)
- **dwInSyncTimeout** (*int*) – Currently not implemented

Returns

EC_E_NOERROR or error code

Return type

[EError](#)

DcmGetAdjust (*out_pnAdjustPermil*: *int*) → [EError](#)

Returns the current adjustment value for the timer.

bCtlOff must be set to EC_TRUE in [EC_T_DCM_CONFIG](#) to enable external adjustment.

Parameters

- **pnAdjustPermil** – Current adjustment value of the timer
- **out_pnAdjustPermil** (*int*) –

Returns

EC_E_NOERROR or error code

Return type

[EError](#)

DcmGetBusShiftConfigured (*out_pbBusShiftConfigured*: *bool*) → [EError](#)

Determines if DCM Bus Shift is configured/possible in configuration (ENI file).

Parameters

- **pbBusShiftConfigured** – EC_TRUE if DCM bus shift mode is supported by the current configuration
- **out_pbBusShiftConfigured** (*bool*) –

Returns

EC_E_NOERROR or error code

Return type

[EError](#)

DcmGetLog (*out_pszLog*: *str*) → [EError](#)

Get logging information from the DCM controller.

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

[EError](#)

Parameters

out_pszLog (*str*) –

DcmGetStatus (
 out_pdwErrorCode: int,
 out_pnDiffCur: int,
 out_pnDiffAvg: int,
 out_pnDiffMax: int
) → *ECErr*
Get DC master synchronization controller status.

Parameters

- **pdwErrorCode** – Pointer to current error code of the DCM controller. Possible values are:
- **pnDiffCur** – Pointer to current difference between set value and actual value of controller in nanoseconds
- **pnDiffAvg** – Pointer to average difference between set value and actual value of controller in nanoseconds
- **pnDiffMax** – Pointer to maximum difference between set value and actual value of controller in nanoseconds
- **out_pdwErrorCode** (*int*) –
- **out_pnDiffCur** (*int*) –
- **out_pnDiffAvg** (*int*) –
- **out_pnDiffMax** (*int*) –

Returns

- EC_E_NOERROR if status retrieval was successful
- EC_E_NOTSUPPORTED if the DC feature is not supported/switched off. EC-Master stack has to be compiled with DC support see “define INCLUDE_DC_SUPPORT”.
- EC_NULL does not appear in normal flow, if EC_NULL is returned we are in a very exceptional case where m_poDcm == NULL (SW error detection)

Return type

ECErr

DcmResetStatus () → *ECErr*

Reset DC master synchronization controller status, average and maximum difference between set value and actual value

Args:

Returns

EC_E_NOERROR or error code

Return type

ECErr

DcmShowStatus () → *ECErr*

Show DC master synchronization status as DbgMsg (for development purposes only).

Args:

Returns

EC_E_NOERROR or error code

Return type

ECErr

DcxGetStatus (
out_pdwErrorCode: int,
out_pnDiffCur: int,
out_pnDiffAvg: int,
out_pnDiffMax: int,
out_pnTimeStampDiff: int
) → *ECErr*
 Get DC master external synchronization controller status.

Parameters

- **pdwErrorCode** – DCX controller error code
- **pnDiffCur** – Current difference between set value and actual value of controller in nanoseconds
- **pnDiffAvg** – Average difference between set value and actual value of controller in nanoseconds
- **pnDiffMax** – Maximum difference between set value and actual value of controller in nanoseconds
- **pnTimeStampDiff** – Difference between external and internal timestamps
- **out_pdwErrorCode** (*int*) –
- **out_pnDiffCur** (*int*) –
- **out_pnDiffAvg** (*int*) –
- **out_pnDiffMax** (*int*) –
- **out_pnTimeStampDiff** (*int*) –

Returns

EC_E_NOERROR or error code

Return type

ECErr

DeinitInstance ()
 Deinitialize EtherCAT wrapper

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

DeinitMotion () → *ECErr*
 Deinitialize the motion

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

ESCTypeText (*byEscType: int, bLong: bool*) → str
 Gets the text of corresponding ESC type

Parameters

- **byEscType** (*int*) – ESC type
- **bLong** (*bool*) – True: long text version, False: short text version
- **strInterfaceGuid** – Interface guid (optional)

Returns

Text of corresponding ESC type

Return type

str

EnableNotification (

dwClientId: int,
tNotifyCode: DN_NotifyCode,
bEnable: bool

) → *ECError*
Enables notification

Parameters

- **tNotifyCode** (DN_NotifyCode) – Code of notification, which should be enabled
- **bEnable** (bool) – True: Enables notification
- **dwClientId** (int) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

EnablePerformanceMeasuring (*bEnable*: bool)

Enables performance monitoring. For every master function, the notification “PerfNotification” will be called.

Parameters

bEnable (bool) – True = Enables performance monitoring

EnableTranslation (*bEnable*: bool)

Enables translation. For every master string, the notification “TranslateNotification” will be called.

Parameters

bEnable (bool) – True = Enables translation

EoeInstallEndpoint (

bCreateTxEvent: bool,
strInterfaceName: str,
strInterfaceGuid: str

) → *ECError*
Install EoE endpoint

Parameters

- **bCreateTxEvent** (bool) – Create TX Event
- **strInterfaceName** (str) – Interface name (optional)
- **strInterfaceGuid** (str) – Interface guid (optional)

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

EoeSendFrame (*wCfgFixedAddress*: int, *pbYFrame*: List[int], *dwFrameLen*: int) → *ECError*

Send EoE frame (queued)

Parameters

- **wCfgFixedAddress** (int) –

- **pbyFrame** (*List[int]*) –
- **dwFrameLen** (*int*) –

Returns

_ #EC_E_NOERROR or error code

Return type

ECErr

EoeTriggerTxEvent () → *ECErr*

Trigger TX Event, which must be called from JobTask after ExecJobProcessAllRxFrames()

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

EoeUninstallEndpoint () → *ECErr*

Uninstall EoE endpoint

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

EthDbgMsg (*byEthTypeByte0: int, byEthTypeByte1: int, szMsg: str*) → *ECErr*

Send a debug message to the EtherCAT Link Layer.

This feature can be used for debugging purposes.

Parameters

- **byEthTypeByte0** (*int*) – Ethernet type byte 0
- **byEthTypeByte1** (*int*) – Ethernet type byte 1
- **szMsg** (*str*) – Message to send to link layer

Returns

EC_E_NOERROR or error code

Return type

ECErr

ExecJob (*eUserJob: DN_EC_T_USER_JOB*) → *ECErr*

Execute or initiate the requested master job.

Parameters

eUserJob (*DN_EC_T_USER_JOB*) – Requested job.

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

ExecJobProcessAllRxFrames (*out_bPrevCycProcessed: bool*) → *ECErr*

calls the process all rx frame job

Parameters

- **bPrevCycProcessed** – True: previous send frame was received and processed, False: otherwise
- **out_bPrevCycProcessed** (*bool*) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

ExecJobSendCycFramesByTaskId (*dwTaskId: int*) → *ECError*

sends a cycle frame by its task id

Parameters

dwTaskId (*int*) – task id of the cyclic frame

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

FastModeInit () → *ECError*

Args:

Returns:

Return type

ECError

FastProcessAllRxFrames (*out_pbAreAllCycFramesProcessed: bool*) → *ECError*

Parameters

- **pbAreAllCycFramesProcessed** –
- **out_pbAreAllCycFramesProcessed** (*bool*) –

Return type

ECError

Returns:

FastSendAllCycFrames () → *ECError*

Args:

Returns:

Return type

ECError

FindInpVarByName (
szVariableName: str,
out_pProcessVarInfoEntry: DN_EC_T_PROCESS_VAR_INFO
) → *ECError*

Finds an input process variable information entry by the variable name.

Parameters

- **szVariableName** (*str*) – Variable name
- **pProcessVarInfoEntry** – Process variable information entry
- **out_pProcessVarInfoEntry** (*DN_EC_T_PROCESS_VAR_INFO*) –

Returns

EC_E_NOERROR or error code

Return type

ECError

FindInpVarByNameEx (
 szVariableName: *str*,
 out_pProcessVarInfoEntry: *DN_EC_T_PROCESS_VAR_INFO_EX*
) → *ECError*
Finds an input process variable extended information entry by the variable name.

Parameters

- **szVariableName** (*str*) – Variable name
- **pProcessVarInfoEntry** – Process variable extended information entry
- **out_pProcessVarInfoEntry** (*DN_EC_T_PROCESS_VAR_INFO_EX*) –

Returns

EC_E_NOERROR or error code

Return type

ECError

FindOutpVarByName (
 szVariableName: *str*,
 out_pSlaveOutpVarInfo: *DN_EC_T_PROCESS_VAR_INFO*
) → *ECError*
Finds an output process variable information entry by the variable name.

Parameters

- **szVariableName** (*str*) – Variable name
- **pSlaveOutpVarInfo** –
- **out_pSlaveOutpVarInfo** (*DN_EC_T_PROCESS_VAR_INFO*) –

Returns

EC_E_NOERROR or error code

Return type

ECError

FindOutpVarByNameEx (
 szVariableName: *str*,
 out_pProcessVarInfoEntry: *DN_EC_T_PROCESS_VAR_INFO_EX*
) → *ECError*
Finds an output process variable extended information entry by the variable name.

Parameters

- **szVariableName** (*str*) – Variable name
- **pProcessVarInfoEntry** – Process variable extended information entry
- **out_pProcessVarInfoEntry** (*DN_EC_T_PROCESS_VAR_INFO_EX*) –

Returns

EC_E_NOERROR or error code

Return type

ECError

```
FoeDownloadReq (
    pMbxTfer: object,
    dwSlaveId: int,
    achFileName: str,
    dwFileNameLen: int,
    dwPassword: int,
    dwTimeout: int
```

) → *ECErr*

Initiates an FoE File download to an EtherCAT slave device.

This function is used to download a complete file and returns immediately. After the download has been successfully completed or an error has occurred, EC_NOTIFY_MBOXRCV is raised. The progress of the file transfer is also notified with EC_NOTIFY_MBOXRCV.

EC_NOWAIT as a timeout is still accepted for reasons of compatibility and sets the timeout to 10 seconds

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object
- **dwSlaveId** (*int*) – Slave ID
- **achFileName** (*str*) – File name of slave file to write
- **dwFileNameLen** (*int*) – Length of slave file name in bytes
- **dwPassword** (*int*) – Slave password
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range or maximum file name length of 64 bytes (MAX_FILE_NAME_SIZE) exceeded
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_ERROR_CODES_FOE “FoE error codes”

Return type

ECErr

```
FoeFileDownload (
    dwSlaveId: int,
    achFileName: str,
    dwFileNameLen: int,
    pbData: List[int],
    dwDataLen: int,
    dwPassword: int,
    dwTimeout: int
```

) → *ECErr*

Execute an FoE File download to an EtherCAT slave device.

This function is used to download a complete file. The function returns after the download has been successfully completed or an error has occurred. This function may not be called from within the JobTask's context.

Parameters

- **dwSlaveId** (*int*) – Slave ID

- **achFileName** (*str*) – File name of slave file to write
- **dwFileNameLen** (*int*) – Length of slave file name in bytes
- **pbyData** (*List[int]*) – Buffer containing transferred data
- **dwDataLen** (*int*) – Buffer length [bytes]
- **dwPassword** (*int*) – Slave password
- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time.

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range or maximum file name length of 64 bytes (MAX_FILE_NAME_SIZE) exceeded
- EC_E_FOE_ERRCODE_NOTINBOOTSTRAP if slave in BOOTSTRAP and file-name not accepted by slave
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_ERROR_CODES_FOE “FoE error code”

Return type

[ESError](#)

```
FoeFileUpload(
    dwSlaveId: int,
    achFileName: str,
    dwFileNameLen: int,
    pbyData: List[int],
    dwDataLen: int,
    out_pdwOutDataLen: int,
    dwPassword: int,
    dwTimeout: int
```

) → [ESError](#)

Execute an FoE File upload from an EtherCAT slave device.

This function is used to upload a complete file. The function returns after the upload has been successfully completed or an error has occurred. This function may not be called from within the JobTask's context.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **achFileName** (*str*) – File name of slave file to read
- **dwFileNameLen** (*int*) – Length of slave file name in bytes
- **pbyData** (*List[int]*) – Buffer receiving transferred data
- **dwDataLen** (*int*) – Buffer length [bytes]
- **pdwOutDataLen** – Length of received data [byte]
- **dwPassword** (*int*) – Slave password
- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time.
- **out_pdwOutDataLen** (*int*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range or maximum file name length of 64 bytes (MAX_FILE_NAME_SIZE) exceeded
- EC_E_TIMEOUT if dwTimeout elapsed during the API call
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_ERROR_CODES_FOE "FoE error code"

Return type

`ECError`

`FoeSegmentedDownloadReq` (

pMbxTfer: object,
dwSlaveId: int,
szFileName: str,
dwFileNameLen: int,
dwFileSize: int,
dwPassword: int,
dwTimeout: int

) → `ECError`

Initiates or continues a segmented FoE File download to an EtherCAT slave device.

This function is used to download a file chunk-by-chunk and returns immediately. An `EC_NOTIFY_MBOXRCV` is raised to request the next chunk from the application or to provide information about the progress and the change in the transfer status. The slave may have a different mailbox size for `BOOTSTRAP` than for `PREOP`, `SAFEOP`, `OP`. See `EC_T_CFG_SLAVE_INFO.dwMbxInSize2`. The maximum chunk size is the slave's mailbox size - 12 bytes overhead for EtherCAT's FoE protocol. The mailbox transfer object's buffer must be at least as big as the chunks to be transferred.

`EC_NOWAIT` as a timeout is still accepted for reasons of compatibility and sets the timeout to 10 seconds

Parameters

- **`pMbxTfer`** (*object*) – Pointer to the corresponding mailbox transfer object. `EC_T_MBXTFER.pbyMbxTferData`: next chunk, `EC_T_MBXTFER.dwDataLen`: next chunk size.
- **`dwSlaveId`** (*int*) – Slave ID
- **`szFileName`** (*str*) – File name of slave file to write. Only evaluated when initiating the request.
- **`dwFileNameLen`** (*int*) – Length of slave file name in bytes
- **`dwFileSize`** (*int*) – Complete file size (mandatory). Used also for progress information. Only evaluated when initiating the request.
- **`dwPassword`** (*int*) – Slave password. Only evaluated when initiating the request.
- **`dwTimeout`** (*int*) – Overall timeout [ms] of the FoE transfer. Only evaluated when initiating the request.

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range or maximum file name length of 64 bytes (MAX_FILE_NAME_SIZE) exceeded

- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_ERROR_CODES_FOE “FoE error code”

Return type

[ECErrror](#)

FoeSegmentedUploadReq (

pMbxTfer: object,
dwSlaveId: int,
szFileName: str,
dwFileNameLen: int,
dwFileSize: int,
dwPassword: int,
dwTimeout: int

) → [ECErrror](#)

Initiates or continues a segmented FoE File upload from an EtherCAT slave device.

This function is used to upload a file chunk-by-chunk and returns immediately. An EC_NOTIFY_MBOXRCV is raised to provide the next chunk to the application or to get information about the progress and the change in the transfer status. The slave may have a different mailbox size for BOOTSTRAP than for PREOP, SAFEOP, OP. See EC_T_CFG_SLAVE_INFO.dwMbxInSize2. The maximum chunk size is the slave’s mailbox size - 12 bytes overhead for EtherCAT’s FoE protocol. The mailbox transfer object’s buffer must be at least as big as the chunks to be transferred.

EC_NOWAIT as a timeout is still accepted for reasons of compatibility and sets the timeout to 10 seconds

Parameters

- **pMbxTfer** (*object*) – Pointer to the corresponding mailbox transfer object. EC_T_MBXTFER.pbyMbxTferData: next chunk, EC_T_MBXTFER.dwDataLen: next chunk size.
- **dwSlaveId** (*int*) – Slave ID
- **szFileName** (*str*) – File name of slave file to write. Only evaluated when initiating the request.
- **dwFileNameLen** (*int*) – Length of slave file name in bytes
- **dwFileSize** (*int*) – Used only for progress information
- **dwPassword** (*int*) – Slave password. Only evaluated when initiating the request.
- **dwTimeout** (*int*) – Overall timeout [ms] of the FoE transfer. Only evaluated when initiating the request.

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn’t initialized
- EC_E_INVALIDPARG if dwInstanceID is out of range or maximum file name length of 64 bytes (MAX_FILE_NAME_SIZE) exceeded
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_ERROR_CODES_FOE “FoE error code”

Return type

[ECErrror](#)

FoeUploadReq (
pMbxTfer: object,
dwSlaveId: int,
achFileName: str,
dwFileNameLen: int,
dwPassword: int,
dwTimeout: int

) → *ECErr*

Initiates an FoE File upload from an EtherCAT slave device.

This function is used to upload a complete file and returns immediately. After the upload has been successfully completed or an error has occurred, EC_NOTIFY_MBOXRCV is raised. The progress of the file transfer is also notified with EC_NOTIFY_MBOXRCV.

EC_NOWAIT as a timeout is still accepted for reasons of compatibility and sets the timeout to 10 seconds

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object
- **dwSlaveId** (*int*) – Slave ID
- **achFileName** (*str*) – File name of slave file to read
- **dwFileNameLen** (*int*) – Length of slave file name in bytes
- **dwPassword** (*int*) – Slave password
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range or maximum file name length of 64 bytes (MAX_FILE_NAME_SIZE) exceeded
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_ERROR_CODES_FOE “FoE error code”

Return type

ECErr

ForceProcessDataBits (
dwClientId: int,
bOutputData: bool,
dwBitOffsetPd: int,
wBitLength: int,
pbData: List[int],
dwTimeout: int

) → *ECErr*

Force a specific number of bits from a given buffer to the process image with a bit offset.

All output data set by this API are overwriting the values set by the application. All input data set by this API are overwriting the values read from the slaves. Forcing will be terminated by calling the corresponding functions. This function may not be called from within the JobTask's context.

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)

- **bOutputData** (*bool*) – EC_TRUE: write output data, EC_FALSE: write input data
- **dwBitOffsetPd** (*int*) – Bit offset in Process data image
- **wBitLength** (*int*) –
- **pbyData** (*List[int]*) – Buffer containing transferred data
- **dwTimeout** (*int*) – Timeout [ms]. The timeout value must not be set to EC_NOWAIT.

Returns

EC_E_NOERROR or error code

Return type

ECError

ForceSlvStatCollection () → ECError

Sends datagrams to collect slave statistics counters.

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

ForceTopologyChange () → ECError

Force changed topology

Trigger HC State Machine.

Args:

Returns

EC_E_NOERROR or error code

Return type

ECError

GetBusScanSlaveInfoDesc (*wAutoIncAddr: int,**out_oSlaveInfoDesc: DN_EC_T_SB_SLAVEINFO_DESC*

) → ECError

This call will return the basic slave info determined in the last bus scan

Parameters

- **wAutoIncAddr** (*int*) – Auto increment address of the slave
- **oSlaveInfoDesc** – Out parameter that contains different slave information after the call
- **out_oSlaveInfoDesc** (*DN_EC_T_SB_SLAVEINFO_DESC*) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

GetBusSlaveInfo (*bStationAddress: bool,**wSlaveAddress: int,**out_pSlaveInfo: DN_EC_T_BUS_SLAVE_INFO*

) → ECError

Return information about a slave connected to the EtherCAT bus

Parameters

- **bStationAddress** (*bool*) –
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **pSlaveInfo** – Information from the slave
- **out_pSlaveInfo** (`DN_EC_T_BUS_SLAVE_INFO`) –

Returns

- `EC_E_NOERROR` if successful
- `EC_E_INVALIDSTATE` if EtherCAT stack isn't initialized
- `EC_E_INVALIDPARAM` if `dwInstanceID` is out of range
- `EC_E_NOTFOUND` if no slave matching `bFixedAddressing` / `wSlaveAddress` can be found

Return type

`ECError`

GetBusTime (*out_pqwBusTime: int*) → `ECError`

This function returns the actual bus time in nanoseconds.

Parameters

- **pqwBusTime** – Bus time [ns]
- **out_pqwBusTime** (*int*) –

Returns

`EC_E_NOERROR` or error code

Return type

`ECError`

GetCfgSlaveEoeInfo (
bStationAddress: bool,
wSlaveAddress: int,
out_pSlaveEoeInfo: DN_EC_T_CFG_SLAVE_EOE_INFO
) → `ECError`

Return EoE information about a configured slave from the ENI file

Parameters

- **bStationAddress** (*bool*) –
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **pSlaveEoeInfo** – Information about the slave
- **out_pSlaveEoeInfo** (`DN_EC_T_CFG_SLAVE_EOE_INFO`) –

Returns

- `EC_E_NOERROR` if successful
- `EC_E_INVALIDSTATE` if EtherCAT stack isn't initialized
- `EC_E_INVALIDPARAM` if `dwInstanceID` is out of range
- `EC_E_NOTFOUND` if no slave matching `bFixedAddressing` / `wSlaveAddress` can be found
- `EC_E_NO_MBX_SUPPORT` if the slave does not support mailbox communication
- `EC_E_NO_EOE_SUPPORT` if the slave supports mailbox communication, but not EoE

Return type*ECError***GetCfgSlaveInfo (***bStationAddress: bool,**wSlaveAddress: int,**out_pSlaveInfo: DN_EC_T_CFG_SLAVE_INFO***) → *ECError***

Return information about a configured slave from the ENI file

Parameters

- **bStationAddress** (*bool*) –
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **pSlaveInfo** – Information about the slave
- **out_pSlaveInfo** (*DN_EC_T_CFG_SLAVE_INFO*) –

Returns

EC_E_NOERROR or error code

Return type*ECError***GetCfgSlaveSmInfo (***bStationAddress: bool,**wSlaveAddress: int,**pSlaveSmInfo: DN_EC_T_CFG_SLAVE_SM_INFO***) → *ECError***

Return SyncManager information of a configured slave from the ENI file

Parameters

- **bStationAddress** (*bool*) –
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **pSlaveSmInfo** (*DN_EC_T_CFG_SLAVE_SM_INFO*) – Information about the slave.

Returns

EC_E_NOERROR or error code

Return type*ECError***GetCyclicConfigInfo (***out_oCyclicConfigInfo: List[DN_EC_T_CYC_CONFIG_DESC]***) → *ECError***

Returns an array of cyclic tasks.

Parameters

out_oCyclicConfigInfo (*List [DN_EC_T_CYC_CONFIG_DESC]*) – out Array of cyclic tasks

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECError***GetDiagnosisImageSize () → *ECError***

Gets the diagnosis image size

Args:

Returns

Diagnosis image size

Return type

[ESError](#)

GetErrorText (*eErrorCode*: [ESError](#)) → str
Return text tokens by Error code from master stack.

Parameters

eErrorCode ([ESError](#)) – Error code

Returns

Error text for supplied error code

Return type

str

GetHcGroupInfo (
 dwHcGroupIdx: int,
 pHcGroupInfo: [DN_EC_T_HC_GROUP_INFO](#)
) → [ESError](#)
Get informations of a specific Hot-Connect group.

Parameters

- **dwHcGroupIdx** (*int*) –
- **pHcGroupInfo** ([DN_EC_T_HC_GROUP_INFO](#)) – Hot connect group info

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

GetHcGroupNumOf () → [ESError](#)
Get number of Hot-Connect groups.

Args:

Returns

number of HC groups

Return type

[ESError](#)

GetLicenseFingerprint (*byMethod*: int, *szLicenseFingerprint*: str) → [ESError](#)
Gets the fingerprint needed for license key generation

Parameters

- **byMethod** (*int*) – Method used to generate the fingerprint
- **szLicenseFingerprint** (*str*) – Buffer for fingerprint string. Size must be at least EC_LICENSE_FINGERPRINT_STRSIZE.

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

GetMasterDump (*pbyBuffer*: List[int], *dwBufferSize*: int, *out_pdwDumpSize*: int) → [ESError](#)
The dump contains relevant information about the master and slave status.

The dump is only intended for internal troubleshooting at acontis. Amongst others it contains the following descriptors: - EC_T_INIT_MASTER_PARMS - EC_T_BUS_DIAGNOSIS_INFO

- EC_T_MAILBOX_STATISTICS - EC_T_CFG_SLAVE_INFO - EC_T_BUS_SLAVE_INFO - EC_T_SLVSTATISTICS_DESC

The buffer is written until all relevant data have been dumped or the buffer size has been exceeded.

Parameters

- **pbyBuffer** (*List[int]*) – Preallocated buffer to dump log data
- **dwBufferSize** (*int*) – Size of preallocated buffer
- **pdwDumpSize** – Size of master dump
- **out_pdwDumpSize** (*int*) –

Returns

- EC_E_NOERROR
- EC_E_NOMEMORY if buffer too small

Return type

[ESError](#)

GetMasterInfo (*out_pMasterInfo: DN_EC_T_MASTER_INFO*) → *ESError*

Get generic information about the Master

Parameters

- **pMasterInfo** – Master information
- **out_pMasterInfo** (*DN_EC_T_MASTER_INFO*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range or pParms is NULL or contains values out of range
- EC_E_ADS_IS_RUNNING if ADS server is running

Return type

[ESError](#)

GetMasterParms (*out_pParms: DN_EC_T_INIT_MASTER_PARMS*) → *ESError*

Get current Master initialization parameters.

If the given buffer is larger than the actual size of structure EC_T_INIT_MASTER_PARMS, the parameters of EC_T_INIT_MASTER_PARMS.pOsParms, EC_T_INIT_MASTER_PARMS.pLinkParms and EC_T_INIT_MASTER_PARMS.pLinkParmsRed are appended.

Parameters

- **pParms** – Buffer to store Master parameters
- **out_pParms** (*DN_EC_T_INIT_MASTER_PARMS*) –

Returns

- EC_E_NOERROR on success
- EC_E_INVALIDSTATE if master isn't initialized
- EC_E_INVALIDPARM if buffer pParms is too small

Return type

EError

GetMasterRedProcessImageInputPtr () → List[int]

Gets the Master Redundancy process data input image pointer

Args:

Returns

Master Redundancy process data input image pointer

Return type

List[int]

GetMasterRedProcessImageOutputPtr () → List[int]

Gets the Master Redundancy process data output image pointer

Args:

Returns

Master Redundancy process data output image pointer

Return type

List[int]

GetMasterRedState (*out_pbActive: bool*) → EError

Gets Master Redundancy State (ACTIVE / INACTIVE).

Parameters

- **pbActive** – Pointer to variable of type EC_T_BOOL
- **out_pbActive** (*bool*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized or MasterRed-Parms.bEnabled = EC_FALSE
- EC_E_INVALIDPARAM if dwInstanceID is out of range
- EC_E_NOTSUPPORTED if EC-Master stack does not include Master Redundancy support

Return type

EError

GetMasterState () → *DN_EC_T_STATE*

Get the EtherCAT master current state.

Args:

Returns

EtherCAT master state

Return type*DN_EC_T_STATE***GetMasterStateEx** (*out_pwCurrState: int, out_pwReqState: int*) → EError

Get the EtherCAT master current and requested state. Possible return values for current and requested state: - #DEVICE_STATE_UNKNOWN - #DEVICE_STATE_INIT - #DEVICE_STATE_PREOP - #DEVICE_STATE_SAFEOP - #DEVICE_STATE_OP

Parameters

- **pwCurrState** – Current master state

- **pwReqState** – Requested master state
- **out_pwCurrState** (*int*) –
- **out_pwReqState** (*int*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range or the output pointers are EC_NULL

Return type

[ESError](#)

GetMasterSyncUnitInfo (*wMsulId*: *int*, *out_pMsuInfo*: [DN_EC_T_MSU_INFO](#)) → [ESError](#)

Get number of Master Sync Units info entries.

Parameters

- **wMsulId** (*int*) –
- **pMsuInfo** –
- **out_pMsuInfo** ([DN_EC_T_MSU_INFO](#)) –

Returns

Number of Master Sync Units info entries

Return type

[ESError](#)

GetMasterSyncUnitInfoNumOf () → *int*

Get number of Master Sync Units info entries.

Args:

Returns

Number of Master Sync Units info entries

Return type

int

GetMemoryUsage (*out_pdwCurrentUsage*: *int*, *out_pdwMaxUsage*: *int*) → [ESError](#)

Returns information about memory usage.

All calls to malloc/free and new/delete are monitored.

Parameters

- **pdwCurrentUsage** – Current memory usage in Bytes at the time where this function is called
- **pdwMaxUsage** – Maximum memory usage in Bytes since initialization at the time where this function is called
- **out_pdwCurrentUsage** (*int*) –
- **out_pdwMaxUsage** (*int*) –

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

GetNumConfiguredSlaves () → int

Returns the number of slaves which are configured in the ENI.

Args:

Returns

Number of slaves

Return type

int

GetNumConnectedSlaves () → int

Get number of currently connected slaves.

Args:

Returns

Number of connected slaves

Return type

int

GetNumConnectedSlavesMain () → int

Get the number of currently connected Slaves to main interface.

Args:

Returns

Number of connected slaves at main interface

Return type

int

GetNumConnectedSlavesRed () → int

Get the number of currently connected Slaves to redundancy interface.

Args:

Returns

Number of connected slaves at redundancy interface

Return type

int

GetProcessData (

bOutputData: bool,

dwOffset: int,

pbyData: List[int],

dwLength: int,

dwTimeout: int

) → *ECError*

Blocking function to retrieve consistent process data from outside the JobTask context.

This function requests a copy of the process data (stored in RAM). The actual memcopy operation is executed by the JobTask to ensure data consistency. While waiting for the copy to complete, the calling context blocks and repeatedly calls sleep with an interval of at least the cycle time or one millisecond, whichever is greater. The function returns either with the requested process data once the copy is finished, or when the specified timeout has expired.

The function is blocking and should be used carefully in time-sensitive contexts and may not be called from within the JobTask context.

This function may not be called from within the JobTask's context.

Parameters

- **bOutputData** (*bool*) – EC_TRUE: read output data, EC_FALSE: read input data

- **dwOffset** (*int*) – Byte offset in Process data to read from
- **pbyData** (*List[int]*) – Buffer receiving transferred data
- **dwLength** (*int*) –
- **dwTimeout** (*int*) – Timeout [ms]

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

GetProcessDataBits (*bool*,

dwBitOffsetPd: int,

pbyDataDst: List[int],

dwBitLengthDst: int,

dwTimeout: int

) → [ESError](#)

Reads a specific number of bits from the process image to the given buffer with a bit offset (synchronized).

This function may not be called from within the JobTask's context.

Parameters

- **bOutputData** (*bool*) – EC_TRUE: read output data, EC_FALSE: write input data
- **dwBitOffsetPd** (*int*) – Bit offset in Process data image
- **pbyDataDst** (*List[int]*) –
- **dwBitLengthDst** (*int*) –
- **dwTimeout** (*int*) – Timeout [ms]. The timeout value must not be set to EC_NOWAIT.

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

GetProcessVarInfoEx (*DN_EC_T_VAR_DIRECTION*,

eVarSource: DN_EC_T_VAR_SOURCE,

bFixedAddress: bool,

wSlaveAddress: int,

aoVarInfo: DN_EC_T_PROCESS_VAR_INFO_EX,

dwMaxProcessVarInfoNumOf: int,

ref_pdwProcessVarInfoNumOf: int

) → [ESError](#)

Get process variables information

Parameters

- **eVarDirection** (*DN_EC_T_VAR_DIRECTION*) – INPUTs, OUTPUTs, See [EC_T_VAR_DIRECTION](#) .
- **eVarSource** (*DN_EC_T_VAR_SOURCE*) – Slave, Master, See [EC_T_VAR_SOURCE](#) .
- **bFixedAddress** (*bool*) – Use station address if EC_TRUE. Otherwise use AutoInc address.
- **wSlaveAddress** (*int*) – Slave address according to bFixedAddress

- **aoVarInfo** (`DN_EC_T_PROCESS_VAR_INFO_EX`) – The read process variable extended information entries
- **dwMaxProcessVarInfoNumOf** (*int*) –
- **pdwProcessVarInfoNumOf** –
- **ref_pdwProcessVarInfoNumOf** (*int*) –

Returns

EC_E_NOERROR or error code

Return type

`ECError`

GetProcessVarInfoNumOf (
eVarDirection: `DN_EC_T_VAR_DIRECTION`,
eVarSource: `DN_EC_T_VAR_SOURCE`,
bFixedAddress: *bool*,
wSlaveAddress: *int*,
ref_pdwProcessVarInfoNumOf: *int*
) → `ECError`
 Get process variables information

Parameters

- **eVarDirection** (`DN_EC_T_VAR_DIRECTION`) – INPUTs, OUTPUTs, See `EC_T_VAR_DIRECTION` .
- **eVarSource** (`DN_EC_T_VAR_SOURCE`) – Slave, Master, See `EC_T_VAR_SOURCE` .
- **bFixedAddress** (*bool*) – Use station address if `EC_TRUE`. Otherwise use `AutoInc` address.
- **wSlaveAddress** (*int*) – Slave address according to `bFixedAddress`
- **pdwProcessVarInfoNumOf** – Process variables count
- **ref_pdwProcessVarInfoNumOf** (*int*) –

Returns

EC_E_NOERROR or error code

Return type

`ECError`

GetRunMode () → `EcRunMode`
 Return run mode

Returns

Run mode

Return type

`EcRunMode`

GetScanBusStatus (*out_oSbStatus*: `DN_EC_T_SB_STATUS_NOTIFY_DESC`) → `ECError`
 Gets the status of the last bus scan.

Parameters

- **oSbStatus** – The last bus scan status
- **out_oSbStatus** (`DN_EC_T_SB_STATUS_NOTIFY_DESC`) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECError***static GetSizeOfObject** (*obj: object, out_size: int*) → *ECError*

Get byte size of object (as required from CoeSdoUpload)

Parameters

- **obj** (*object*) – Object
- **out_size** (*int*) – Byte size of object

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECError***GetSlaveFixedAddr** (*dwSlaveId: int, out_pwFixedAddr: int*) → *ECError*

Determine slave station address according to its slave ID.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **pwFixedAddr** – Corresponding fixed address
- **out_pwFixedAddr** (*int*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range or the output pointer is EC_NULL
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found

Return type*ECError***GetSlaveId** (*wStationAddress: int*) → *int*

Determines the slave ID using the slave station address.

Parameters**wStationAddress** (*int*) – Station address of the slave**Returns**

Slave ID or INVALID_SLAVE_ID if the slave could not be found or the stack is not initialized

Return type*int***GetSlaveIdAtPosition** (*wAutoIncAddress: int*) → *int*

Determines the slave ID using the slave auto increment address.

Parameters**wAutoIncAddress** (*int*) – Auto increment address of the slave**Returns**

Slave ID or INVALID_SLAVE_ID if no slave matching wAutoIncAddress can be found

Return type*int*

```

GetSlaveInfo (
    bFixedAddressing: bool,
    wSlaveAddress: int,
    out_pGetSlaveInfo: DN_EC_T_GET_SLAVE_INFO
) → ECErr
    Get Slave Info

```

Use `emGetCfgSlaveInfo` or `emGetBusSlaveInfo` instead

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **pGetSlaveInfo** – Slave information
- **out_pGetSlaveInfo** (DN_EC_T_GET_SLAVE_INFO) –

Returns

EC_E_NOERROR or error code

Return type

ECErr

```

GetSlaveInfoEx (
    oReq: DN_EC_T_SB_SLAVEINFO_REQ_DESC,
    out_oRes: DN_EC_T_SB_SLAVEINFO_RES_DESC
) → ECErr
    Gets the extended slave info determined in the last bus scan.

```

Parameters

- **oReq** (DN_EC_T_SB_SLAVEINFO_REQ_DESC) – Request parameter
- **out_oRes** (DN_EC_T_SB_SLAVEINFO_RES_DESC) – The extended slave information structure

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

```

GetSlaveInpVarByObjectEx (
    bFixedAddressing: bool,
    wSlaveAddress: int,
    wIndex: int,
    wSubIndex: int,
    out_pProcessVarInfoEntry: DN_EC_T_PROCESS_VAR_INFO_EX
) → ECErr
    Gets the input process variable extended information entry by object index, subindex of a specific slave.

```

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **wIndex** (*int*) – Object index
- **wSubIndex** (*int*) – Object sub index
- **pProcessVarInfoEntry** – Process variable extended information entry

- **out_pProcessVarInfoEntry** (DN_EC_T_PROCESS_VAR_INFO_EX) –

Returns

EC_E_NOERROR or error code

Return type

[EError](#)

GetSlaveInpVarInfo (

bFixedAddressing: bool,
wSlaveAddress: int,
wNumOfVarsToRead: int,
out_pSlaveProcVarInfoEntries: DN_EC_T_PROCESS_VAR_INFO,
out_pwReadEntries: int

) → [EError](#)

Gets the number of input variables of a specific slave.

Parameters

- **bFixedAddressing** (bool) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (int) – Slave address according bFixedAddressing
- **wNumOfVarsToRead** (int) –
- **pSlaveProcVarInfoEntries** –
- **pwReadEntries** –
- **out_pSlaveProcVarInfoEntries** (DN_EC_T_PROCESS_VAR_INFO) –
- **out_pwReadEntries** (int) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range or the output pointer is EC_NULL
- EC_E_NOTFOUND if no slave matching bFixedAddressing / wSlaveAddress can be found

Return type

[EError](#)

GetSlaveInpVarInfoEx (

bFixedAddressing: bool,
wSlaveAddress: int,
wNumOfVarsToRead: int,
out_pSlaveProcVarInfoEntries: DN_EC_T_PROCESS_VAR_INFO_EX,
out_pwReadEntries: int

) → [EError](#)

Gets the input process variable extended information entries of a specific slave.

Parameters

- **bFixedAddressing** (bool) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (int) – Slave address according bFixedAddressing
- **wNumOfVarsToRead** (int) – Number process variable entries that have been stored in pSlaveProcVarInfoEntries

- **pSlaveProcVarInfoEntries** – Number process variable entries that have been stored in pSlaveProcVarInfoEntries
- **pwReadEntries** – The number of read process variable information entries
- **out_pSlaveProcVarInfoEntries** (`DN_EC_T_PROCESS_VAR_INFO_EX`) –
- **out_pwReadEntries** (*int*) –

Returns

- `EC_E_NOERROR` if successful
- `EC_E_INVALIDSTATE` if EtherCAT stack isn't initialized
- `EC_E_INVALIDPARAM` if dwInstanceID is out of range or the output pointer is `EC_NULL`
- `EC_E_NOTFOUND` if no slave matching bFixedAddressing / wSlaveAddress can be found

Return type

`ECErr`

```
GetSlaveInpVarInfoNumOf (
    bFixedAddressing: bool,
    wSlaveAddress: int,
    out_pwSlaveInpVarInfoNumOf: int
) → ECErr
    Gets the number of input variables of a specific slave.
```

Parameters

- **bFixedAddressing** (*bool*) – `EC_TRUE`: use station address, `EC_FALSE`: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **pwSlaveInpVarInfoNumOf** – Number of found process variable entries
- **out_pwSlaveInpVarInfoNumOf** (*int*) –

Returns

- `EC_E_NOERROR` if successful
- `EC_E_INVALIDSTATE` if EtherCAT stack isn't initialized
- `EC_E_INVALIDPARAM` if dwInstanceID is out of range or the output pointer is `EC_NULL`
- `EC_E_NOTFOUND` if no slave matching bFixedAddressing / wSlaveAddress can be found

Return type

`ECErr`

```
GetSlaveOutpVarByObjectEx (
    bFixedAddressing: bool,
    wSlaveAddress: int,
    wIndex: int,
    wSubIndex: int,
    out_pProcessVarInfoEntry: DN_EC_T_PROCESS_VAR_INFO_EX
) → ECErr
    Gets the input process variable extended information entry by object index, subindex of a specific slave.
```

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **wIndex** (*int*) – Object index
- **wSubIndex** (*int*) – Object sub index
- **pProcessVarInfoEntry** – Process variable extended information entry
- **out_pProcessVarInfoEntry** (DN_EC_T_PROCESS_VAR_INFO_EX) –

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

```
GetSlaveOutpVarInfo (
    bFixedAddressing: bool,
    wSlaveAddress: int,
    wNumOfVarsToRead: int,
    out_pSlaveProcVarInfoEntries: DN_EC_T_PROCESS_VAR_INFO,
    out_pwReadEntries: int
) → ESError
    Gets the number of output variables of a specific slave.
```

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **wNumOfVarsToRead** (*int*) –
- **pSlaveProcVarInfoEntries** –
- **pwReadEntries** –
- **out_pSlaveProcVarInfoEntries** (DN_EC_T_PROCESS_VAR_INFO) –
- **out_pwReadEntries** (*int*) –

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

```
GetSlaveOutpVarInfoEx (
    bFixedAddressing: bool,
    wSlaveAddress: int,
    wNumOfVarsToRead: int,
    out_pSlaveProcVarInfoEntries: DN_EC_T_PROCESS_VAR_INFO_EX,
    out_pwReadEntries: int
) → ESError
    Gets the output process variable extended information entries of a specific slave.
```

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address

- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **wNumOfVarsToRead** (*int*) – Number of process variable information entries
- **pSlaveProcVarInfoEntries** – The read process extended variable entries
- **pwReadEntries** – The number of read process variable information entries
- **out_pSlaveProcVarInfoEntries** (DN_EC_T_PROCESS_VAR_INFO_EX)
–
- **out_pwReadEntries** (*int*) –

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

GetSlaveOutpVarInfoNumOf (

bFixedAddressing: *bool*,

wSlaveAddress: *int*,

out_pwSlaveOutpVarInfoNumOf: *int*

) → [ESError](#)

Gets the number of output variables of a specific slave.

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **pwSlaveOutpVarInfoNumOf** – Number of found process variables
- **out_pwSlaveOutpVarInfoNumOf** (*int*) –

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

GetSlavePortState (*dwSlaveId*: *int*, *out_pwPortState*: *int*) → [ESError](#)

Returns the state of the slave ports.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **pwPortState** – Slave port state.Format: www xxxx yyyy zzzz (each nibble : port 3210)
- **out_pwPortState** (*int*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range or the output pointer is EC_NULL
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found

Return type

[ESError](#)

GetSlaveProp (*dwSlaveId: int, out_pSlaveProp: DN_EC_T_SLAVE_PROP*) → bool

Determines the properties of the slave device.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **pSlaveProp** – Slave properties
- **out_pSlaveProp** (*DN_EC_T_SLAVE_PROP*) –

Returns

EC_TRUE if the slave exists, EC_FALSE if no slave matching dwSlaveId can be found

Return type

bool

GetSlaveState (*dwSlaveId: int, out_pwCurrDevState: int, out_pwReqDevState: int*) → *ECError*

Get the slave state.

The slave state is always read automatically from the AL_STATUS register whenever necessary. It is not forced by calling this function. This function may be called from within the JobTask's context.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **pCurrDevState** – Current slave state
- **pReqDevState** – Requested slave state
- **out_pCurrDevState** (*int*) –
- **out_pReqDevState** (*int*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range or the output pointers are EC_NULL
- EC_E_SLAVE_NOT_PRESENT if the slave is not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found

Return type

ECError

GetSlaveStatistics (
dwSlaveId: int,
out_pSlaveStatisticsDesc: DN_EC_T_SLVSTATISTICS_DESC
) → *ECError*

Get Slave's statistics counter.

Parameters

- **dwSlaveId** (*int*) – Slave id
- **pSlaveStatisticsDesc** – Pointer to structure EC_T_SLVSTATISTICS_DESC
- **out_pSlaveStatisticsDesc** (*DN_EC_T_SLVSTATISTICS_DESC*) –

Returns

EC_E_NOERROR or error code

Return type*ECError*

GetSlvStatistics (
dwSlaveId: int,
out_oStatistics: DN_EC_T_SLVSTATISTICS_DESC
) → *ECError*
Returns slave statistics counters.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **oStatistics** – out Statistics counters
- **out_oStatistics** (*DN_EC_T_SLVSTATISTICS_DESC*) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECError*

GetSrcMacAddress (*out_pMacSrc: DN_ETHERNET_ADDRESS*) → *ECError*
Gets the source MAC address

Parameters

- **pMacSrc** – 6-byte buffer to write source MAC address to
- **out_pMacSrc** (*DN_ETHERNET_ADDRESS*) –

Returns

EC_E_NOERROR or error code

Return type*ECError*

GetVersion (*out_pdwVersion: int*, *out_pdwVersionType: int*) → *ECError*
Gets the version information

Parameters

- **pdwVersion** – Pointer to EC_T_DWORD to carry out version number as a 32-bit value
- **pdwVersionType** – Pointer to EC_T_DWORD to carry out version type. See EC_VERSION_TYPES “EC_VERSION_TYPE”.
- **out_pdwVersion** (*int*) –
- **out_pdwVersionType** (*int*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range or the output pointer is EC_NULL

Return type*ECError*

HCAcceptTopoChange () → *ECError*

Accept last detected topology change

If Hot connect is configured in manual mode by `EC_IOCTL_HC_SETMODE`, the master will generate the notifications `EC_NOTIFY_HC_PROBEALLGROUPS` or `EC_NOTIFY_HC_DETECTADDDGROUPS` after a topology change was detected. This function will set all new detected slaves to the current master state.

Args:

Returns

`EC_E_NOERROR` or error code

Return type

ECError

HCGetNumGroupMembers (*dwGroupIndex: int*) → *ECError*

Get number of slaves belonging to a specific Hot-Connect group.

Parameters

dwGroupIndex (*int*) – Index of Hot-Connect group, 0 is the mandatory group

Returns

Number of slaves

Return type

ECError

HCGetSlaveIdsOfGroup (

dwGroupIndex: int,

out_adwSlaveId: int,

dwMaxNumSlaveIds: int

) → *ECError*

Get a list of Slave ID's belonging to a specific Hot-Connect group.

Parameters

- **dwGroupIndex** (*int*) – Index of Hot-Connect group, 0 is the mandatory group
- **adwSlaveId** – Preallocated Slave ID list buffer
- **dwMaxNumSlaveIds** (*int*) – Size of Slave ID list buffer
- **out_adwSlaveId** (*int*) –

Returns

`EC_E_NOERROR` or error code

Return type

ECError

InitInstance (*oParms, bStartTimingTask*) → *ECError*

Initializes EtherCAT wrapper

Parameters

- **oParms** – Parameters
- **bStartTimingTask** – Start the EcWrapper provided timing task

Returns

`EC_E_NOERROR` on success, otherwise an error code.

Return type

ECError

InitMotion () → *ECError*

Initialize the motion

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

Ioctl (

dwCode: int,
pbInBuf: List[int],
dwInBufSize: int,
pbOutBuf: List[int],
dwOutBufSize: int,
out_pdwNumOutData: int

) → *ECError*

A generic control interface between the application, the EtherCAT stack and its Link Layers.

Parameters

- **dwCode** (*int*) – IOCTL code (EC_IOCTL...)
- **pbInBuf** (*List[int]*) – IOCTL input parameters
- **dwInBufSize** (*int*) – Size of IOCTL input parameters in bytes
- **pbOutBuf** (*List[int]*) – Buffer for IOCTL output
- **dwOutBufSize** (*int*) – Size of buffer at pbOutBuf in bytes
- **pdwNumOutData** – Amount of bytes written to pbOutBuf by IOCTL. EC_NULL: amount not set by IOCTL.
- **out_pdwNumOutData** (*int*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer
- EC_E_NOMEMORY if memory cannot be allocated
- EC_E_ADS_IS_RUNNING if the ADS server is running

Return type

ECError

IsConfigured (*out_pbIsConfigured: bool*) → *ECError*

Returns if configuration has been applied

Parameters

- **pbIsConfigured** – EC_TRUE if configuration has been applied
- **out_pbIsConfigured** (*bool*) –

Returns

EC_E_NOERROR or error code

Return type

ECError

static IsRemoteServerUp (*abyIpAddr: List[int], wPort: int*) → bool

Checks if remote server is up by sending a “ping”

Parameters

- **abyIpAddr** (*List[int]*) – IP Address
- **wPort** (*int*) – Port

Returns

True, if server is up, False otherwise

Return type

bool

IsSlavePresent (*dwSlaveId: int, out_pbPresence: bool*) → *ECError*

Returns whether a specific slave is currently connected to the Bus.

This function may be called from within the JobTask.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **pbPresence** – EC_TRUE if the slave is currently connected to the bus, EC_FALSE if not
- **out_pbPresence** (*bool*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found

Return type

ECError

IsThrottledNotification (*tNotifyCode: DN_NotifyCode*) → bool

Checks if notification is throttled

Returns

True, if notification is throttled

Parameters

tNotifyCode (*DN_NotifyCode*) –

Return type

bool

IsTopologyChangeDetected (*out_pbTopologyChangeDetected: bool*) → *ECError*

Returns whether topology change detected.

Parameters

- **pbTopologyChangeDetected** – Pointer to EC_T_BOOL value: EC_TRUE if Topology Change Detected, EC_FALSE if not
- **out_pbTopologyChangeDetected** (*bool*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized

- **EC_E_INVALIDPARM** if `dwInstanceID` is out of range
- **EC_E_MASTER_RED_STATE_INACTIVE** if Master Redundancy is configured and master is inactive

Return type`ECErr`

IsTopologyKnown (*out_pbTopologyKnown: bool*) → `ECErr`

Returns whether topology known

Parameters

- **pbTopologyKnown** – Topology known
- **out_pbTopologyKnown** (*bool*) –

Returns

`EC_E_NOERROR` or error code

Return type`ECErr`

LinkInputVarByName (

szVarName: str,

pLinkVarInfo: DN_EC_T_LINK_VAR_INFO

) → `ECErr`

Link application variable to input process image. Input process variable is given by its name.

Parameters

- **szVarName** (*str*) – Name of variable
- **pLinkVarInfo** (*DN_EC_T_LINK_VAR_INFO*) – Information about linked variable

Returns

`EC_E_NOERROR` or an error code

Return type`ECErr`

LinkInputVarByObject (

bFixedAddressing: bool,

wSlaveAddress: int,

wIndex: int,

wSubIndex: int,

pLinkVarInfo: DN_EC_T_LINK_VAR_INFO

) → `ECErr`

Link application variable to input process image. Input process variable is given by its object index and subindex of a specific slave.

Parameters

- **bFixedAddressing** (*bool*) – Use station address if `EC_TRUE`. Otherwise use `AutoInc` address.
- **wSlaveAddress** (*int*) – Slave address according to `bFixedAddressing`
- **wIndex** (*int*) – Object index
- **wSubIndex** (*int*) – Object subindex
- **pLinkVarInfo** (*DN_EC_T_LINK_VAR_INFO*) – Information about linked variable

Returns

EC_E_NOERROR or an error code

Return type

[EError](#)

LinkOutputVarByName (

szVarName: *str*,
pLinkVarInfo: [DN_EC_T_LINK_VAR_INFO](#)

) → [EError](#)

Link application variable to output process image. Output process variable is given by its name.

Parameters

- **szVarName** (*str*) – Name of variable
- **pLinkVarInfo** ([DN_EC_T_LINK_VAR_INFO](#)) – Information about linked variable

Returns

EC_E_NOERROR or an error code

Return type

[EError](#)

LinkOutputVarByObject (

bFixedAddressing: *bool*,
wSlaveAddress: *int*,
wIndex: *int*,
wSubIndex: *int*,
pLinkVarInfo: [DN_EC_T_LINK_VAR_INFO](#)

) → [EError](#)

Link application variable to output process image. Output process variable is given by its object index and subindex of a specific slave.

Parameters

- **bFixedAddressing** (*bool*) – Use station address if EC_TRUE. Otherwise use AutoInc address.
- **wSlaveAddress** (*int*) – Slave address according to bFixedAddressing
- **wIndex** (*int*) – Object index
- **wSubIndex** (*int*) – Object subindex
- **pLinkVarInfo** ([DN_EC_T_LINK_VAR_INFO](#)) – Information about linked variable

Returns

EC_E_NOERROR or an error code

Return type

[EError](#)

MbxGatewayClntAddConnection (

oMbxGatewayParams: [DN_EC_T_INIT_MBXGATEWAY_PARMS](#)

) → [EError](#)

Establish connection to a mailbox gateway server.

Parameters

oMbxGatewayParams ([DN_EC_T_INIT_MBXGATEWAY_PARMS](#)) – Parameter for connection

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECError***MbxGatewayCIntRemoveConnection** () → *ECError*

Tear down an existing connection to a mailbox gateway server.

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECError***MbxGatewayCoeSdoDownload** (

wAddress: int,
wObIndex: int,
byObSubIndex: int,
pbyData: List[int],
dwDataLen: int,
dwTimeout: int,
dwFlags: int

) → *ECError*

Initiate CoE SDO download transfer to mailbox gateway server

Parameters

- **wAddress** (*int*) – slave address to download the SDO
- **wObIndex** (*int*) – object index
- **byObSubIndex** (*int*) – object sub-index
- **pbyData** (*List[int]*) –
- **dwDataLen** (*int*) –
- **dwTimeout** (*int*) – Timeout in milliseconds
- **dwFlags** (*int*) – mailbox transfer flags, see EC_MAILBOX_FLAG

Returns

Depends on the implementation of the function OsSetLastError(), normally returns EC_E_NOERROR if it succeeds or error code if it fails

Return type*ECError***MbxGatewayCoeSdoUpload** (

wAddress: int,
wObIndex: int,
byObSubIndex: int,
pbyData: List[int],
dwDataLen: int,
pdwOutDataLen: int,
dwTimeout: int,
dwFlags: int

) → *ECError*

Initiate CoE SDO upload transfer from mailbox gateway server

Parameters

- **wAddress** (*int*) – slave address to upload the SDO
- **wObIndex** (*int*) – object index
- **byObSubIndex** (*int*) – object sub-index
- **pbyData** (*List[int]*) –

- **dwDataLen** (*int*) –
- **dwTimeout** (*int*) – Timeout in milliseconds
- **dwFlags** (*int*) – mailbox transfer flags, see EC_MAILBOX_FLAG
- **pdwOutDataLen** (*int*) –

Returns

Depends on the implementation of the function `OsSetLastError()`, normally returns `EC_E_NOERROR` if it succeeds or error code if it fails

Return type

`ECError`

MbxTferAbort (*pMbxTfer: object*) → `ECError`

Abort a running mailbox transfer.

This function may not be called from within the `JobTask`'s context.

Parameters

pMbxTfer (*object*) – Mailbox transfer object created with `emMbxTferCreate`

Returns

`EC_E_NOERROR` if successful

Return type

`ECError`

MbxTferCopyFrom (
pMbxTfer: object,
abyData: List[int],
dwDataLen: int,
out_pdwOutDataLen: int

) → `ECError`
Copies data from the mailbox transfer buffer

Parameters

- **pMbxTfer** (*object*) – mailbox transfer object
- **abyData** (*List[int]*) – `abyData`
- **dwDataLen** (*int*) – `dwDataLen`
- **out_pdwOutDataLen** (*int*) – `pdwOutDataLen`

Returns

`EC_E_NOERROR` on success, otherwise an error code.

Return type

`ECError`

MbxTferCopyTo (*pMbxTfer: object*, *abyData: List[int]*, *dwDataLen: int*) → `ECError`

Copies data to the mailbox transfer buffer

Parameters

- **pMbxTfer** (*object*) – mailbox transfer object
- **abyData** (*List[int]*) – `abyData`
- **dwDataLen** (*int*) – `dwDataLen`

Returns

`EC_E_NOERROR` on success, otherwise an error code.

Return type

`ECError`

MbxTferCreate (*dwTferId: int, dwBufferSize: int, out_pMbxTfer: object*) → *ECError*

Creates a mailbox transfer object

Parameters

- **dwTferId** (*int*) – transfer ID (optional, can be 0)
- **dwBufferSize** (*int*) – buffer size
- **out_pMbxTfer** (*object*) – out mailbox transfer object

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

MbxTferDelete (*pMbxTfer: object*)

Deletes a mailbox transfer object.

A transfer object may only be deleted if it is in the Idle state.

Parameters

pMbxTfer (*object*) – Mailbox transfer object created with emMbxTferCreate

MbxTferReset (*pMbxTfer: object*)

Creates a mailbox transfer object

Parameters

pMbxTfer (*object*) – mailbox transfer object

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

MbxTferWait (*pMbxTfer: object*) → *ECError*

Waits until mailbox transfer is finished

Parameters

pMbxTfer (*object*) – mailbox transfer object

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

NotifyApp (

dwCode: int,

pInData: List[int],

wDataLen: int,

out_pOutData: List[int],

wOutLen: int,

out_pdwOutDataLen: int

) → *ECError*

Notifies the master application

Parameters

- **dwCode** (*int*) – Code
- **pInData** (*List[int]*) – Input data
- **wDataLen** (*int*) – Length of in data
- **out_pOutData** (*List[int]*) – Output data

- **wOutLen** (*int*) – Length of output data field
- **out_pdwOutDataLen** (*int*) – Length of actual out data

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErrors

OnHandleEcNotification (

type: *DN_NotifyType*,
code: *DN_NotifyCode*,
data: *object*,
_errMsgs: *List[str]*

)

Callback function that will be called after the scan bus has been finished. The scan bus result will be stored in *m_eLastScanBusRes*.

Parameters

- **type** – Type
- **code** (*DN_NotifyCode*) – Code
- **data** (*object*) – Data
- **errMsgs** – Error messages
- **_type_** (*DN_NotifyType*) –
- **_errMsgs** (*List[str]*) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErrors

OpenBlockedPorts (*dwTimeout*: *int*) → *ECErrors*

Opens all blocked ports

This call allows re-opening all blocked ports to check whether mismatch cause is removed from bus. This function may only be called from within the JobTask's context with parameter *dwTimeout* set to EC_NOWAIT.

Parameters

dwTimeout (*int*) – Timeout [ms]. The function will block at most for this time.

Returns

EC_E_NOERROR or error code

Return type

ECErrors

PassThroughSrvDisable (*dwTimeout*: *int*) → *ECErrors*

Disables the Pass-through server.

Parameters

dwTimeout (*int*) – Timeout [ms]

Returns

EC_E_NOERROR or error code

Return type

ECErrors

PassThroughSrvEnable (*dwTimeout*: *int*) → *ECErrors*

Enables the Pass-through server.

Parameters**dwTimeout** (*int*) – Timeout [ms]**Returns**

EC_E_NOERROR or error code

Return type

ESError

PassThroughSrvGetStatus () → *DN_EC_PTS_STATE*

Gets the status of the Pass-through server.

Args:

Returns

EC_E_NOERROR or error code

Return type*DN_EC_PTS_STATE***PassThroughSrvStart** (*poPtsStartParams*: *DN_EC_T_PTS_SRV_START_PARAMS*,*dwTimeout*: *int*) → *ESError*

Starts the Pass Through Server

Parameters

- **poPtsStartParams** (*DN_EC_T_PTS_SRV_START_PARAMS*) – Pass through server start parameter
- **dwTimeout** (*int*) – Timeout

Returns

EC_E_NOERROR or error code

Return type

ESError

PassThroughSrvStop (*dwTimeout*: *int*) → *ESError*

Stops the Pass Through Server

Parameters**dwTimeout** (*int*) – Timeout**Returns**

EC_E_NOERROR or error code

Return type

ESError

PerfMeasAppCreate (*pPerfMeasAppParams*: *DN_EC_T_PERF_MEAS_APP_PARAMS*,*out_ppvPerfMeas*: *object*) → *ESError*

Create PerfMeas object and bind it to the master instance

This API can be called multiple times to create PerfMeas objects. The performance counters in each of the objects can be accessed in the following two ways: - by passing the PerfMeas object and the index of the performance measurement. The index ranges from [0-pPerfMeasAppParams->dwNumAppMeas] - by passing EC_NULL instead of a PerfMeas object and an index. In this case the index works across all PerfMeas objects bound to the master instance.

Parameters

- **pPerfMeasAppParams** (*DN_EC_T_PERF_MEAS_APP_PARAMS*) – Pointer to parameter definitions

- **ppvPerfMeas** – Created PerfMeas object.
- **out_ppvPerfMeas** (*object*) –

Returns

EC_E_NOERROR or an error code

Return type

[EError](#)

PerfMeasAppDelete (*pvPerfMeas: object*) → [EError](#)

Delete application performance measurement and unbind it from the master instance

Objects which are not deleted using PerfMeasAppDelete are automatically deleted when calling DeinitMaster.

ote This invalidates the global index used when passing EC_NULL into the other PerfMeasApp functions

Args:

pvPerfMeas: PerfMeas object to delete

Returns:

EC_E_NOERROR or an error code

Parameters

ppvPerfMeas (*object*) –

Return type

[EError](#)

PerfMeasAppEnd (*pvPerfMeas: object, dwIndex: int*) → [EError](#)

Stop application performance measurement

Parameters

- **pvPerfMeas** (*object*) – PerfMeas object or EC_NULL to use continuous index
- **dwIndex** (*int*) – Index of the performance measurement

Returns

EC_E_NOERROR or an error code

Return type

[EError](#)

PerfMeasAppGetInfo (

pvPerfMeas: object,

dwIndex: int,

out_pPerfMeasInfo: List[DN_EC_T_PERF_MEAS_INFO],

dwPerfMeasNumOf: int

) → [EError](#)

Get general info about one/all application performance measurement

Parameters

- **pvPerfMeas** (*object*) – PerfMeas object or EC_NULL to use continuous index
- **dwIndex** (*int*) – Index of the performance measurement information, use 0xFFFFFFFF to get all
- **pPerfMeasInfo** – Pointer to a buffer receiving one/all performance measurement information
- **dwPerfMeasNumOf** (*int*) – Number of elements allocated in pPerfMeasInfo
- **out_pPerfMeasInfo** (*List[DN_EC_T_PERF_MEAS_INFO]*) –

Returns

EC_E_NOERROR or an error code

Return type

[ESError](#)

PerfMeasAppGetNumOf (*pvPerfMeas: object, out_pdwNumOf: int*) → [ESError](#)

Reset number of application performance measurement

Parameters

- **pvPerfMeas** (*object*) – PerfMeas object or EC_NULL to get the number of performance measurements in all PerfMeas objects
- **pdwNumOf** – Number of performance measurements
- **out_pdwNumOf** (*int*) –

Returns

EC_E_NOERROR or an error code

Return type

[ESError](#)

PerfMeasAppGetRaw (*pvPerfMeas: object,*

dwIndex: int,

ref_pPerfMeasVal: List[DN_EC_T_PERF_MEAS_VAL],

ref_pPerfMeasHistogram: List[DN_EC_T_PERF_MEAS_HISTOGRAM],

dwPerfMeasNumOf: int

) → [ESError](#)

Get raw data of one/all application performance measurement

Parameters

- **pvPerfMeas** (*object*) – PerfMeas object or EC_NULL to use continuous index
- **dwIndex** (*int*) – Index of the performance measurement, use 0xFFFFFFFF to get all
- **pPerfMeasVal** – Pointer to a buffer receiving one/all performance measurement values or EC_NULL
- **pPerfMeasHistogram** – Pointer to a buffer receiving one/all performance measurement histograms or EC_NULL
- **dwPerfMeasNumOf** (*int*) – Number of elements allocated in pPerfMeasVal and pPerfMeasHistogram
- **ref_pPerfMeasVal** (*List [DN_EC_T_PERF_MEAS_VAL]*) –
- **ref_pPerfMeasHistogram** (*List [DN_EC_T_PERF_MEAS_HISTOGRAM]*) –

Returns

EC_E_NOERROR or an error code

Return type

[ESError](#)

PerfMeasAppReset (*pvPerfMeas: object, dwIndex: int*) → [ESError](#)

Reset application performance measurement

Parameters

- **pvPerfMeas** (*object*) – PerfMeas object or EC_NULL to use continuous index

- **dwIndex** (*int*) – Index of the performance measurement, use 0xFFFFFFFF to reset all

Returns

EC_E_NOERROR or an error code

Return type

EError

PerfMeasAppStart (*pvPerfMeas: object, dwIndex: int*) → EError

Start application performance measurement

Parameters

- **pvPerfMeas** (*object*) – PerfMeas object or EC_NULL to use continuous index
- **dwIndex** (*int*) – Index of the performance measurement

Returns

EC_E_NOERROR or an error code

Return type

EError

PerfMeasGetInfoByTaskId (
dwTaskId: int,
dwIndex: int,
out_pPerfMeasInfo: List[DN_EC_T_PERF_MEAS_INFO],
dwPerfMeasNumOf: int
) → EError

Get general info about one/all internal performance measurement

Parameters

- **dwTaskId** (*int*) – Task Job ID
- **dwIndex** (*int*) – Index of the performance measurement, use 0xFFFFFFFF to get all
- **pPerfMeasInfo** – Pointer to a buffer receiving one/all performance measurement infos
- **dwPerfMeasNumOf** (*int*) – Number of elements allocated in pPerfMeasInfo
- **out_pPerfMeasInfo** (*List [DN_EC_T_PERF_MEAS_INFO]*) –

Returns

EC_E_NOERROR or an error code

Return type

EError

PerfMeasGetNumOfByTaskId (*dwTaskId: int, out_pdwNumOf: int*) → EError

Reset number of internal performance measurement

Parameters

- **dwTaskId** (*int*) – Task Job ID
- **pdwNumOf** – Number of performance measurements
- **out_pdwNumOf** (*int*) –

Returns

EC_E_NOERROR or an error code

Return type

EError

PerfMeasGetRawByTaskId (
dwTaskId: int,
dwIndex: int,
ref_pPerfMeasVal: List[DN_EC_T_PERF_MEAS_VAL],
ref_pPerfMeasHistogram: List[DN_EC_T_PERF_MEAS_HISTOGRAM],
dwPerfMeasNumOf: int
) → *ECError*
 Get raw data of one/all application performance measurement

Parameters

- **dwTaskId** (*int*) – Task Job ID
- **dwIndex** (*int*) – Index of the performance measurement, use 0xFFFFFFFF to get all
- **pPerfMeasVal** – Pointer to a buffer receiving one/all performance measurement values or EC_NULL
- **pPerfMeasHistogram** – Pointer to a buffer receiving one/all performance measurement histograms or EC_NULL
- **dwPerfMeasNumOf** (*int*) – Number of elements allocated in pPerfMeasVal and pPerfMeasHistogram
- **ref_pPerfMeasVal** (*List [DN_EC_T_PERF_MEAS_VAL]*) –
- **ref_pPerfMeasHistogram** (*List [DN_EC_T_PERF_MEAS_HISTOGRAM]*) –

Returns

EC_E_NOERROR or an error code

Return type

ECError

PerfMeasResetByTaskId (*dwTaskId: int, dwIndex: int*) → *ECError*
 Reset internal performance measurement

Parameters

- **dwTaskId** (*int*) – Task Job ID
- **dwIndex** (*int*) – Index of the performance measurement, use 0xFFFFFFFF to reset all

Returns

EC_E_NOERROR or an error code

Return type

ECError

QueueRawCmd (
wInvokeId: int,
byCmd: int,
dwMemoryAddress: int,
pbData: List[int],
wLen: int
) → *ECError*
 Transfers a raw EtherCAT command to one or multiple slaves.

All registered clients will be notified. This function may not be called from within the JobTask's context.

Parameters

- **wInvokeId** (*int*) – Invoke ID to reassign the results to the sent CMD

- **byCmd** (*int*) – EtherCAT command
- **dwMemoryAddress** (*int*) – Slave memory address, depending on the command to be sent this is either a physical or a logical address
- **pbyData** (*List[int]*) – [in, out] Buffer containing or receiving transferred data. In case a read-only command is queued (e.g. APRD) this pointer should be set to a value of EC_NULL.
- **wLen** (*int*) – Number of bytes to transfer

Returns

EC_E_NOERROR or error code

Return type

ECError

RasClntAddConnection (*oRasParams: DN_EC_T_INITRASPAMS*) → *ECError*

Establish connection to a remote server.

Parameters

oRasParams (*DN_EC_T_INITRASPAMS*) – Parameter for connection

Returns

Error code

Return type

ECError

RasGetConnectionInfo (*out_pConInfo: DN_EC_T_RAS_CONNECTION_INFO*) → *ECError*

Get actual connection information

Parameters

- **pConInfo** – connection information
- **out_pConInfo** (*DN_EC_T_RAS_CONNECTION_INFO*) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

ReadIdentifyObj (*wFixedAddr: int*)

Reads the identify object of a slave

Parameters

wFixedAddr (*int*) – Fixed station address

static ReadPdBitsFromAddress (

address: object,
bitOffset: int,
bitLength: int,
out_pdBits: List[int]

)

Reads process data bits from address

Parameters

- **address** (*object*) – Address
- **bitOffset** (*int*) – Bit offset
- **bitLength** (*int*) – Bit length
- **out_pdBits** (*List[int]*) – Process data bits

```
static ReadPdBitsFromBytes (  
    bytes_: List[int],  
    bitOffset: int,  
    bitLength: int,  
    out_pdBits: List[int]  
)
```

Reads process data bits from bytes

Parameters

- **bytes** – Bytes
- **bitOffset** (*int*) – Bit offset
- **bitLength** (*int*) – Bit length
- **out_pdBits** (*List [int]*) – Process data bits
- **bytes_** (*List [int]*) –

```
static ReadPdBytesFromAddress (  
    address: object,  
    offset: int,  
    length: int,  
    out_pdBytes: List[int]  
)
```

Reads process data bytes from address

Parameters

- **address** (*object*) – Address
- **offset** (*int*) – Offset
- **length** (*int*) – Length
- **out_pdBytes** (*List [int]*) – Process data bytes

```
static ReadPdBytesFromBytes (  
    bytes_: List[int],  
    offset: int,  
    length: int,  
    out_pdBytes: List[int]  
)
```

Reads process data bytes from bytes

Parameters

- **bytes** – Bytes
- **offset** (*int*) – Offset
- **length** (*int*) – Length
- **out_pdBytes** (*List [int]*) – Process data bytes
- **bytes_** (*List [int]*) –

```

ReadSlaveEEPROM (
    bFixedAddressing: bool,
    wSlaveAddress: int,
    wEEPROMStartOffset: int,
    pwReadData: List[int],
    dwReadLen: int,
    out_pdwNumOutData: int,
    dwTimeout: int
) → ECErr
    Read EEPROM data from slave

```

Parameters

- **bFixedAddressing** (*bool*) – True: use station addressing, False: use auto increment addressing
- **wSlaveAddress** (*int*) – Slave Address, station or auto increment address depending on bFixedAddressing
- **wEEPROMStartOffset** (*int*) – Address to start EEPROM Read from
- **pwReadData** (*List[int]*) – Pointer to ushort array to carry the read data
- **dwReadLen** (*int*) – Size of the ushort array provided at pwReadData (in ushorts)
- **out_pdwNumOutData** (*int*) – out Pointer to uint carrying actually read data (in ushorts) after completion
- **dwTimeout** (*int*) – Timeout in milliseconds. The function will block at most for this time. The timeout value must not be set to EC_NOWAIT

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

```

ReadSlaveEEPROMReq (
    dwClientId: int,
    dwTferId: int,
    bFixedAddressing: bool,
    wSlaveAddress: int,
    wEEPROMStartOffset: int,
    out_pwReadData: int,
    dwReadLen: int,
    ref_pdwNumOutData: int,
    dwTimeout: int
) → ECErr
    Requests an EEPROM data read operation from a slave and returns immediately.

```

An EC_NOTIFY_EEPROM_OPERATION is given on completion or timeout. This function may be called from within the JobTask's context.

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **dwTferId** (*int*) – Transfer ID. The application can set this ID to identify the transfer. It will be passed back to the application within EC_T_EEPROM_OPERATION_NOTIFY_DESC.
- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address

- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **wEEPROMStartOffset** (*int*) – Word address to start EEPROM read from
- **pwReadData** – Pointer to EC_T_WORD array to carry the read data, must be valid until the operation complete
- **dwReadLen** (*int*) – Size of the EC_T_WORD array provided at pwReadData (in EC_T_WORDS)
- **pdwNumOutData** – Pointer to EC_T_DWORD carrying actually read data (in EC_T_WORDS) after completion
- **dwTimeout** (*int*) – Timeout [ms]
- **out_pwReadData** (*int*) –
- **ref_pdwNumOutData** (*int*) –

Returns

EC_E_NOERROR or error code

Return type

ECError

ReadSlaveEEPROMWithReq (

dwClientId: int,
dwTferId: int,
bFixedAddressing: bool,
wSlaveAddress: int,
wEEPROMStartOffset: int,
pwReadData: List[int],
dwReadLen: int,
out_pdwNumOutData: int,
dwTimeout: int

) → *ECError*

Read EEPROM data from slave (REQ)

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **dwTferId** (*int*) – Transfer ID. The application can set this ID to identify the transfer. It will be passed back to the application within EC_T_SLAVEREGISTER_TRANSFER_NOTIFY_DESC.
- **bFixedAddressing** (*bool*) – True: use station addressing, False: use auto increment addressing
- **wSlaveAddress** (*int*) – Slave Address, station or auto increment address depending on bFixedAddressing
- **wEEPROMStartOffset** (*int*) – Address to start EEPROM Read from
- **pwReadData** (*List[int]*) – Pointer to ushort array to carry the read data
- **dwReadLen** (*int*) – Size of the ushort array provided at pwReadData (in ushorts)
- **out_pdwNumOutData** (*int*) – out Pointer to uint carrying actually read data (in ushorts) after completion
- **dwTimeout** (*int*) – Timeout in milliseconds. The function will block at most for this time. The timeout value must not be set to EC_NOWAIT

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECErrror***ReadSlaveIdentification (***bFixedAddressing: bool,**wSlaveAddress: int,**wAdo: int,**out_pwValue: int,**dwTimeout: int***) → *ECErrror***

Read identification value from a slave.

This function may not be called from within the JobTask's context.

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **wAdo** (*int*) – ADO used for identification command
- **pwValue** – Pointer to Word value containing the Identification value
- **dwTimeout** (*int*) – Timeout [ms]
- **out_pwValue** (*int*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range or the command is not supported or the timeout value is set to EC_NOWAIT
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching bFixedAddressing / wSlaveAddress can be found
- EC_E_TIMEOUT if dwTimeout elapsed during the API call
- EC_E_BUSY if another transfer request is already pending or the master or the corresponding slave is currently changing its operational state
- EC_E_NOTREADY if the working counter was not set when sending the command (slave may not be connected or did not respond)
- EC_E_ADO_NOT_SUPPORTED if the slave does not support requesting ID mechanism
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive

Return type*ECErrror*

ReadSlaveIdentificationReq (

dwClientId: int,
dwTferId: int,
bFixedAddressing: bool,
wSlaveAddress: int,
wAdo: int,
out_pwValue: int,
dwTimeout: int

) → *ECErr*

Request the identification value from a slave and returns immediately.

A notification EC_NOTIFY_SLAVE_IDENTIFICATION is given on completion or timeout. This function may be called from within the JobTask's context.

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **dwTferId** (*int*) – Transfer ID. The application can set this ID to identify the transfer. It will be passed back to the application within EC_T_SLAVE_IDENTIFICATION_NTIFY_DESC.
- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **wAdo** (*int*) – ADO used for identification command
- **pwValue** – Pointer to Word value containing the Identification value, must be valid until the request complete.
- **dwTimeout** (*int*) – Timeout [ms]
- **out_pwValue** (*int*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range or the command is not supported or the timeout value is set to EC_NOWAIT
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching bFixedAddressing / wSlaveAddress can be found
- EC_E_ADO_NOT_SUPPORTED if the slave does not support requesting ID mechanism

Return type*ECErr***ReadSlaveRegister** (

bFixedAddressing: bool,
wSlaveAddress: int,
wRegisterOffset: int,
phyData: List[int],
wLen: int,
dwTimeout: int

) → *ECErr*

Reads data from the ESC memory that have so far been transferred to a slave and received by the EC-Monitor.

!(EC_MONITOR)

Reads data from the ESC memory of a specified slave.

This function may not be called from within the JobTask's context.

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **wRegisterOffset** (*int*) – Register offset. I.e. use 0x0130 to read the AL Status register.
- **pbyData** (*List[int]*) – Buffer receiving transferred data
- **wLen** (*int*) – Number of bytes to receive
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range or the command is not supported or the timeout value is set to EC_NOWAIT
- EC_E_SLAVE_NOT_PRESENT if the slave is not present
- EC_E_NOTFOUND if no slave matching bFixedAddressing / wSlaveAddress can be found
- EC_E_TIMEOUT if dwTimeout elapsed during the API call
- EC_E_BUSY if another transfer request is already pending or the master or the corresponding slave is currently changing its operational state
- EC_E_NOTREADY if the working counter was not set when sending the command (slave may not be connected or did not respond)
- EC_E_INVALIDSIZE if the size of the complete command does not fit into a single Ethernet frame. The maximum amount of data to transfer must not exceed 1486 bytes.

Return type

[ECError](#)

ReadSlaveRegisterReq (

dwClientId: int,
dwTferId: int,
bFixedAddressing: bool,
wSlaveAddress: int,
wRegisterOffset: int,
pbyData: List[int],
wLen: int

) → [ECError](#)

Requests data read transfer from the ESC memory of a specified slave and returns immediately.

A notification EC_NOTIFY_SLAVE_REGISTER_TRANSFER is given on completion. This function may be called from within the JobTask's context.

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **dwTferId** (*int*) – Transfer ID. The application can set this ID to identify the transfer. It will be passed back to the application within EC_T_SLAVeregISTER_TRANSFER_NOTIFY_DESC.
- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **wRegisterOffset** (*int*) – Register offset, e.g. use 0x0130 to read the AL Status register
- **pbyData** (*List[int]*) – Buffer receiving transferred data
- **wLen** (*int*) – Number of bytes to receive

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARG if dwInstanceID is out of range or the command is not supported or the timeout value is set to EC_NOWAIT
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching bFixedAddressing / wSlaveAddress can be found
- EC_E_INVALIDSIZE if the size of the complete command does not fit into a single Ethernet frame. The maximum amount of data to transfer must not exceed 1486 bytes.

Return type

ECErr

static ReadValueFromAddress (

address: object,
bitOffset: int,
bitLength: int,
type_: DN_EC_T_DEFTYPE,
out_value: object

) → *ECErr*
 Read value from address

Parameters

- **address** (*object*) – Address
- **bitOffset** (*int*) – Bit offset
- **bitLength** (*int*) – Bit length
- **type** – Data type
- **out_value** (*object*) – Value
- **type_** (*DN_EC_T_DEFTYPE*) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

```

static ReadValueFromBytes (
    bytes_: List[int],
    bitOffset: int,
    bitLength: int,
    type_: DN_EC_T_DEFTYPE,
    out_value: object
) → ECErr
    Read value from bytes

```

Parameters

- **bytes** – Bytes
- **bitOffset** (*int*) – Bit offset
- **bitLength** (*int*) – Bit length
- **type** – Data type
- **out_value** (*object*) – Value
- **bytes_** (*List[int]*) –
- **type_** (*DN_EC_T_DEFTYPE*) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

```

RegisterClient (out_pRegRes: DN_EC_T_REGISTERRESULTS) → ECErr

```

Register a client with the EtherCAT Master

Parameters

out_pRegRes (*DN_EC_T_REGISTERRESULTS*) – out Registration results, a pointer to a structure of type REGISTERRESULTS

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

```

ReleaseAllProcessDataBits (dwClientId: int, dwTimeout: int) → ECErr

```

Release all previously forced process data for a dedicated client.

- Forced output: Value set by application become valid again. Because forced process data bits are written directly into the process output image, the application has to update the process image with the required value, otherwise the forced value is still valid.
- Forced input: Value read from the slaves become valid again.

This function may not be called from within the JobTask's context.

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **dwTimeout** (*int*) – Timeout [ms]. The timeout value must not be set to EC_NOWAIT.

Returns

EC_E_NOERROR or error code

Return type

ECErr

ReleaseProcessDataBits (

dwClientId: int,
bOutputData: bool,
dwBitOffsetPd: int,
wBitLength: int,
dwTimeout: int

) → *ECError*

Release previously forced process data.

- Forced output: Value set by application become valid again. Because forced process data bits are written directly into the process output image, the application has to update the process image with the required value, otherwise the forced value is still valid.
- Forced input: Value read from the slaves become valid again.

This function may not be called from within the JobTask's context.

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **bOutputData** (*bool*) – EC_TRUE: write output data, EC_FALSE: write input data
- **dwBitOffsetPd** (*int*) – Bit offset in Process data image
- **wBitLength** (*int*) – Number of bits that shall be written to the process image.
- **dwTimeout** (*int*) – Timeout [ms]. The timeout value must not be set to EC_NOWAIT.

Returns

EC_E_NOERROR or error code

Return type

ECError

ReloadSlaveEEPROM (bFixedAddressing: bool, wSlaveAddress: int, dwTimeout: int) → ECError

Causes a slave to reload its EEPROM values to ESC registers.

Alias address at 0x12 is not reloaded through this command, this is prevented by the slave hardware. The slave controller must be reset to reload the alias address. This function may not be called from within the JobTask's context.

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time. The timeout value must not be set to EC_NOWAIT.

Returns

EC_E_NOERROR or error code

Return type

ECError

ReloadSlaveEEPROMReq (

dwClientId: int,
dwTferId: int,
bFixedAddressing: bool,
wSlaveAddress: int,
dwTimeout: int

) → *ECError*

Request a slave to reload its EEPROM values to ESC registers, and returns immediately.

Alias address at 0x12 is not reloaded through this command, this is prevented by the slave hardware. The slave controller must be reset to reload the alias address. An `EC_NOTIFY_EEPROM_OPERATION` is given on completion or timeout. This function may be called from within the JobTask's context.

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **dwTferId** (*int*) – Transfer ID. The application can set this ID to identify the transfer. It will be passed back to the application within `EC_T_EEPROM_OPERATION_NOTIFY_DESC`.
- **bFixedAddressing** (*bool*) – `EC_TRUE`: use station address, `EC_FALSE`: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **dwTimeout** (*int*) – Timeout [ms]

Returns

`EC_E_NOERROR` or error code

Return type

[ECErrors](#)

RescueScan (*dwTimeout: int*) → [ECErrors](#)

Recovers the bus from permanent frame loss situations

Scans all connected slaves. Closes and opens ports on the network to rule out slaves which permanently discard frames. The Master notifies every slave port which permanently discards frames with `EC_NOTIFY_FRAMELOSS_AFTER_SLAVE`. Due to port opening and closing the scanning time is increased about 2 seconds per slave. The Master will not automatically re-open this port. The application can force to open the port again. This function may not be called from within the JobTask's context.

Parameters

dwTimeout (*int*) – Timeout [ms]

Returns

`EC_E_NOERROR` or error code

Return type

[ECErrors](#)

ResetSlaveController (

bFixedAddressing: bool,

wSlaveAddress: int,

dwTimeout: int

) → [ECErrors](#)

Reset EtherCAT slave controller (ESC)

A special sequence of three independent and consecutive frames/commands is sent to the slave (reset register `ECAT 0x0040` or `PDI 0x0041`), after which the slave resets. If that fails, the reset sequence is repeated until it succeeds or the timeout expires. The ESC must support resetting and the slave state should be `INIT` when calling this function. The number of acyclic frames per cycle `EC_T_INIT_MASTER_PARAMS.dwMaxAcycFramesPerCycle` must be at least 3, otherwise an error is returned. This function may not be called from within the JobTask's context.

Parameters

- **bFixedAddressing** (*bool*) – `EC_TRUE`: use station address, `EC_FALSE`: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing

- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time. The timeout value must not be set to EC_NOWAIT.

Returns

- EC_E_NOERROR or error code
- EC_E_NOTSUPPORTED if EC_T_INIT_MASTER_PARMS.dwMaxAcycFramesPerCycle is less than 3
- EC_E_SLAVE_NOT_PRESENT if the slave is not present
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive

Return type

ECErr

RestartScanBus (*dwTimeout: int, bReadRevisionNo: int, bReadSerialNo: int*) → *ECErr*

Trigger Bus Scan

Parameters

- **dwTimeout** (*int*) – Time out of bus scan
- **bReadRevisionNo** (*int*) – Read revision number
- **bReadSerialNo** (*int*) – Read serial number

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

ScanBus (*dwTimeout: int*) → *ECErr*

Scans all connected slaves.

Scans all connected slaves connected to EC-Master. If a configuration has been loaded, a validation between the configuration and the connected slaves is done. This function should not be called from within the JobTask's context.

Parameters

dwTimeout (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range
- EC_E_LINK_DISCONNECTED if link is disconnected
- EC_E_TIMEOUT if dwTimeout elapsed during the API call
- EC_E_BUSCONFIG_MISMATCH if the slaves found do not match the configured ones
- EC_E_LINE_CROSSED if a line crossed (cabling wrong) condition has been detected
- EC_E_REDLINEBREAK if cable redundancy is configured and a line break condition has been detected
- EC_E_JUNCTION_RED_LINE_BREAK if junction redundancy is configured and a line break condition has been detected
- EC_E_MAX_BUS_SLAVES_EXCEEDED if the amount of slaves found exceeds EC_T_INIT_MASTER_PARMS.dwMaxBusSlaves

- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if the ADS server is running

Return type

ESError

SdoUploadMasterOd (*wObIndex: int, dwTimeout: int, out_pobjMasterOd: object*) → ESError

SdoUpload from Master Object Dictionary

Parameters

- **wObIndex** (*int*) – Object index
- **dwTimeout** (*int*) – Timeout in milliseconds
- **pobjMasterOd** – Read object (e.g. DN_EC_T_OBJ3XXX, EC_T_OBJ8XXX, EC_T_OBJ9XXX ...)
- **out_pobjMasterOd** (*object*) –

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ESError

SelfTestScan (*pParms: DN_EC_T_SELFTESTSCAN_PARMS*) → ESError

Self test scan

Send a burst of numerous frames and analyze the slave connections. After deactivating the job task, frames will be sent as fast as the LinkLayer can send them. The size of the frames increases and decreases between the defined limits. Dependent on the parameters the BadConnectionsDetect API will analyze the slave connections.

Parameters**pParms** (*DN_EC_T_SELFTESTSCAN_PARMS*) – Self-test scan parameters**Returns**

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range or pParms is NULL or contains values out of range
- EC_E_BAD_CONNECTION if bad connection was detected
- EC_E_FRAME_LOST if frame(s) lost during self-test
- EC_E_NOTSUPPORTED if not in polling mode
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive

Return type

ESError

SetAllSlavesMustReachState (*bAllSlavesMustReachState: bool*) → ESError

Sets flag that all slaves must reach the requested master state

Parameters**bAllSlavesMustReachState** (*bool*) – True: All slaves must reach the requested master state**Returns**

EC_E_NOERROR on success, otherwise an error code.

Return type*ECError***SetBusCnfReadProp** (*eEscSiiReg*: `DN_ESC_SII_REG`, *dwTimeout*: *int*) → *ECError*

Sets read property for bus configuration

Parameters

- **eEscSiiReg** (`DN_ESC_SII_REG`) – SII register
- **dwTimeout** (*int*) – Time out of bus scan

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECError***SetCycFrameRxCallback** (*pfnCallback*: *object*) → *ECError*

Set cyclic frame RX callback function

This function will be called after the cyclic frame is received, if there is more than one cyclic frame after the last frame. The application has to assure that these functions will not block.

Parameters**pfnCallback** (*object*) – Callback function**Returns**

EC_E_NOERROR or error code

Return type*ECError***SetLicenseKey** (*szLicenseKey*: *str*) → *ECError*

Sets the license key for the protected version of EC-Master.

Must be called after initialization and before configuration. This function may not be called if a non protected version is used.

Parameters**szLicenseKey** (*str*) – License key as zero terminated string with 26, 53 or 56 characters**Returns**

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range
- EC_E_INVALIDSIZE if the format of the license key is wrong. The correct length is 26, 53 or 56 characters.
- EC_E_LICENSE_MISSING if the license key doesn't match the MAC Address

Return type*ECError***SetLogParms** (*pLogParms*: `DN_EC_T_LOG_PARMS`) → *ECError*

Sets log parameters. Used to change the parameters provided by emInitMaster().

EC_MONITOR

Sets log parameters. Used to change the parameters provided by emonInitMonitor().

EC_SIMULATOR

Sets log parameters. Used to change the parameters provided by esInitSimulator().

Parameters**pLogParms** (`DN_EC_T_LOG_PARMS`) – New Log parameters**Return type**`ECErr`

Returns:

SetMasterParms (*pParms*: `DN_EC_T_INIT_MASTER_PARMS`) → `ECErr`

Change Master initialization parameters.

Currently the following parameters cannot be changed: - `EC_T_INIT_MASTER_PARMS.pOsParms`
 - `EC_T_INIT_MASTER_PARMS.pLinkParms` - `EC_T_INIT_MASTER_PARMS.pLinkParmsRed` -
`EC_T_INIT_MASTER_PARMS.dwMaxBusSlaves` - `EC_T_INIT_MASTER_PARMS.dwMaxAcycFramesQueued`
 - `EC_T_INIT_MASTER_PARMS.dwAdditionalEoEEndpoints`

Parameters**pParms** (`DN_EC_T_INIT_MASTER_PARMS`) – New Master parameters**Returns**

- `EC_E_NOERROR` on success
- `EC_E_INVALIDSTATE` if master isn't initialized

Return type`ECErr`**SetMasterRedStateReq** (*bActive*: `bool`) → `ECErr`

Requests Master Redundancy State ACTIVE / INACTIVE.

Parameters**bActive** (`bool`) –**Returns**

- `EC_E_NOERROR` if successful
- `EC_E_INVALIDSTATE` if EtherCAT stack isn't initialized or `MasterRedParms.bEnabled = EC_FALSE`
- `EC_E_INVALIDPARG` if `dwInstanceID` is out of range
- `EC_E_NOTSUPPORTED` if EC-Master stack does not include Master Redundancy support

Return type`ECErr`**SetMasterState** (*dwTimeout*: `int`, *eReqState*: `DN_EC_T_STATE`) → `ECErr`

Set the master (and all slaves) into the requested EtherCAT state.

If the function is called with `EC_NOWAIT`, the client may wait for reaching the requested state using the notification callback (`EC_NOTIFY_STATECHANGED`). By default the Master will just change to a higher state if all slaves have reached the requested state. It may happen that some slaves are in higher state at network than the Master, e.g.: - Master and all slaves are in `PREOP` - Application requests `SAFEOP` - Master starts transition for all slaves - Some slaves changed to `SAFEOP`, but some fail and therefore stay in `PREOP` - Master state stays in `PREOP`, function returns with error

The application can request `SAFEOP` again to re-request the state of previously failed slaves. Transition to a lower state: The master changes to a lower state even if one slave is not able to follow. This function may not be called from within the `JobTask`'s context with `dwTimeout` other than `EC_NOWAIT`.

Parameters

- **dwTimeout** (*int*) – Timeout [ms]. 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.
- **eReqState** (DN_EC_T_STATE) – Requested System state

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range
- EC_E_TIMEOUT if dwTimeout elapsed during the API call
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if the ADS server is running

Return type

`ECError`

SetMasterStateReq (*dwTimeout: int, eReqState: DN_EC_T_STATE*) → *ECError*

Request to set the master (and all slaves) into the requested EtherCAT state and return immediately.

The Master by default will just change to a higher state, if all slaves have reached the requested state. It may happen that some slaves are in higher state at network than Master, e.g.: - Master and all slaves are in PREOP - Application requests SAFEOP - Master starts transition for all slaves - Some slaves changed to SAFEOP, but some fail and therefore stay in PREOP - Master state stays in PREOP, function returns with error

The application can request SAFEOP again to re-request state of previously failed slaves. Transition to lower state: The master changes to lower state, even if one slave is not able to follow.

See also EC_NOTIFY_STATECHANGED.

Parameters

- **dwTimeout** (*int*) – Timeout [ms]. 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.
- **eReqState** (DN_EC_T_STATE) – Requested System state

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if ADS server is running

Return type

`ECError`

```

SetProcessData (
    bOutputData: bool,
    dwOffset: int,
    pbyData: List[int],
    dwLength: int,
    dwTimeout: int

```

) → *ECError*
Write Process data synchronized.

This function may not be called from within the JobTask's context.

Parameters

- **bOutputData** (*bool*) – EC_TRUE: write output data, EC_FALSE: write input data
- **dwOffset** (*int*) – Byte offset in Process data to write to
- **pbyData** (*List[int]*) – Buffer containing transferred data
- **dwLength** (*int*) –
- **dwTimeout** (*int*) – Timeout [ms]

Returns

EC_E_NOERROR or error code

Return type

ECError

```

SetProcessDataBits (
    bOutputData: bool,
    dwBitOffsetPd: int,
    pbyDataSrc: List[int],
    dwBitLengthSrc: int,
    dwTimeout: int

```

) → *ECError*
Writes a specific number of bits from a given buffer to the process image with a bit offset (synchronized).

This function may not be called from within the JobTask's context.

Parameters

- **bOutputData** (*bool*) – EC_TRUE: write output data, EC_FALSE: write input data
- **dwBitOffsetPd** (*int*) – Bit offset in Process data image
- **pbyDataSrc** (*List[int]*) –
- **dwBitLengthSrc** (*int*) –
- **dwTimeout** (*int*) – Timeout [ms]. The timeout value must not be set to EC_NOWAIT.

Returns

EC_E_NOERROR or error code

Return type

ECError

```

SetSlaveDisabled (bFixedAddressing: bool, wSlaveAddress: int, bDisabled: bool) → ECError
    Enable or disable a specific slave

```

The EtherCAT state of disabled slaves cannot be set higher than PREOP. If the state is higher than PREOP at the time this function is called the state will be automatically changed to PREOP. The information about the last requested state is lost and is set to PREOP too.

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **bDisabled** (*bool*) – EC_TRUE: Disable slave, EC_FALSE: Enable slave

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

SetSlaveDisconnected (

bFixedAddressing: bool,

wSlaveAddress: int,

bDisconnected: bool

) → [ESError](#)

Mark specific slave for connection or disconnection

The EtherCAT state of disconnected slaves cannot be set higher than INIT. If the state is higher than INIT at the time this function is called, the state will be automatically changed to INIT. The information about the last requested state is lost and is set to INIT too.

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **bDisconnected** (*bool*) – EC_TRUE: Mark slave for disconnection, EC_FALSE: Mark slave for (re-)connection

Returns

EC_E_NOERROR or error code

Return type

[ESError](#)

SetSlavePortState (

dwSlaveId: int,

wPort: int,

bClose: bool,

bForce: bool,

dwTimeout: int

) → [ESError](#)

Open or close slave port

This function allows to open or close a specific slave port in different ways. It can also be used to re-open ports closed by a rescue scan.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **wPort** (*int*) – Port to open or close. Can be ESC_PORT_A, ESC_PORT_B, ESC_PORT_C, ESC_PORT_D.
- **bClose** (*bool*) – EC_TRUE: close port, EC_FALSE: open port
- **bForce** (*bool*) – EC_TRUE: port will be closed or open, EC_FALSE: port will be set in AutoClose mode
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive

Return type

[ESError](#)

SetSlavePortStateReq (

dwClientId: int,
dwTferId: int,
dwSlaveId: int,
wPort: int,
bClose: bool,
bForce: bool,
dwTimeout: int

) → [ESError](#)

Requests Open or close slave port operation and returns immediately.

An EC_T_PORT_OPERATION_NOTIFY_DESC is given on completion. This function can be called to re-open ports closed by a rescue scan.

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **dwTferId** (*int*) – Transfer ID. The application can set this ID to identify the transfer. It will be passed back to the application within EC_T_PORT_OPERATION_NOTIFY_DESC.
- **dwSlaveId** (*int*) – Slave ID
- **wPort** (*int*) – Port to open or close. Can be ESC_PORT_A, ESC_PORT_B, ESC_PORT_C, ESC_PORT_D.
- **bClose** (*bool*) – EC_TRUE: close port, EC_FALSE: open port
- **bForce** (*bool*) – EC_TRUE: port will be closed or open, EC_FALSE: port will be set in AutoClose mode
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found

Return type

[ESError](#)

SetSlaveState (*dwSlaveId: int, wDeviceState: int, dwTimeout: int*) → *ECError*

Set a specified slave into the requested EtherCAT state.

The requested state shall not be higher than the overall operational state. `DEVICE_STATE_BOOTSTRAP` can only be requested if the slave's state is `INIT`. This function may not be called from within the `JobTask`'s context.

If the function is called with `EC_NOWAIT`, the client may wait for reaching the requested state using the notification callback (`EC_NOTIFY_SLAVE_STATECHANGED`).

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **wDeviceState** (*int*) – Requested device state. See `DEVICE_STATES`
- **dwTimeout** (*int*) – Timeout [ms]. 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.

Returns

- `EC_E_NOERROR` if successful
- `EC_E_INVALIDSTATE` if EtherCAT stack isn't initialized or denies the requested state, see comments below
- `EC_E_INVALIDPARAM` if `dwInstanceID` is out of range or `BOOTSTRAP` was requested for a slave that does not support it
- `EC_E_NOTFOUND` if no slave matching `dwSlaveId` can be found
- `EC_E_TIMEOUT` if `dwTimeout` elapsed during the API call
- `EC_E_BUSY` if the EtherCAT stack cannot execute the request at this time, the function has to be called at a later time
- `EC_E_NOTREADY` if the working counter was not set when requesting the slave's state (slave may not be connected or did not respond)
- `EC_E_MASTER_RED_STATE_INACTIVE` if Master Redundancy is configured and master is inactive

Return type

ECError

SetSlaveStateReq (*dwSlaveId: int, wDeviceState: int, dwTimeout: int*) → *ECError*

Request to set a specified slave into the requested EtherCAT state and return immediately.

The requested state shall not be higher than the overall operational state. `DEVICE_STATE_BOOTSTRAP` can only be requested if the slave's state is `INIT`.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **wDeviceState** (*int*) – Requested device state. See `DEVICE_STATES`
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- `EC_E_NOERROR` if successful
- `EC_E_INVALIDSTATE` if EtherCAT stack isn't initialized or denies the requested state, see comments below

- **EC_E_INVALIDPARM** if dwInstanceID is out of range or BOOTSTRAP was requested for a slave that does not support it
- **EC_E_NOTFOUND** if no slave matching dwSlaveId can be found
- **EC_E_BUSY** if the EtherCAT stack cannot execute the request at this time, the function has to be called at a later time
- **EC_E_MASTER_RED_STATE_INACTIVE** if Master Redundancy is configured and master is inactive

Return type

`ECErr`

```
SetSlavesDisabled (
    bFixedAddressing: bool,
    wSlaveAddress: int,
    eSlaveSelection: DN_EC_T_SLAVE_SELECTION,
    bDisabled: bool
```

) → `ECErr`
Enable or disable a specific group of slaves

The EtherCAT state of disabled slaves cannot be set higher than PREOP. If the state is higher than PREOP at the time this function is called, the state will be automatically change to PREOP. The information about the last requested state is lost and is set to PREOP too.

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **eSlaveSelection** (*DN_EC_T_SLAVE_SELECTION*) – Slave selection criteria for following slaves
- **bDisabled** (*bool*) – EC_TRUE: Disable slaves, EC_FALSE: Enable slaves

Returns

EC_E_NOERROR or error code

Return type

`ECErr`

```
SetSlavesDisconnected (
    bFixedAddressing: bool,
    wSlaveAddress: int,
    eSlaveSelection: DN_EC_T_SLAVE_SELECTION,
    bDisconnected: bool
```

) → `ECErr`
Mark a specific group of slaves for connection or disconnection

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **eSlaveSelection** (*DN_EC_T_SLAVE_SELECTION*) – Slave selection criteria
- **bDisconnected** (*bool*) – EC_TRUE: mark slaves for disconnection, EC_FALSE: mark slaves for connection

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range
- EC_E_NOTFOUND if no slave matching bFixedAddressing / wSlaveAddress can be found

Return type

ELError

SlaveParallelMbxTfers (*dwSlaveId: int*) → *ELError*

Re-enable the parallel mailbox transfers to the specified slave

Allows parallel (overlapped) usage of more than one protocol (CoE, EoE, FoE, etc.). By default parallel mailbox transfers are enabled.

Parameters**dwSlaveId** (*int*) – Slave ID**Returns**

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if slave does not support mailbox transfers

Return type

ELError

SlaveProdCodeText (*dwVendorId: int, dwProductCode: int*) → str

Gets the text of slave product code

Parameters

- **dwVendorId** (*int*) – Vendor ID
- **dwProductCode** (*int*) – Product code

Returns

Text of slave product code

Return type

str

SlaveSerializeMbxTfers (*dwSlaveId: int*) → *ELError*

Serializes all mailbox transfers to the specified slave

The parallel (overlapped) usage of more than one protocol (CoE, EoE, FoE, etc.) will be disabled. By default parallel mailbox transfers are enabled.

Parameters**dwSlaveId** (*int*) – Slave ID**Returns**

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found

- `EC_E_NO_MBX_SUPPORT` if slave does not support mailbox transfers

Return type`ECErr`**SlaveVendorText** (*dwVendorId: int*) → str

Gets the text of slave vendor

Parameters**dwVendorId** (*int*) – Vendor ID**Returns**

Text of slave vendor

Return type

str

SoeAbortProcCmd (

dwSlaveId: int,
byDriveNo: int,
pbyElementFlags: List[int],
wIDN: int,
dwTimeout: int

) → `ECErr`

Abort SSC Procedure Command sequence.

A Procedure Command takes up some time. After a procedure command has started, the slave generates a normal SSC Write Response. The end of a procedure command is indicated by the Notify SSC Command Execution Service.

This function may not be called from within the JobTask's context.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **byDriveNo** (*int*) – Drive number to address inside slave device
- **pbyElementFlags** (*List[int]*) – [in/out] SoE ElementFlags. Flag indicating elements to address.
- **wIDN** (*int*) – IDN of the object to address
- **dwTimeout** (*int*) – Timeout [ms]

Returns`EC_E_NOERROR` or error code**Return type**`ECErr`**SoeRead** (

dwSlaveId: int,
byDriveNo: int,
pbyElementFlags: List[int],
wIDN: int,
pbyData: List[int],
dwDataLen: int,
out_pdwOutDataLen: int,
dwTimeout: int

) → `ECErr`

Execute an SoE SCC Read service which uploads data from an EtherCAT SoE slave device.

The received data can consist of several fragments. The reserved data buffer (*pbyData*) must have space for all received data segments and the selected Timeout should be increasing with the count of fragments.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **byDriveNo** (*int*) – Drive number to address inside slave device
- **pbyElementFlags** (*List[int]*) – [in/out] SoE ElementFlags. Flag indicating elements to address.
- **wIDN** (*int*) – IDN of the object to address
- **pbyData** (*List[int]*) – Buffer receiving transferred data
- **dwDataLen** (*int*) – Buffer length [byte]
- **pdwOutDataLen** – Length of received data [byte]
- **dwTimeout** (*int*) – Timeout [ms]
- **out_pdwOutDataLen** (*int*) –

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer, or dwTimeout is EC_NOWAIT
- EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if slave has no mailbox support
- EC_E_INVALID_SLAVE_STATE if slave is in an invalid state for mailbox transfer
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if ADS server is running
- EC_ERROR_CODES_SOE “SoE error code”

Return type

`ECErrror`

```
SoeReadReq (
    pMbxTfer: object,
    dwSlaveId: int,
    byDriveNo: int,
    pbyElementFlags: List[int],
    wIDN: int,
    dwTimeout: int
```

) → `ECErrror`

Requests an SoE SSC Read and returns immediately.

This function may be called from within the JobTask's context.

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object
- **dwSlaveId** (*int*) – Slave ID
- **byDriveNo** (*int*) – Drive number to address inside slave device

- **pbyElementFlags** (*List[int]*) – [in/out] SoE ElementFlags. Flag indicating elements to address.
- **wIDN** (*int*) – IDN of the object to address
- **dwTimeout** (*int*) – Timeout [ms]. Must not be set to EC_NOWAIT.

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer, or dwTimeout is EC_NOWAIT
- EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if slave has no mailbox support
- EC_E_INVALID_SLAVE_STATE if slave is in an invalid state for mailbox transfer
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if ADS server is running
- EC_ERROR_CODES_SOE “SoE error code”

Return type

[ECError](#)

```
SoeWrite (
    dwSlaveId: int,
    byDriveNo: int,
    pbyElementFlags: List[int],
    wIDN: int,
    pbyData: List[int],
    dwDataLen: int,
    dwTimeout: int
```

) → [ECError](#)

Execute an SoE SSC Write service which downloads data to an EtherCAT slave device.

The function returns after timeout expired or download is completed successfully (Write response is received). It can also perform an SoE SSC Procedure Command. After a procedure command has started, the slave generates a normal SSC Write Response, and the function returns. If the data to be sent with the write service exceeds the mailbox size, the data will be sent fragmented. The fragmented write operation consists of several Write SSC Fragment Services. Therefore the selected Timeout should be increasing with the count of fragments. This function may not be called from within the JobTask's context.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **byDriveNo** (*int*) – Drive number to address inside slave device
- **pbyElementFlags** (*List[int]*) – [in/out] SoE ElementFlags. Flag indicating elements to address.
- **wIDN** (*int*) – IDN of the object to address
- **pbyData** (*List[int]*) – Buffer containing transferred data

- **dwDataLen** (*int*) – Buffer length [byte]
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer, or dwTimeout is EC_NOWAIT
- EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if slave has no mailbox support
- EC_E_INVALID_SLAVE_STATE if slave is in an invalid state for mailbox transfer
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if ADS server is running
- EC_ERROR_CODES_SOE “SoE error code”

Return type

[ESError](#)

```
SoeWriteReq (
    pMbxTfer: object,
    dwSlaveId: int,
    byDriveNo: int,
    pbyElementFlags: List[int],
    wIDN: int,
    dwTimeout: int
```

) → [ESError](#)

Requests an SoE SSC Write and returns immediately.

This function may be called from within the JobTask's context.

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object
- **dwSlaveId** (*int*) – Slave ID
- **byDriveNo** (*int*) – Drive number to address inside slave device
- **pbyElementFlags** (*List[int]*) – [in/out] SoE ElementFlags. Flag indicating elements to address.
- **wIDN** (*int*) – IDN of the object to address
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARM if dwInstanceID is out of range, the input pointer is EC_NULL or contains EC_NULL pointer, or dwTimeout is EC_NOWAIT

- EC_E_NOMEMORY if the mailbox protocol queue of the slave is full
- EC_E_SLAVE_NOT_PRESENT if slave not present
- EC_E_NOTFOUND if no slave matching dwSlaveId can be found
- EC_E_NO_MBX_SUPPORT if slave has no mailbox support
- EC_E_INVALID_SLAVE_STATE if slave is in an invalid state for mailbox transfer
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive
- EC_E_ADS_IS_RUNNING if ADS server is running
- EC_ERROR_CODES_SOE “SoE error code”

Return type

`ECErrror`

`TferSingleRawCmd` (

byCmd: int,
dwMemoryAddress: int,
pbyData: List[int],
wLen: int,
dwTimeout: int

) → `ECErrror`

Transfers a single raw EtherCAT command to one or multiple slaves and waits for the result.

Using this function it is possible to exchange arbitrary data between the master and the slaves. When the master receives the response to the queued frame it raises `EC_NOTIFY_RAWCMD_DONE` to all clients. This function blocks until the command is completely processed. In case of read commands the slave data will be written back into the given memory area. If a timeout occurs (e.g. due to a bad line quality) the corresponding frame will be sent again. The timeout value and retry counter can be set using the master configuration parameters `dwEcatCmdTimeout` and `dwEcatCmdMaxRetries`. The call will return in any case (without waiting for the number of retries specified in `dwEcatCmdMaxRetries`) if the time determined with the `dwTimeout` parameter elapsed. Caveat: Using auto increment addressing (APRD, APWR, APRW) may lead to unexpected results in case the selected slave does not increment the working counter. In such cases the EtherCAT command would be handled by the slave directly behind the selected one. This function may not be called from within the `JobTask`'s context.

Parameters

- **`byCmd`** (*int*) – EtherCAT command type. See `EC_CMD_TYPE`
- **`dwMemoryAddress`** (*int*) – Slave memory address, depending on the command to be sent this is either a physical or a logical address
- **`pbyData`** (*List[int]*) – [in, out] Buffer containing or receiving transferred data
- **`wLen`** (*int*) – Number of bytes to transfer
- **`dwTimeout`** (*int*) – Timeout [ms]

Returns

- `EC_E_NOERROR` if successful
- `EC_E_INVALIDSTATE` if EtherCAT stack isn't initialized
- `EC_E_INVALIDPARAM` if `dwInstanceID` is out of range or the command is not supported or the timeout value is set to `EC_NOWAIT`
- `EC_E_TIMEOUT` if `dwTimeout` elapsed during the API call
- `EC_E_BUSY` if another transfer request is already pending or the master or the corresponding slave is currently changing its operational state

- **EC_E_NOTREADY** if the working counter was not set when sending the command (slave may not be connected or did not respond)
- **EC_E_INVALIDSIZE** if the size of the complete command does not fit into a single Ethernet frame. The maximum amount of data to transfer must not exceed 1486 bytes.
- **EC_E_MASTER_RED_STATE_INACTIVE** if Master Redundancy is configured and master is inactive

Return type*ECErr***ThrottleNotification** (*tNotifyCode: DN_NotifyCode, dwTimeout: int*) → *ECErr*

Throttles notification

Parameters

- **tNotifyCode** (*DN_NotifyCode*) – Code of notification, which should be throttled
- **dwTimeout** (*int*) – 0 = Not throttled, > 0 = Throttle timeout in ms

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type*ECErr***ThrowDbgMsgEvent** (*type_: int, severity: int, msg: str, _pUserData: object*)

Throws the debug message events.

Parameters

- **type_** (*int*) –
- **severity** (*int*) –
- **msg** (*str*) –
- **_pUserData** (*object*) –

ThrowEcEvent (*dwCode: int, unmParms: object*) → *int*

Throws the EtherCAT-Notifications. The Type of the event depends on the notification code.

Parameters

- **dwCode** (*int*) –
- **unmParms** (*object*) –

Return type*int***ThrowPerfEvent** (*pszFktName: str, dwTime: int, _pUserData: object*)

Throws the performance monitoring events.

Parameters

- **pszFktName** (*str*) –
- **dwTime** (*int*) –
- **_pUserData** (*object*) –

ThrowRasCheckTokenEvent (
 _pvCheckTokenContext: object,
 pbyToken: object,
 dwTokenSize: int
) → int
 Throws the RAS check token events.

Parameters

- **_pvCheckTokenContext** (*object*) –
- **pbyToken** (*object*) –
- **dwTokenSize** (*int*) –

Return type
 int

ThrowRasEvent (*dwCode: int, unmParms: object*) → int
 Throws the RAS Events. The Type of the event depends on the notification code.

Parameters

- **dwCode** (*int*) –
- **unmParms** (*object*) –

Return type
 int

ThrowTranslateEvent (*_code: int, msg: str, _pUserData: object*)
 Throws the performance monitoring events.

Parameters

- **_code** (*int*) –
- **msg** (*str*) –
- **_pUserData** (*object*) –

TraceDataConfig (*wTraceDataSize: int*) → *ECError*
 Configures a trace data buffer and enables it for transmission

Must be called after initialization and before configuration.

If *wTraceDataSize* is too large, configuration will fail with return code #EC_E_XML_CYCCMDS_SIZEMISMATCH.

Parameters

wTraceDataSize (*int*) – Size of Trace Data in bytes

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range
- EC_E_NOTSUPPORTED if eCycFrameLayout_FIXED is configured

Return type
ECError

TraceDataGetInfo (*pTraceDataInfo*: DN_EC_T_TRACE_DATA_INFO) → *ECError*

Get information about the offset and size of trace data.

The trace data buffer is located in EC_T_TRACE_DATA_INFO.pbyData at the byte offset EC_T_TRACE_DATA_INFO.dwOffset.

Parameters

pTraceDataInfo (DN_EC_T_TRACE_DATA_INFO) – Information about trace data

Returns

EC_E_NOERROR or error code

Return type

ECError

UnregisterClient () → *ECError*

Unregister a client from the EtherCAT master

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

VoERead (

dwSlaveId: int,

pbyData: List[int],

dwDataLen: int,

out_pdwOutDataLen: int,

dwTimeout: int

) → *ECError*

Retrieves VoE mailbox, that was sent by an EtherCAT slave device.

If a VoE mailbox was already received, further received VoE mailboxes will be discarded as long as this function was not called. The received data includes the Mailbox header of type ETHERCAT_MBOX_HEADER followed by the VoE payload. This function blocks until the VoE data has been successfully received or an error has occurred.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **pbyData** (*List[int]*) – Buffer receiving transferred data
- **dwDataLen** (*int*) – Buffer length [byte]
- **pdwOutDataLen** – Length of received data [byte]
- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time. EC_NOWAIT returns the function immediately.
- **out_pdwOutDataLen** (*int*) –

Returns

- EC_E_NOERROR if the VoE slave has provided some VoE data
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized
- EC_E_INVALIDPARAM if dwInstanceID is out of range
- EC_E_TIMEOUT if dwTimeout elapsed during the API call
- EC_E_VOE_NO_MBX_RECEIVED if no VoE data was received
- EC_E_MASTER_RED_STATE_INACTIVE if Master Redundancy is configured and master is inactive

Return type

ESError

VoeWrite (*dwSlaveId: int, pbyData: List[int], dwDataLen: int, dwTimeout: int*) → *ESError*

Execute a VoE mailbox write to an EtherCAT slave device.

This function blocks until the VoE write has been successfully completed or an error has occurred.

Parameters

- **dwSlaveId** (*int*) – Slave ID
- **pbyData** (*List[int]*) – Buffer containing transferred data
- **dwDataLen** (*int*) – Buffer length [bytes]. The maximum data length including 6 bytes for the mailbox header is given at `EC_T_CFG_SLAVE_INFO.dwMbxOutSize`.
- **dwTimeout** (*int*) – Timeout [ms]. The function will block at most for this time.

Returns

EC_E_NOERROR or error code

Return type

ESError

VoeWriteReq (*pMbxTfer: object, dwSlaveId: int, dwTimeout: int*) → *ESError*

Initiates a VoE mailbox write to an EtherCAT slave device.

The amount of data bytes to write has to be stored in `EC_T_MBXTFER.dwDataLen`. The maximum data length including 6 bytes for the mailbox header is given at `EC_T_CFG_SLAVE_INFO.dwMbxOutSize`. A unique transfer ID must be written into `EC_T_MBXTFER.dwTferId`.

Parameters

- **pMbxTfer** (*object*) – Mailbox transfer object
- **dwSlaveId** (*int*) – Slave ID
- **dwTimeout** (*int*) – Timeout [ms], `EC_NOWAIT` returns the function immediately

Returns

EC_E_NOERROR or error code

Return type

ESError

WaitForTimingTaskEvent () → bool

Waits for Timing Task Event

Returns

Wait event was triggered

Return type

bool

static WritePdBitsToAddress (*address: object,**bitOffset: int,**bitLength: int,**pdBits: List[int]*

)

Writes process data bits to address

Parameters

- **address** (*object*) – Address

- **bitOffset** (*int*) – Bit offset
- **bitLength** (*int*) – Bit length
- **pdBits** (*List[int]*) – Process data bits

static WritePdBitsToBytes (*bytes_: List[int], bitOffset: int, bitLength: int, pdBits: List[int]*)
Writes process data bits to bytes

Parameters

- **bytes** – Bytes
- **bitOffset** (*int*) – Bit offset
- **bitLength** (*int*) – Bit length
- **pdBits** (*List[int]*) – Process data bits
- **bytes_** (*List[int]*) –

static WritePdBytesToAddress (*address: object, offset: int, pdBytes: List[int]*)
Writes process data bytes to address

Parameters

- **address** (*object*) – Address
- **offset** (*int*) – Offset
- **pdBytes** (*List[int]*) – Process data bytes

static WritePdBytesToBytes (*bytes_: List[int], offset: int, pdBytes: List[int]*)
Writes process data bytes to bytes

Parameters

- **bytes** – Bytes
- **offset** (*int*) – Offset
- **pdBytes** (*List[int]*) – Process data bytes
- **bytes_** (*List[int]*) –

WriteSlaveEEPROM (
bFixedAddressing: bool,
wSlaveAddress: int,
wEEPROMStartOffset: int,
pwWriteData: List[int],
dwWriteLen: int,
dwTimeout: int
) → *ECError*
 Write EEPROM data from slave

Parameters

- **bFixedAddressing** (*bool*) – True: use station addressing, False: use auto increment addressing
- **wSlaveAddress** (*int*) – Slave Address, station or auto increment address depending on bFixedAddressing
- **wEEPROMStartOffset** (*int*) – Address to start EEPROM Read from
- **pwWriteData** (*List[int]*) – Pointer to WORD array carrying the write data.
- **dwWriteLen** (*int*) – Size of Write Data WORD array (in WORDS)

- **dwTimeout** (*int*) – Timeout in milliseconds. The function will block at most for this time. The timeout value must not be set to EC_NOWAIT

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECErr

```
WriteSlaveEEPROMReq (
    dwClientId: int,
    dwTferId: int,
    bFixedAddressing: bool,
    wSlaveAddress: int,
    wEEPROMStartOffset: int,
    ref_pwWriteData: int,
    dwWriteLen: int,
    dwTimeout: int
) → ECErr
```

Requests an EEPROM data write operation from a slave and returns immediately.

The EEPROM's CRC is updated automatically. A reset of the slave controller is needed to reload the alias address in register 0x12. An EC_NOTIFY_EEPROM_OPERATION is given on completion or timeout. This function may be called from within the JobTask's context.

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **dwTferId** (*int*) – Transfer ID. The application can set this ID to identify the transfer. It will be passed back to the application within EC_T_EEPROM_OPERATION_NOTIFY_DESC.
- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **wEEPROMStartOffset** (*int*) – Word address to start EEPROM Write from
- **pwWriteData** – Pointer to WORD array carrying the write data, must be valid until operation complete
- **dwWriteLen** (*int*) – Size of Write Data WORD array (in WORDS)
- **dwTimeout** (*int*) – Timeout [ms]
- **ref_pwWriteData** (*int*) –

Returns

EC_E_NOERROR or error code

Return type

ECErr

```
WriteSlaveEEPROMWithReq (
    dwClientId: int,
    dwTferId: int,
    bFixedAddressing: bool,
    wSlaveAddress: int,
    wEEPROMStartOffset: int,
    pwWriteData: List[int],
    dwWriteLen: int,
    dwTimeout: int
) → ECErr
```

Write EEPROM data from slave (REQ)

Parameters

- **dwClientId** (*int*) – Client ID returned by RegisterClient (0 if all registered clients shall be notified)
- **dwTferId** (*int*) – Transfer ID. The application can set this ID to identify the transfer. It will be passed back to the application within EC_T_SLAVEREGISTER_TRANSFER_NOTIFY_DESC.
- **bFixedAddressing** (*bool*) – True: use station addressing, False: use auto increment addressing
- **wSlaveAddress** (*int*) – Slave Address, station or auto increment address depending on bFixedAddressing
- **wEEPROMStartOffset** (*int*) – Address to start EEPROM Read from
- **pwWriteData** (*List[int]*) – Pointer to WORD array carrying the write data.
- **dwWriteLen** (*int*) – Sizeof Write Data WORD array (in WORDS)
- **dwTimeout** (*int*) – Timeout in milliseconds. The function will block at most for this time. The timeout value must not be set to EC_NOWAIT

Returns

EC_E_NOERROR on success, otherwise an error code.

Return type

ECError

```
WriteSlaveRegister (  
    bFixedAddressing: bool,  
    wSlaveAddress: int,  
    wRegisterOffset: int,  
    pbData: List[int],  
    wLen: int,  
    dwTimeout: int
```

) → *ECError*

Writes data into the ESC memory of a specified slave.

This function may not be called from within the JobTask's context

Changing contents of ESC registers may lead to unpredictable behavior of the slaves and/or the master.

Parameters

- **bFixedAddressing** (*bool*) – EC_TRUE: use station address, EC_FALSE: use AutoInc address
- **wSlaveAddress** (*int*) – Slave address according bFixedAddressing
- **wRegisterOffset** (*int*) – Register offset. E.g. use 0x0120 to write to the AL Control register.
- **pbData** (*List[int]*) – Buffer containing transferred data
- **wLen** (*int*) – Number of bytes to send
- **dwTimeout** (*int*) – Timeout [ms]

Returns

- EC_E_NOERROR if successful
- EC_E_INVALIDSTATE if EtherCAT stack isn't initialized

- `EC_E_INVALIDPARM` if `dwInstanceId` is out of range or the command is not supported or the timeout value is set to `EC_NOWAIT`
- `EC_E_SLAVE_NOT_PRESENT` if slave not present
- `EC_E_NOTFOUND` if no slave matching `bFixedAddressing` / `wSlaveAddress` can be found
- `EC_E_TIMEOUT` if `dwTimeout` elapsed during the API call
- `EC_E_BUSY` if another transfer request is already pending or the master or the corresponding slave is currently changing its operational state
- `EC_E_NOTREADY` if the working counter was not set when sending the command (slave may not be connected or did not respond)
- `EC_E_INVALIDSIZE` if the size of the complete command does not fit into a single Ethernet frame. The maximum amount of data to transfer must not exceed 1486 bytes.

Return type

`ECError`

`WriteSlaveRegisterReq` (

dwClientId: int,
dwTferId: int,
bFixedAddressing: bool,
wSlaveAddress: int,
wRegisterOffset: int,
pbData: List[int],
wLen: int

) → `ECError`

Requests a data write transfer into the ESC memory of a specified slave and returns immediately.

A notification `EC_NOTIFY_SLAVE_REGISTER_TRANSFER` is given on completion. This function may be called from within the `JobTask`'s context.

Changing contents of ESC registers may lead to unpredictable behavior of the slaves and/or the master.

Parameters

- `dwClientId` (*int*) – Client ID returned by `RegisterClient` (0 if all registered clients shall be notified)
- `dwTferId` (*int*) – Transfer ID. The application can set this ID to identify the transfer. It will be passed back to the application within `EC_T_SLAVEREGISTER_TRANSFER_NOTIFY_DESC`.
- `bFixedAddressing` (*bool*) – `EC_TRUE`: use station address, `EC_FALSE`: use `AutoInc` address
- `wSlaveAddress` (*int*) – Slave address according `bFixedAddressing`
- `wRegisterOffset` (*int*) – Register offset. E.g. use 0x0120 to write to the AL Control register.
- `pbData` (*List[int]*) – Buffer containing transferred data
- `wLen` (*int*) – Number of bytes to send

Returns

- `EC_E_NOERROR` if successful
- `EC_E_INVALIDSTATE` if EtherCAT stack isn't initialized
- `EC_E_INVALIDPARM` if `dwInstanceId` is out of range or the command is not supported or the timeout value is set to `EC_NOWAIT`

- `EC_E_SLAVE_NOT_PRESENT` if slave not present
- `EC_E_NOTFOUND` if no slave matching `bFixedAddressing` / `wSlaveAddress` can be found
- `EC_E_INVALIDSIZE` if the size of the complete command does not fit into a single Ethernet frame. The maximum amount of data to transfer must not exceed 1486 bytes.

Return type`ECErrror`

```
static WriteValueToAddress (
    address: object,
    bitOffset: int,
    bitLength: int,
    type_: DN_EC_T_DEFTYPE,
    value: object
)
Write value to address
```

Parameters

- **address** (*object*) – Address
- **bitOffset** (*int*) – Bit offset
- **bitLength** (*int*) – Bit length
- **type** – Data type
- **value** (*object*) – Value
- **type_** (`DN_EC_T_DEFTYPE`) –

Returns`EC_E_NOERROR` on success, otherwise an error code.**Return type**`ECErrror`

```
static WriteValueToBytes (
    bytes_: List[int],
    bitOffset: int,
    bitLength: int,
    type_: DN_EC_T_DEFTYPE,
    value: object
) → ECErrror
Write value to bytes
```

Parameters

- **bytes** – Bytes
- **bitOffset** (*int*) – Bit offset
- **bitLength** (*int*) – Bit length
- **type** – Data type
- **value** (*object*) – Value
- **bytes_** (`List[int]`) –
- **type_** (`DN_EC_T_DEFTYPE`) –

Returns`EC_E_NOERROR` on success, otherwise an error code.

Return type
ECError

3.3 Types

```

class EcWrapperPythonTypes.CEcWrapperPythonTypes
    static Create_ArrayByType (type_, size)

class EcWrapperPythonTypes.Conv_DN_EC_T_DEFTYPE
    static CreateMap ()

    static TypeFromString (type_)

    static TypeToString (type_)

class EcWrapperPythonTypes.DN_ECMASERRAS_T_CONNOTIFYDESC
class EcWrapperPythonTypes.DN_ECMASERRAS_T_MARSHALERRORDESC
class EcWrapperPythonTypes.DN_ECMASERRAS_T_NONOTIFYMEMORYDESC
class EcWrapperPythonTypes.DN_ECMASERRAS_T_REGNOTIFYDESC
class EcWrapperPythonTypes.DN_EC_CMD_TYPE (value)
    EtherCAT commands

    APRD = 1
        APRD

    APRW = 3
        APRW

    APWR = 2
        APWR

    ARMW = 13
        ARMW

    BCppDummy = 4294967295
        BCppDummy

    BRD = 7
        BRD

    BRW = 9
        BRW

    BWR = 8
        BWR

    EXT = 255
        EXT

    FPRD = 4
        FPRD

    FPRW = 6
        FPRW

    FPWR = 5
        FPWR

    FRMW = 14
        FRMW

    LRD = 10
        LRD

```

```
LRW = 12
LRW
```

```
LWR = 11
LWR
```

```
NOP = 0
NOP
```

```
class EcWrapperPythonTypes.DN_EC_LOG_LEVEL (value)
```

```
ANY = 1
```

```
ANY
```

```
CRITICAL = 2
CRITICAL
```

```
ERROR = 3
ERROR
```

```
INFO = 5
INFO
```

```
INFO_API = 6
INFO_API
```

```
SILENT = 0
SILENT
```

```
UNDEFINED = 4294967295
UNDEFINED
```

```
VERBOSE = 7
VERBOSE
```

```
VERBOSE_CYC = 8
VERBOSE_CYC
```

```
WARNING = 4
WARNING
```

```
class EcWrapperPythonTypes.DN_EC_LOG_TYPE (value)
```

```
An enumeration.
```

```
DAQ = 6
```

```
DAQREADER = 7
```

```
MASTER = 0
```

```
MBXGATEWAY = 3
```

```
MBXGATEWAYSERVER = 10
```

```
MONITOR = 9
```

```
MONITORRASSERVER = 5
```

```
RASCLIENT = 1
```

```
RASSERVER = 2
```

```
SIMULATOR = 8
```

```
SIMULATORRASSERVER = 4
```

```
UNKNOWN = 4294967295
```

```
class EcWrapperPythonTypes.DN_EC_PTS_STATE (value)
```

```
pass through server states
```

```
ePtsStateDummy = 4294967295
ePtsStateDummy
```

```

ePtsStateNone = 0
    ePtsStateNone

ePtsStateNotRunning = 1
    ePtsStateNotRunning

ePtsStateRunningDisabled = 2
    ePtsStateRunningDisabled

ePtsStateRunningEnabled = 3
    ePtsStateRunningEnabled

class EcWrapperPythonTypes.DN_EC_RAS_ACCESS_LEVEL (value) ALLOW_ALL =
    ALLOW_ALL
    BLOCK_ALL = 4
        BLOCK_ALL

    EXCLUDED = 4294967295
        EXCLUDED

    READONLY = 3
        READONLY

    READWRITE = 2
        READWRITE

class EcWrapperPythonTypes.DN_EC_T_ADS_ADAPTER_START_PARMS
    cpuAffinityMask: int
        cpuAffinityMask

    dwStackSize: int
        dwStackSize

    dwThreadPriority: int
        dwThreadPriority

    targetNetID: EcWrapperPythonTypes.DN_EC_T_AOE_NETID
        targetNetID

    targetPort: int
        targetPort

class EcWrapperPythonTypes.DN_EC_T_AOE_CMD_RESPONSE
    AoE mailbox response error codes

    dwCmdResult: int
        AoE command result code

    dwErrorCode: int
        AoE response error code

    dwRsvd: int
        dwRsvd

class EcWrapperPythonTypes.DN_EC_T_AOE_NETID
    AoE NetID

    aby: List[int]
        AoE net id

class EcWrapperPythonTypes.DN_EC_T_BAD_CONNECTION_NOTIFY_DESC
    SlavePropChild: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties of child slave

    SlavePropParent: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties of parent slave

```

wPortAtChild: int
Port at child slave

wPortAtParent: int
Port at parent slave

class EcWrapperPythonTypes.DN_EC_T_BUS_DIAGNOSIS_INFO

See also Object 0x2002 Bus Diagnosis Object

dwAcyclicDatagrams: int
Number of acyclic datagrams / EtherCAT commands sent

dwAcyclicFrames: int
Number of acyclic frames sent

dwAcyclicLostFrames: int
Number of acyclic lost frames

dwCRC32ConfigChecksum: int
CRC32 checksum of the loaded configuration

dwClearCounters: int
Clear frame / datagram counter bit field

dwCyclicDatagrams: int
Number of cyclic datagrams / EtherCAT commands sent

dwCyclicFrames: int
Number of cyclic frames sent

dwCyclicLostFrames: int
Number of cyclic lost frames

dwLostFrames: int
Number of lost frames

dwNumCfgSlaves: int
Number of slaves in ENI

dwNumDCSlavesFound: int
Number of slaves with DC enabled connected

dwNumMbxSlaves: int
Number of slaves in ENI with mailbox support

dwNumSlavesFound: int
Number of slaves connected

dwRXFrames: int
Number of frames received

dwRes: List[int]
dwRes

dwTXFrames: int
Number of frames sent

class EcWrapperPythonTypes.DN_EC_T_BUS_SLAVE_INFO

adwPortSlaveIds: List[int]
[out] The slave's ID of the slaves connected to ports. See /ref EC_SLAVE_IDS 'Port slave ID's'.

bDc64Support: bool
[out] Slave supports 64 Bit DC (Bus Topology Scan)

bDcSupport: bool
[out] Slave supports DC (Bus Topology Scan)

bIsDeviceEmulation: bool
[out] Slave without Firmware. ESC register 0x0141, enabled by EEPROM offset 0x0000.8.

bIsRefClock: bool
[out] Slave is reference clock

bLineCrossed: bool
[out] Line crossed was detected at this slave

byESCRevision: int
[out] Revision number of ESC (Value of slave ESC register 0x0001)

byESCType: int
[out] Type of ESC (Value of slave ESC register 0x0000)

byPortDescriptor: int
[out] Port descriptor (Value of slave ESC register 0x0007)

dwCyclicWkcErrorCnt: int
[out] Counter for Cyclic WC Error

dwProductCode: int
[out] Product Code stored in the EEPROM at offset 0x000A

dwPropagDelay: int
[out] Propagation delay [ns]. ESC register 0x0928. This value is only valid if a DC configuration is used.

dwRevisionNumber: int
[out] Revision number stored in the EEPROM at offset 0x000C

dwSerialNumber: int
[out] Serial number stored in the EEPROM at offset 0x000E

dwSlaveAbsentCnt: int
[out] Counter for Absent/Not Present Slaves

dwSlaveDelay: int
[out] Delay behind slave [ns]. This value is only valid if a DC configuration is used.

dwSlaveId: int
[out] The slave's ID to bind bus slave and config slave information

dwSystemTimeDifference: int
[out] System time difference. (Value of slave ESC register 0x092C)

dwUnexpectedStateCnt: int
[out] Counter for Abnormal State Change

dwVendorId: int
[out] Vendor Identification stored in the EEPROM at offset 0x0008

wAlStatus: int
[out] AL status (Value of slave ESC register 0x0130)

wAlStatusCode: int
[out] AL status code. (Value of slave ESC register 0x0134 during last error acknowledge). This value is reset after a slave state change.

wAliasAddress: int
[out] The slave's alias address (Value of slave ESC register 0x0012)

wAutoIncAddress: int
[out] The slave's auto increment address

wDlStatus: int
[out] DL status (Value of slave ESC register 0x0110)

wESCBuild: int
[out] Build number of ESC (Value of slave ESC register 0x0002)

wFeaturesSupported: int
[out] Features supported (Value of slave ESC register 0x0008)

```

wIdentifyData: int
    [out] Last read identification value see EC_T_CFG_SLAVE_INFO.wIdentifyAdo

wLineCrossedFlags: int
    [out] Combination of /ref EC_LINECROSSED_FLAGS 'Line crossed flags'

wMbxSupportedProtocols: int
    [out] Supported Mailbox Protocols stored in the EEPROM at offset 0x001C

wPortState: int
    wPortState

wPrevPort: int
    [out] Connected port of the previous slave

wStationAddress: int
    [out] The slave's station address (Value of slave ESC register 0x0010)

class EcWrapperPythonTypes.DN_EC_T_CFG_SLAVE_EOE_INFO
    ecatGetCfgSlaveEoeInfo, ENI: Mailbox/EoE/InitCmds/InitCmd/Data

abyMacAddr: List[int]
    [out] MAC address

bDefaultGateway: bool
    [out] Indicates whether the default gateway could be read and is valid

bDisableEoe: bool
    [out] Indicates whether the EoE is Disabled or not

bDnsName: bool
    [out] Indicates whether the DNS name could be read and is valid

bDnsServer: bool
    [out] Indicates whether the DNS server could be read and is valid

bIpAddr: bool
    [out] Indicates whether the IP address could be read and is valid

bMacAddr: bool
    [out] Indicates whether the MAC address could be read and is valid

bSubnetMask: bool
    [out] Indicates whether the subnet mask could be read and is valid

dwRes: List[int]
    dwRes

dwSlaveId: int
    [out] Slave ID

oDefaultGateway: EcWrapperPythonTypes.DN_EC_T_IPADDR
    [out] Default gateway

oDnsServer: EcWrapperPythonTypes.DN_EC_T_IPADDR
    [out] DNS server

oIpAddr: EcWrapperPythonTypes.DN_EC_T_IPADDR
    [out] IP address

oSubnetMask: EcWrapperPythonTypes.DN_EC_T_IPADDR
    [out] Subnet mask

szDnsName: str
    [out] DNS name

class EcWrapperPythonTypes.DN_EC_T_CFG_SLAVE_INFO
    ecatGetCfgSlaveInfo

```

abyDeviceName: str
[out] Slave's configured name (80 Byte) (from ENI file)

awMasterSyncUnitIn: List[int]
ProcessData/TxPdo[1..4]@Su

Type
[out] Sync Unit (ENI)

awMasterSyncUnitOut: List[int]
ProcessData/RxPdo[1..4]@Su

Type
[out] Sync Unit (ENI)

bDcPotentialRefClock: bool
[out] Slave can be used as a reference clock (from ENI file)

bDcReferenceClock: bool
[out] Slave is reference clock (from ENI file)

bDcSupport: bool
[out] Slave supports DC (from ENI file)

bDisabled: bool
[out] Slave disabled by API SetSlaveDisabled / SetSlavesDisabled.

bDisconnected: bool
[out] Slave disconnected by API SetSlaveDisconnected / SetSlavesDisconnected.

bExtended: bool
[out] Slave generated by API ConfigExtend

bIsHCGroupPresent: bool
[out] Slave's Hot Connect group present on bus

bIsPresent: bool
[out] Slave present on bus

byPortDescriptor: int
[out] Port descriptor (ESC register 0x0007) (from ENI file)

dwDcCycleTime0: int
[out] Cycle time of Sync0 event [ns] (from ENI file)

dwDcCycleTime1: int
[out] Calculated value dwDcCycleTime1 [ns] = Cycle time of Sync1 event - Cycle time of Sync1 event + Shift time of Sync0 event (from ENI file)

dwHCGroupIdx: int
[out] Index of Hot Connect group, 0 for mandatory

dwMbxInSize: int
[out] Mailbox input byte size (from ENI file)

dwMbxInSize2: int
[out] Bootstrap mailbox input byte size (from ENI file)

dwMbxOutSize: int
[out] Mailbox output byte size (from ENI file)

dwMbxOutSize2: int
[out] Bootstrap mailbox output byte size (from ENI file)

dwMbxSupportedProtocols: int
[out] Mailbox protocols supported by the slave (from ENI file). Combination of /ref EC_MBX_PROTOCOLS 'Supported mailbox protocols' flags.

dwPdOffsIn: int
[out] Process input data bit offset (from ENI file)

dwPdOffsIn2: int
[out] 2nd sync unit process input data bit offset (from ENI file)

dwPdOffsIn3: int
[out] 3rd sync unit process input data bit offset (from ENI file)

dwPdOffsIn4: int
[out] 4th sync unit process input data bit offset (from ENI file)

dwPdOffsOut: int
[out] Process output data bit offset (from ENI file)

dwPdOffsOut2: int
[out] 2nd sync unit process output data bit offset (from ENI file)

dwPdOffsOut3: int
[out] 3rd sync unit process output data bit offset (from ENI file)

dwPdOffsOut4: int
[out] 4th sync unit process output data bit offset (from ENI file)

dwPdSizeIn: int
[out] Process input data bit size (from ENI file)

dwPdSizeIn2: int
[out] 2nd sync unit process input data bit size (from ENI file)

dwPdSizeIn3: int
[out] 3rd sync unit process input data bit size (from ENI file)

dwPdSizeIn4: int
[out] 4th sync unit process input data bit size (from ENI file)

dwPdSizeOut: int
[out] Process output data bit size (from ENI file)

dwPdSizeOut2: int
[out] 2nd sync unit process output data bit size (from ENI file)

dwPdSizeOut3: int
[out] 3rd sync unit process output data bit size (from ENI file)

dwPdSizeOut4: int
[out] 4th sync unit process output data bit size (from ENI file)

dwProductCode: int
[out] Product code (from ENI file)

dwRevisionNumber: int
[out] Revision number (from ENI file)

dwSerialNumber: int
[out] Serial number (from ENI file)

dwSlaveId: int
[out] Slave's ID to bind bus slave and config slave information

dwVendorId: int
[out] Vendor identification (from ENI file)

nDcShiftTime: int
[out] Shift time of Sync0 event [ns] (from ENI file)

wAutoIncAddress: int
[out] Slave's auto increment address (may differ from ENI file)

wIdentifyAdo: int
[out] ADO used for identification command (from ENI file)

wIdentifyData: int
[out] Identification value to be validated (from ENI file)

wNumProcessVarsInp: int
[out] Number of input process data variables (from ENI file)

wNumProcessVarsOutp: int
[out] Number of output process data variables (from ENI file)

wPrevPort: int
[out] Connected port of the previous slave (from ENI file)

wPrevStationAddress: int
[out] Station address of the previous slave (from ENI file)

wStationAddress: int
[out] Slave's configured station address (from ENI file)

wWkcStateDiagOffsIn: List[int]
0xFFFFFFFF = offset not available. WkcState bit values: 0 = Data valid, 1 = Data invalid.

Type

[out] Offset of WkcState bit in diagnosis image (ENI

Type

ProcessData/Recv[1..4]/BitStart)

wWkcStateDiagOffsOut: List[int]
0xFFFFFFFF = offset not available. WkcState bit values: 0 = Data valid, 1 = Data invalid.

Type

[out] Offset of WkcState bit in diagnosis image (ENI

Type

ProcessData/Send[1..4]/BitStart)

class EcWrapperPythonTypes.DN_EC_T_CFG_SLAVE_SM_ENTRY
ecatGetCfgSlaveSmInfo

byDirection: int
[out] Bits 2..3 ESC (0x804 + y * 8)

byOpMode: int
[out] Bits 0..1 ESC (0x804 + y * 8)

dwPdBitOffs: int
[out] Process input data bit offset (from ENI file)

dwPdBitSize: int
[out] Process input data bit size (from ENI file)

wLength: int
[out] ESC (0x802 + y * 8)

wMasterSyncUnit: int
ProcessData/TxPdo[1..4]@Su)

Type

[out] Sync Unit (ENI

wPhysAddr: int
[out] ESC (0x800 + y * 8)

wWkcStateDiagBitOffs: int
[out] Offset of WkcState bit in diagnosis image

class EcWrapperPythonTypes.DN_EC_T_CFG_SLAVE_SM_INFO

```

aoSmInfos: List[EcWrapperPythonTypes.DN_EC_T_CFG_SLAVE_SM_ENTRY]
    [out] Sync managers info

dwSlaveId: int
    [out] Slave ID

dwSmInfoNumOf: int
    [out] Number of available sync managers

class EcWrapperPythonTypes.DN_EC_T_CNF_TYPE (value) BCppDummy =
    BCppDummy

    ConfigData = 8
        Binary structured configuration
        Type
        pbyCnfData

    Data = 2
        ENI data
        Type
        pbyCnfData

    Datadiag = 3
        ENI data for diagnosis
        Type
        pbyCnfData

    FileByApp = 10
        File access provided by user application, See ef EC_T_CNF_FILEBYAPP_DESC

    Filename = 1
        ENI filename to read
        Type
        pbyCnfData

    GenEBI = 11
        Generate EBI based on bus-scan result

    GenOpENI = 6
        Generate ENI based on bus-scan result to get into OP state. The default PDO mapping read from the
        slaves is activated. See ETG2010 ‘SII Specification’, Table 14 ‘Structure Category TXPDO and RXPDO
        for each PDO’.

    GenOpENINoStrings = 9
        Generate ENI based on bus-scan result to get into OP state, does not read strings from EEPROM

    GenPreopENI = 4
        Generate ENI based on bus-scan result to get into PREOP state

    GenPreopENIWithCRC = 5
        Same as eCnfType_GenPreopENI with CRC protection

    None_ = 7
        Reset configuration

    Unknown = 0
        Unknown

class EcWrapperPythonTypes.DN_EC_T_COE_EMERGENCY
    EtherCAT CoE emergency request

    abyData: List[int]
        Error data

```

```

byErrorRegister: int
    Error register

wErrorCode: int
    Error code according to EtherCAT specification

wStationAddress: int
    Slave node address of the faulty slave

class EcWrapperPythonTypes.DN_EC_T_COE_ENTRYDESC
class EcWrapperPythonTypes.DN_EC_T_COE_OBDESC
class EcWrapperPythonTypes.DN_EC_T_COE_OBJ1018
class EcWrapperPythonTypes.DN_EC_T_COE_OBJ2001
class EcWrapperPythonTypes.DN_EC_T_COE_OBJ2001_DESC
    static Create (obj2001)

class EcWrapperPythonTypes.DN_EC_T_COE_ODLIST_TYPE (value)
    EtherCAT CoE OD list type values

ALL = 1
    List contains all objects

BCppDummy = 4294967295
    BCppDummy

Lengths = 0
    Lengths of each list type

RxPdoMap = 2
    List with PDO mappable objects

StartupParm = 5
    Only startup parameter objects

StoredFRepl = 4
    Only stored for a device replacement objects

TxPdoMap = 3
    List with objects that can be changed

class EcWrapperPythonTypes.DN_EC_T_COMMUNICATION_TIMEOUT_NOTIFY_DESC
bMainTapPortIn: bool
    Timeout occurred at the input port of the Ethernet TAP for the EtherCAT main line

    Type
        EC_TRUE

bMainTapPortOut: bool
    Timeout occurred at the output port of the Ethernet TAP for the EtherCAT main line

    Type
        EC_TRUE

class EcWrapperPythonTypes.DN_EC_T_CONFIG_LOAD_NOTIFY_DESC
dwCnfDataLen: int
    dwCnfDataLen

dwCnfType: int
    dwCnfType

pbyCnfData: object
    pbyCnfData

class EcWrapperPythonTypes.DN_EC_T_CYC_CONFIG_DESC
    descriptor for EC_IOCTL_GET_CYCLIC_CONFIG_INFO call
  
```

dwCycleTime: int
[out] Cycle time of selected cyclic entry

dwPriority: int
[out] Priority of selected cyclic entry

dwTaskId: int
Cyclic/TaskId)

Type

[out] Task ID of selected cyclic entry (ENI)

class `EcWrapperPythonTypes.DN_EC_T_CYC_COPY_INFO`

The master has to copy valid input data of this command from the source offset (bit offs in the complete process image) to a destination offset.

dwDstBitOffs: int
dwDstBitOffs

dwRes: List[int]
dwRes

dwSrcBitOffs: int
dwSrcBitOffs

wBitSize: int
wBitSize

wFlags: int
wFlags

wRes: int
wRes

wTaskId: int
wTaskId

class `EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG`

BusShift: `EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG_BUSSHIFT`
[in] BusShift configuration. Valid if eMode is set to eDcmMode_BusShift.

Dcx: `EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG_DCX`
[in] DCX configuration. Valid if eMode is set to eDcmMode_Dcx.

LinkLayerRefClock:

`EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG_LINKLAYERREFCLOCK`
[in] LinkLayerRefClock configuration. Valid if eMode is set to eDcmMode_LinkLayerRefClock.

MasterRefClock: `EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG_MASTERREFCLOCK`
[in] MasterRefClock configuration. Valid if eMode is set to eDcmMode_MasterRefClock.

MasterShift: `EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG_MASTERSHIFT`
[in] MasterShift configuration. Valid if eMode is set to eDcmMode_MasterShift.

eMode: `EcWrapperPythonTypes.DN_EC_T_DCM_MODE`
[in] DCM mode

class `EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG_BUSSHIFT`

bCtlOff: bool
[in] If set to EC_TRUE, control loop is disabled. Combined with bLogEnabled, it makes possible to analyze the natural drift between the stack cycle and the reference clock.

bLogEnabled: bool
[in] If set to EC_TRUE, logging information are generated and can be got by calling emDcmGetLog()

bUseDcLoopCtlStdValues: bool

[in] If set to EC_TRUE, the values of ESC DC time loop control register 0x930 and 0x934 are not changed by master. This could increase the time it takes to get the InSync. Use only if there are problems with the reference clock to get InSync.

dwInSyncLimit: int

[in] Limit [ns] for InSync monitoring. Default value is 20% of the cycle time. A value of 0 sets the default value.

dwInSyncSettleTime: int

[in] Settle time [ms] for InSync monitoring. Default value is 1500ms. A value of 0 sets the default value.

dwInSyncStartDelayCycle: int

[in] Delay time [ms] before InSync monitoring start

nCtlDriftErrorGain: int

[in] Multiplier for drift error. Default value is 3. A value of 0 sets the default value.

nCtlGain: int

[in] Proportional gain in ppt (part per thousand). Default is value 2. A value of 0 sets the default value.

nCtlSetVal: int

[in] Controller set value [ns]. This is the time distance between the cyclic frame send time and the DC base on bus (SYNC0 if shift is zero).

nMaxValidVal: int

[in] Error inputs above this value are considered invalid. If error input prediction is valid then the difference between the error input and the expected value is taken. Default value is 3000. A value of 0 sets the default value.

pGetTimeElapsedSinceCycleStartContext: object

[in] Optional context for the pfnGetTimeElapsedSinceCycleStart function, will be passed as first parameter to this function/static method

class `EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG_DCX`

MasterShift: `EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG_MASTERSHIFT`

[in] DCM MasterShift configuration

bCtlOff: bool

[in] If set to EC_TRUE, control loop is disabled. Combined with bLogEnabled, it makes possible to analyze the natural drift between the stack cycle and the reference clock. Also it provides reading of current adjustment value using emDcmGetAdjust function.

bLogEnabled: bool

[in] If set to EC_TRUE, logging information are generated and can be got by calling emDcmGetLog()

dwExtClockTimeout: int

[in] Wait timeout for external clock slave

dwInSyncLimit: int

[in] Limit [ns] for InSync monitoring. Default value is 20% of the cycle time. A value of 0 sets the default value.

dwInSyncSettleTime: int

[in] Settle time [ms] for InSync monitoring. Default value is 1500ms. A value of 0 sets the default value.

dwInSyncStartDelayCycle: int

[in] Delay time [ms] before InSync monitoring start

dwMaxErrCompensableOnExtClockReconnect: int

[in] Maximum error in nanoseconds that should be compensated after a reconnect of the external clock device. Synchronization restart if error exceeds this limit.

nCtlDriftErrorGain: int

[in] Multiplier for drift error. Default value is 3. A value of 0 sets the default value.

nCtlGain: int
[in] Proportional gain in ppt (part per thousand). Default is value 2. A value of 0 sets the default value.

nCtlSetVal: int
[in] Controller set value [ns]. This is the time distance between the cyclic frame send time and the DC base on bus (SYNC0 if shift is zero).

nMaxValidVal: int
[in] Error inputs above this value are considered invalid. If error input prediction is valid then the difference between the error input and the expected value is taken. Default value is 3000. A value of 0 sets the default value.

wExtClockFixedAddr: int
[in] Fixed address of external clock slave (publishing PDO 0x10F4) (optional if ENI is generated by EcEngineer)

class EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG_LINKLAYERREFCLOCK

DcStartTimeCallbackDesc:

EcWrapperPythonTypes.DN_EC_T_DC_STARTTIME_CB_DESC

[in] If not null, DC start time calculated by application, otherwise by master. See also EC_T_DC_STARTTIME_CB_DESC. Shift value configured in ENI will still be applied.

bLogEnabled: bool
[in] If set to EC_TRUE, logging information are generated and can be got by calling emDcmGetLog()

dwInSyncLimit: int
[in] Limit [ns] for InSync monitoring. Default value is 20% of the cycle time. A value of 0 sets the default value.

dwInSyncSettleTime: int
[in] Settle time [ms] for InSync monitoring. Default value is 1500ms. A value of 0 sets the default value.

nCtlSetVal: int
[in] Controller set value [ns]. This is the time distance between the cyclic frame send time and the DC base on bus (SYNC0 if shift is zero).

class EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG_MASTERREFCLOCK

bLogEnabled: bool

[in] If set to EC_TRUE, logging information are generated and can be got by calling emDcmGetLog()

dwInSyncLimit: int
[in] Limit [ns] for InSync monitoring. Default value is 20% of the cycle time. A value of 0 sets the default value.

dwInSyncSettleTime: int
[in] Settle time [ms] for InSync monitoring. Default value is 1500ms. A value of 0 sets the default value.

nCtlSetVal: int
[in] Controller set value [ns]. This is the time distance between the cyclic frame send time and the DC base on bus (SYNC0 if shift is zero).

pGetHostTimeContext: object

[in] Optional context for the pfnGetHostTime function, will be passed as first parameter to this function/static method

class EcWrapperPythonTypes.DN_EC_T_DCM_CONFIG_MASTERSHIFT

bCtlOff: bool

[in] If set to EC_TRUE, control loop is disabled. Combined with bLogEnabled, it makes possible to analyze the natural drift between the stack cycle and the reference clock. Also it provides reading of current adjustment value using emDcmGetAdjust function.

bLogEnabled: bool

[in] If set to EC_TRUE, logging information are generated and can be got by calling emDcmGetLog()

dwInSyncLimit: int

[in] Limit [ns] for InSync monitoring. Default value is 20% of the cycle time. A value of 0 sets the default value.

dwInSyncSettleTime: int

[in] Settle time [ms] for InSync monitoring. Default value is 1500ms. A value of 0 sets the default value.

dwInSyncStartDelayCycle: int

[in] Delay time [ms] before InSync monitoring start

nCtlDriftErrorGain: int

[in] Multiplier for drift error. Default value is 3. A value of 0 sets the default value.

nCtlGain: int

[in] Proportional gain in ppt (part per thousand). Default is value 2. A value of 0 sets the default value.

nCtlSetVal: int

[in] Controller set value [ns]. This is the time distance between the cyclic frame send time and the DC base on bus (SYNC0 if shift is zero).

nMaxValidVal: int

[in] Error inputs above this value are considered invalid. If error input prediction is valid then the difference between the error input and the expected value is taken. Default value is 3000. A value of 0 sets the default value.

pAdjustCycleTimeContext: object

[in] Optional context for the pfnAdjustCycleTime function, will be passed as first parameter to this function/static method

pGetTimeElapsedSinceCycleStartContext: object

[in] Optional context for the pfnGetTimeElapsedSinceCycleStart function, will be passed as first parameter to this function/static method

class EcWrapperPythonTypes.DN_EC_T_DCM_MODE (*value*) **BCppDummy =**

BCppDummy

BusShift = 1

DCM BusShift mode

Dcx = 5

DCM DCX External synchronization mode

LinkLayerRefClock = 3

DCM LinkLayer Ref Clock mode

MasterRefClock = 4

DCM Master Ref Clock mode

MasterShift = 2

DCM MasterShift mode

MasterShiftByApp = 6

DCM MasterShift controlled by application mode

Off = 0

DCM disabled

class EcWrapperPythonTypes.DN_EC_T_DCM_SYNC_NTIFY_DESC

Distributed clocks master sync (DCM)

IsInSync: int

[in] EC_TRUE as long as time of master and reference clock are in sync. False if the InSyncLimit from the bus shift configuration is exceeded.

nCtlErrorNsecAvg: int

[in] Average difference [ns] between set value and actual value of controller

```

nCtlErrorNsecCur: int
    [in] Current difference [ns] between set value and actual value of controller

nCtlErrorNsecMax: int
    [in] Maximum difference [ns] between set value and actual value of controller

class EcWrapperPythonTypes.DN_EC_T_DCX_SYNC_NTIFY_DESC
    Distributed clocks master external sync (DCX)

IsInSync: int
    EC_TRUE if external(other EtherCAT segment) and internal reference clock are in sync respectively

dwErrorCode: int
    DCX external clock error code

nCtlErrorNsecAvg: int
    Average DCX controller error [ns]

nCtlErrorNsecCur: int
    Current DCX controller error [ns]

nCtlErrorNsecMax: int
    Maximum DCX controller error [ns]

nTimeStampDiff: int
    Difference between external and internal time stamp [ns]

class EcWrapperPythonTypes.DN_EC_T_DC_CONFIGURE
bAcycDistributionDisabled: bool
    [in] If EC_TRUE, acyclic distribution is disabled

bBulkInLinkLayer: bool
    bBulkInLinkLayer

bDcInitBeforeSlaveStateChange: bool
    [in] If EC_TRUE, DC is initialized before slaves state change to PREOP

dwBurstBulk: int
    [in] Amount of burst frames per cycle during initialization burst /default-
    parm{EC_DEFAULTPARM_dwBurstBulk}

dwClntId: int
    [in] Reserved

dwDcStartTimeGrid: int
    [in] Time grid [ns] to align DC start time. With the help of the grid, several EtherCAT networks can be
    synchronized without a random shift value between the SYNC signals.

dwDevLimit: int
    dwDevLimit

dwSettleTime: int
    dwSettleTime

dwTimeout: int
    [in] Timeout [ms] for the DC initialization in which time offsets and propagation delays are evaluated
    /defaultparm{EC_DEFAULTPARM_dwDcTimeout}

dwTotalBurstLength: int
    [in] Overall amount of burst frames sent /defaultparm{EC_DEFAULTPARM_dwTotalBurstLength}

class EcWrapperPythonTypes.DN_EC_T_DC_STARTTIME_CB_DESC
pvContext: object
    [in] Arbitrarily application-defined parameter passed to pfnCallback

class EcWrapperPythonTypes.DN_EC_T_DC_SYNC_NTIFY_DESC
    Distributed clocks (DC)

```

IsInSync: int

Wire or'ed deviation value does not meet limit requirements. The limit is set by emDcConfigure().

Type

[in] EC_TRUE

Type

Wire or'ed deviation value meets limit requirements. EC_FALSE

IsNegative: int

Deviation value is positive

Type

[in] EC_TRUE

Type

Deviation value is negative, EC_FALSE

SlaveProp: *EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP*

[in] Slave properties in case of out of sync

dwDeviation: int

[in] Wire or'ed deviation value [ns] in case of in sync

class EcWrapperPythonTypes.DN_EC_T_DEFTYPE (value)

ARRAY_OF_BYTE

ARRAY_OF_BYTE

ARRAY_OF_DINT = 610

ARRAY_OF_DINT

ARRAY_OF_INT = 608

ARRAY_OF_INT

ARRAY_OF_SINT = 609

ARRAY_OF_SINT

ARRAY_OF_UDINT = 611

ARRAY_OF_UDINT

ARRAY_OF_UINT = 11

ARRAY_OF_UINT

BACKUP_PARAMETER = 43

BACKUP_PARAMETER

BIT1 = 48

BIT1

BIT10 = 57

BIT10

BIT11 = 58

BIT11

BIT12 = 59

BIT12

BIT13 = 60

BIT13

BIT14 = 61

BIT14

BIT15 = 62

BIT15

BIT16 = 63

BIT16

```
BIT2 = 49  
    BIT2  
  
BIT3 = 50  
    BIT3  
  
BIT4 = 51  
    BIT4  
  
BIT5 = 52  
    BIT5  
  
BIT6 = 53  
    BIT6  
  
BIT7 = 54  
    BIT7  
  
BIT8 = 55  
    BIT8  
  
BIT9 = 56  
    BIT9  
  
BITARR16 = 46  
    BITARR16  
  
BITARR32 = 47  
    BITARR32  
  
BITARR8 = 45  
    BITARR8  
  
BOOLEAN = 1  
    BOOLEAN  
  
BYTE = 30  
    BYTE  
  
COMMAND = 37  
    COMMAND  
  
DIAGNOSIS_OBJECT = 642  
    DIAGNOSIS_OBJECT  
  
DOMAIN = 15  
    DOMAIN  
  
DWORD = 32  
    DWORD  
  
ENUM = 40  
    ENUM  
  
ERROR_SETTING = 641  
    ERROR_SETTING  
  
EXTERNAL_SYNC_SETTINGS = 644  
    EXTERNAL_SYNC_SETTINGS  
  
EXTERNAL_SYNC_STATUS = 643  
    EXTERNAL_SYNC_STATUS  
  
FSOECOMMPAR = 646  
    FSOECOMMPAR  
  
FSOEFRAME = 645  
    FSOEFRAME
```

```
GUID = 29
    GUID

HISTORY = 642
    HISTORY

IDENTITY = 35
    IDENTITY

INTEGER16 = 3
    INTEGER16

INTEGER24 = 16
    INTEGER24

INTEGER32 = 4
    INTEGER32

INTEGER40 = 18
    INTEGER40

INTEGER48 = 19
    INTEGER48

INTEGER56 = 20
    INTEGER56

INTEGER64 = 21
    INTEGER64

INTEGER8 = 2
    INTEGER8

MODULAR_DEVICE_PROFILE = 44
    MODULAR_DEVICE_PROFILE

NULL = 0
    NULL

OCTETSTRING = 10
    OCTETSTRING

PDOCMPAR = 39
    PDOCMPAR

PDOMAPPING = 33
    PDOMAPPING

REAL32 = 8
    REAL32

REAL64 = 17
    REAL64

RECORD = 42
    RECORD

SMPAR = 41
    SMPAR

TIMEDIFFERENCE = 13
    TIMEDIFFERENCE

TIMEOFDAY = 12
    TIMEOFDAY

UNICODESTRING = 11
    UNICODESTRING
```

```

UNSIGNED16 = 6
    UNSIGNED16

UNSIGNED24 = 22
    UNSIGNED24

UNSIGNED32 = 7
    UNSIGNED32

UNSIGNED40 = 24
    UNSIGNED40

UNSIGNED48 = 25
    UNSIGNED48

UNSIGNED56 = 26
    UNSIGNED56

UNSIGNED64 = 27
    UNSIGNED64

UNSIGNED8 = 5
    UNSIGNED8

VISIBLESTRING = 9
    VISIBLESTRING

WORD = 31
    WORD

class EcWrapperPythonTypes.DN_EC_T_EEPROM_ACCESS_DENIED_DESC
    Slave EEPROM access denied

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

    SlavePropParent: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties of parent slave

    wPortAtParent: int
        Port at parent slave

class EcWrapperPythonTypes.DN_EC_T_EEPROM_CHECKSUM_ERROR_DESC
    Slave EEPROM checksum error

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

class EcWrapperPythonTypes.DN_EC_T_EOM_COMMUNICATION_TYPE (value)
    BCppDummyCommType

    RAW = 1
        RAW communication

    UDP = 0
        UDP communication

class EcWrapperPythonTypes.DN_EC_T_ETHERNET_TAP_TYPE (value)
    BCppDummyCommType
    AutoDetect =

    Auto detect TAP type. If no suitable type is detected, eEthTap_Generic is used.

    Beckhoff_ET2000 = 3
        Beckhoff ET2000 Ethernet probe

    Dummy = 4294967295
        Dummy

    Generic = 2
        Generic Ethernet switch

```

```

Hilscher_netANALYZER = 5
    Not Supported

Kunbus_TapCurious = 4
    Kunbus TAP Curious Ethernet probe

PortMirror = 6
    Port mirror Ethernet probe with separate monitoring ports for TX and RX. Only RX port supported.

Unknown = 0
    Unknown type

class EcWrapperPythonTypes.DN_EC_T_FRAMELOSS_AFTER_SLAVE_NOTIFY_DESC
    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

    wPort: int
        Port

class EcWrapperPythonTypes.DN_EC_T_FRAME_RSPERR_DESC
    EErrorType: EcWrapperPythonTypes.DN_EC_T_FRAME_RSPERR_TYPE
        Frame response error type

    bIsCyclicFrame: bool
        Indicates whether the lost frame was a cyclic frame

    byEcCmdHeaderIdxAct: int
        eRspErr_WRONG_IDX and eRspErr_UNEXPECTED

        Type
            Actually received IDX value, this value is only valid for acyclic frames in case of
            EErrorType is equal to

    byEcCmdHeaderIdxSet: int
        Expected IDX value, this value is valid only for acyclic frames in case EErrorType is not equal to
        eRspErr_UNEXPECTED

    dwTaskId: int
        Cyclic/TaskId). Only valid if bIsCyclicFrame is set

        Type
            Cyclic Task ID (ENI

    wCycFrameNum: int
        Number of the lost cyclic frame from the ENI

class EcWrapperPythonTypes.DN_EC_T_FRAME_RSPERR_TYPE (value)
    EtherCAT frame response error descriptor

    BCppDummy = 4294967295
        BCppDummy

    CRC = 8
        Ethernet frame with CRC error received

    FOREIGN_SRC_MAC = 6
        Frame with MAC from other Master received

    FRAME_RETRY = 4
        Ethernet frame will be re-sent (timeout, frame loss)

    NON_EC_CAT_FRAME = 7
        Non EtherCAT frame received

    NO_RESPONSE = 1
        No Ethernet frame received (timeout, frame loss)

    RETRY_FAIL = 5
        All retry mechanism fails to re-sent acyclic frames

```

```

UNDEFINED = 0
    Undefined

UNEXPECTED = 3
    Unexpected frame was received

WRONG_IDX = 2
    Wrong IDX value in acyclic frame

class EcWrapperPythonTypes.DN_EC_T_FSOE_CONNECTION_STATE (value)
    FSOE state (see ETG.5001.4 v0.1.1, table 9)

    BCppDummy = 4294967295
        BCppDummy

    CONNECTION = 2
        CONNECTION

    DATA = 4
        DATA

    FAILSAFE = 5
        FAILSAFE

    PARAMETER = 3
        PARAMETER

    RESET = 0
        RESET

    SESSION = 1
        SESSION

class EcWrapperPythonTypes.DN_EC_T_FSOE_CONNECTION_STATECHANGED_NOTIFY_DESC
    FSoE connection state changed descriptor

    eConnStateNew: EcWrapperPythonTypes.DN_EC_T_FSOE_CONNECTION_STATE
        New connection state

    eConnStateOld: EcWrapperPythonTypes.DN_EC_T_FSOE_CONNECTION_STATE
        New connection state

    oSlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

    wFsoeSlaveSafeAddress: int
        Safe address of the FSoE slave

class EcWrapperPythonTypes.DN_EC_T_GET_SLAVE_INFO
    ecatGetSlaveInfo

    aPortSlaveIds: List[int]
        [out] slave IDs connected to ports

    abyDeviceName: str
        abyDeviceName

    bDc64Support: bool
        [out] Slave does support 64Bit DC

    bDcSupport: bool
        [out] Slave does support DC

    bIsHCGroupPresent: bool
        [out] The hot connect group of the slave is present

    bIsMailboxSlave: bool
        [out] Whether slave support mailboxes

```

bIsOptional: bool
[out] Slave is in an optional hot connect group

bIsPresent: bool
[out] Slave is currently present on bus

byESCType: int
[out] ESC Node Type

byPortDescriptor: int
[out] Port Descriptor (ESC register 0x0007)

dwErrorCode: int
[out] Last return code

dwMbxInSize: int
[out] Mailbox 1 input size

dwMbxInSize2: int
[out] Mailbox 2 input size

dwMbxOutSize: int
[out] Mailbox 1 output size

dwMbxOutSize2: int
[out] Mailbox 2 output size

dwPdOffsIn: int
[out] Process data offset of Input Data (in bits)

dwPdOffsIn2: int
[out] Process data offset of Input data (in bits)

dwPdOffsIn3: int
[out] Process data offset of Input Data (in bits)

dwPdOffsIn4: int
[out] Process data offset of Input Data (in bits)

dwPdOffsOut: int
[out] Process data offset of Output Data (in bits)

dwPdOffsOut2: int
[out] Process data offset of Output Data (in bits)

dwPdOffsOut3: int
[out] Process data offset of Output Data (in bits)

dwPdOffsOut4: int
[out] Process data offset of Output Data (in bits)

dwPdSizeIn: int
[out] Process data size of Input Data (in bits)

dwPdSizeIn2: int
[out] Process data size of Input Data (in bits)

dwPdSizeIn3: int
[out] Process data size of Input Data (in bits)

dwPdSizeIn4: int
[out] Process data size of Input Data (in bits)

dwPdSizeOut: int
[out] Process data size of Output Data

dwPdSizeOut2: int
[out] Process data size of Output Data

dwPdSizeOut3: int
[out] Process data size of Output Data

dwPdSizeOut4: int
[out] Process data size of Output Data

dwProductCode: int
Product Code

dwRevisionNumber: int
Revision Number

dwSBErrorCode: int
[out] Last return value from SB

dwScanBusStatus: int
Status during last Bus Scan

dwSerialNumber: int
Serial Number

dwSlaveId: int
[out] Slave ID

dwSystemTimeDifference: int
dwSystemTimeDifference

dwVendorId: int
Vendor Identification

wAlStatusCode: int
[out] AL Status Code (ESC register 0x0134)

wAlStatusValue: int
[out] AL Status Register Value (ESC register 0x0130)

wAliasAddress: int
[out] Slave alias address

wCfgPhyAddress: int
[out] Slave configured station address

wNumProcessVarsInp: int
[out] Number of output process data variables

wNumProcessVarsOutp: int
[out] Number of input process data variables

wPhysAddress: int
[out] Slave station address

wPortState: int
[out] port link state (SB Instance)

wSupportedMbxProtocols: int
AoE, EoE, CoE, FoE, SoE

Type

[out] Supported mailbox protocols

```
class EcWrapperPythonTypes.DN_EC_T_HC_DETECTALLGROUP_NOTIFY_DESC
  adwGroupMask: List[int]
    Bitmask of first 3200 Groups

  dwGroupCount: int
    Total number of Groups

  dwGroupMask: int
    Bitmask of first 32 Groups. 1 = present, 0 = absent.
```

```

dwGroupsPresent: int
    Number of connected groups

dwResultCode: int
    Result of Group detection

class EcWrapperPythonTypes.DN_EC_T_HC_GROUP_INFO
bIsMandatory: bool
    [out] EC_TRUE if HC group is mandatory, EC_FALSE otherwise

bIsPresent: bool
    [out] EC_TRUE if HC group is present, EC_FALSE otherwise

dwHcGroupIdx: int
    [out] Index of Hot Connect group

dwNumOfSlaves: int
    [out] Number of Slaves within the group

szGroupName: str
    [out] Hot connect group name (from ENI file)

wHeadSlaveStationAddress: int
    [out] Physical address of the head slave of the HC group

wIdentificationAdo: int
    [out] Ado of Identification check

wIdentificationVal: int
    [out] Val of Identification check

class EcWrapperPythonTypes.DN_EC_T_INITCMD_ERR_DESC
EErrorType: EcWrapperPythonTypes.DN_EC_T_INITCMD_ERR_TYPE
    Init command error type

SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
    Slave properties

achStateChangeName: str
    State change description when the error occurred

szComment: str
    Comment (ENI)

class EcWrapperPythonTypes.DN_EC_T_INITCMD_ERR_TYPE (value)
    EtherCAT init command response error descriptor

ALSTATUS_ERROR = 5
    Error in AL Status Register

BCppDummy = 4294967295
    BCppDummy

FAILED = 3
    Init commands failed (state could not be reached)

MBXSLAVE_ERROR = 6
    Error at Mailbox Init Command

NOT_PRESENT = 4
    Slave not present on the bus

NO_ERROR = 0
    No error

NO_RESPONSE = 1
    No Ethernet frame received (timeout)

```

PDI_WATCHDOG = 7
PDI watchdog has been detected

VALIDATION_ERR = 2
Validation error (invalid slave command response)

class EcWrapperPythonTypes.DN_EC_T_INITRASP_PARAMS

RAS parameters

class EcWrapperPythonTypes.DN_EC_T_INIT_MASTER_PARAMS

MasterRedParams: *EcWrapperPythonTypes.DN_EC_T_MASTER_RED_PARAMS*

[in] Master Redundancy parameters

PerfMeasInternalParams:

EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_INTERNAL_PARAMS

[in] Internal performance measurement parameters

bApiLockByApp: bool
application is responsible for locking.

Type

[in] Lock pending API against emDeinitMaster(). EC_FALSE (default)

Type

locked internally. EC_TRUE

bNoConsecutiveAcycFrames: bool
Don't process and send acyclic frames in the same cycle to reduce CPU load.

Type

[in] EC_FALSE(default)

Type

no restriction. EC_TRUE

dwAdditionalEoEEndpoints: int

[in] Additional EoE endpoints

dwBusCycleTimeNsec: int

[in] Bus cycle time [ns]

dwEcatCmdMaxRetries: int

[in] Maximum retries to send pending EtherCAT command frames /default-param{EC_DEFAULTPARAM_dwMaxAcycCmdRetries}

dwEcatCmdTimeout: int

[in] Timeout [ms] to send pending EtherCAT command frames /default-param{EC_DEFAULTPARAM_dwAcycCmdTimeout}

dwLogLevel: *EcWrapperPythonTypes.DN_EC_LOG_LEVEL*

dwLogLevel

dwMaxAcycBytesPerCycle: int

[in] Maximum bytes sent during eUsrJob_SendAcycFrames per cycle /default-param{EC_DEFAULTPARAM_dwMaxAcycBytesPerCycle}

dwMaxAcycCmdsPerCycle: int

[in] Maximum commands sent during eUsrJob_SendAcycFrames per cycle /default-param{EC_DEFAULTPARAM_dwMaxAcycCmdsPerFrame}

dwMaxAcycFramesPerCycle: int

[in] Maximum frames sent during eUsrJob_SendAcycFrames per cycle /default-param{EC_DEFAULTPARAM_dwMaxAcycFramesPerCycle}

dwMaxAcycFramesQueued: int

[in] Maximum queued acyclic frames

```

dwMaxBusSlaves: int
    [in] Maximum pre-allocated bus slave objects /defaultparm{EC_DEFAULTPARM_dwMaxBusSlaves}

dwMaxQueuedS2SMbxTfer: int
    [in] S2S Fifo number of entries /defaultparm{EC_DEFAULTPARM_dwMaxQueuedS2SMbxTfer}

dwMaxS2SMbxSize: int
    [in] Size of the queued S2S mailbox in bytes /defaultparm{EC_DEFAULTPARM_dwMaxS2SMbxSize}

dwMaxSlavesProcessedPerCycle: int
    [in] Maximum slave-related state machine calls per cycle (default = all)

oLinkParms: EcWrapperPythonTypes.DN_EC_T_LINK_PARMS
    oLinkParms

oLinkParmsRed: EcWrapperPythonTypes.DN_EC_T_LINK_PARMS
    oLinkParmsRed

wMaxSlavesProcessedPerBusScanStep: int
    [in] Maximum slave-related calls per cycle during bus scans (default = all)

class EcWrapperPythonTypes.DN_EC_T_INIT_MBXGATEWAY_PARMS
    MBX Gateway Parameters

class EcWrapperPythonTypes.DN_EC_T_INIT_PARMS
    Abstract initialization parameters

class EcWrapperPythonTypes.DN_EC_T_INIT_PARMS_MASTER
    EcMaster initialization parameters

class EcWrapperPythonTypes.DN_EC_T_INIT_PARMS_MASTER_RAS_SERVER
    RAS server of EcMaster initialization parameters

class EcWrapperPythonTypes.DN_EC_T_INIT_PARMS_MBXGATEWAY_CLIENT
    Mailbox gateway client of EcMaster initialization parameters

class EcWrapperPythonTypes.DN_EC_T_INIT_PARMS_MBXGATEWAY_SERVER
    Mailbox gateway server of EcMaster initialization parameters

class EcWrapperPythonTypes.DN_EC_T_INIT_PARMS_MONITOR
    EcMonitor initialization parameters

class EcWrapperPythonTypes.DN_EC_T_INIT_PARMS_MONITOR_RAS_SERVER
    RAS server of EcMonitor initialization parameters

class EcWrapperPythonTypes.DN_EC_T_INIT_PARMS_RAS_CLIENT
    RAS client initialization parameters

class EcWrapperPythonTypes.DN_EC_T_INIT_PARMS_SIMULATOR
    EcSimulator initialization parameters

class EcWrapperPythonTypes.DN_EC_T_INIT_PARMS_SIMULATOR_RAS_SERVER
    RAS server of EcSimulator initialization parameters

class EcWrapperPythonTypes.DN_EC_T_INNER_IPADDR
    by: List[int]
        IPv4 address (endianness independent)

class EcWrapperPythonTypes.DN_EC_T_IOCTL (value)
    Specifies IOCTL codes

    ACTIVATE_VOE_RECV_FIFO = 81

    ADDITIONAL_VARIABLES_FOR_SPECIFIC_DATA_TYPES = 58

    ADD_BRD_SYNC_WINDOW_MONITORING = 20

    ADD_COE_INITCMD = 26

    ADD_PADDING_TO_DC_FRAMES = 80

```

```
ALL_SLAVES_MUST_REACH_MASTER_STATE = 49
CHECK_OEM_KEY = 16711686
CLEAR_MASTER_INFO_COUNTERS = 63
CLR_SLVSTATISTICS = 46
DCM_GET_LOG = 458756
DCM_REGISTER_STARTSO_CALLBACK = 458755
DCM_REGISTER_TIMESTAMP = 458753
DCM_UNREGISTER_TIMESTAMP = 458754
DC_ENABLE_ALL_DC_SLV = 196628
DC_FIRST_DC_SLV_AS_REF_CLOCK = 196626
DC_SETSYNCSTARTOFFSET = 196625
DC_SET_RED_PROPAGDELAY = 196629
DC_SHIFT_SYSTIME = 196624
DC_SLAVE_CONTROLLED_BY_PDI = 196627
DC_SLV_SYNC_DEVLIMIT_GET = 196615
DC_SLV_SYNC_DEVLIMIT_SET = 196614
DC_SLV_SYNC_STATUS_GET = 196613
FORCE_BROADCAST_DESTINATION = 42
FORCE_SLVSTAT_COLLECTION = 44
GET_CYCLIC_CONFIG_INFO = 15
GET_FAST_CONTEXT = 16711684
GET_LINKLAYER_MODE = 16
GET_MASTER_MAX_STATE = 74
GET_NOTIFICATION_ENABLED = 54
GET_PDMEMORYSIZE = 40
GET_SLVSTATISTICS = 45
GET_SLVSTAT_PERIOD = 66
HC_CONFIGURETIMEOUTS = 393219
HC_GETMODE = 393218
HC_SETMODE = 393217
INITIATE_UPDATE_ALL_SLAVE_STATE = 19
ISLINK_CONNECTED = 6
IS_MAIN_LINK_CONNECTED = 24
IS_RED_LINK_CONNECTED = 25
IS_SLAVETOSLAVE_COMM_CONFIGURED = 17
LINKLAYER_DBG_MSG = 10
MASTEROD_SET_VALUE = 51
MONITOR_SET_COESDO_CLEAR_ON_READ = 3439329281
MONITOR_SET_IGNORE_COE_API_TIMEOUTS = 3439329282
```

```
ONLY_PROCESS_DATA_IN_IMAGE = 21
REALLOC_MBX_QUEUE = 84
REGISTERCLIENT = 2
REGISTER_CYCFRAME_RX_CB = 22
REGISTER_FRAMECALLBACK = 82
REGISTER_PDMEMORYPROVIDER = 41
REG_DC_SLV_SYNC_NTIFY = 196611
RESET_SLAVE = 13
SB_ACCEPT_TOPOLOGY_CHANGE = 327696
SB_BUSCNF_GETSLAVE_INFO = 327685
SB_BUSCNF_GETSLAVE_INFO_EEP = 327686
SB_BUSCNF_GETSLAVE_INFO_EX = 327689
SB_ENABLE = 327687
SB_GET_BUS_SLAVE_PORTS_INFO = 327702
SB_NOTIFY_UNEXPECTED_BUS_SLAVES = 327697
SB_RESTART = 327681
SB_SET_BUSCNF_READ_PROP = 327692
SB_SET_BUSCNF_VERIFY = 327683
SB_SET_BUSCNF_VERIFY_PROP = 327684
SB_SET_ERROR_ON_CROSSED_LINES = 327694
SB_SET_ERROR_ON_LINE_BREAK = 327703
SB_SET_IDENTIFICATION_FALLBACK_ENABLED = 327704
SB_SET_JUNCTION_REDUNDANCY_MODE = 327701
SB_SET_NOTIFY_NOT_CONNECTED_PORT_A = 327699
SB_SET_NOTIFY_UNEXPECTED_CONNECTED_PORT = 327700
SB_SET_NO_DC_SLAVES_AFTER_JUNCTION = 327705
SB_SET_RED_ENHANCED_LINE_CROSSED_DETECTION_ENABLED = 327698
SB_SET_TOPOLOGY_CHANGED_DELAY = 327693
SB_SET_TOPOLOGY_CHANGED_DELAYS = 327706
SB_SET_TOPOLOGY_CHANGE_AUTO_MODE = 327695
SB_STATUS_GET = 327682
SET_ADJUST_CYCFRAMES_AFTER_SLAVES_STATE_CHANGE = 65
SET_AUTO_ACK_AL_STATUS_ERROR_ENABLED = 61
SET_AUTO_ADJUST_CYCCMD_WKC_ENABLED = 62
SET_AUTO_ADJUST_MBX_STATE_COUNT_ENABLED = 89
SET_AUTO_REFRESH_LINK_STATUS = 86
SET_BUS_CYCLE_TIME = 57
SET_BUS_DIAGNOSIS_COUNTERS_OVERFLOW_ENABLED = 78
SET_CLEAR_ERROR_REGISTERS_INITCMD_ENABLED = 88
```

```
SET_CONFIGDATA_MEMORY_POOL = 75
SET_COPYINFO_IN_SENDCYCFRAMES = 56
SET_CYCFRAME_LAYOUT = 52
SET_DIAGMSG_CODE_BASE = 77
SET_EOE_DEFERRED_SWITCHING_ENABLED = 67
SET_FRAME_LOSS_SIMULATION = 16711681
SET_FRAME_RESPONSE_ERROR_NOTIFY_MASK = 8
SET_GENENI_ASSIGN_EEPROM_BACK_TO_ECAT = 60
SET_GEN_ENI_PARM = 83
SET_IGNORE_INPUTS_ON_WKC_ERROR = 59
SET_IGNORE_SWAPDATA = 72
SET_MAILBOX_POLLING_CYCLES = 71
SET_MASTER_DEFAULT_TIMEOUTS = 55
SET_MASTER_MAX_STATE = 73
SET_MBX_RETRYACCESS_COUNT = 47
SET_MBX_RETRYACCESS_PERIOD = 48
SET_NEW_BUSSLAVES_TO_INIT = 68
SET_NOTIFICATION_CTL = 50
SET_NOTIFICATION_ENABLED = 53
SET_OEM_KEY = 16711685
SET_RXFRAME_LOSS_SIMULATION = 16711682
SET_SENDCYCFRAMES_BEFORE_PROCESSALLRXFRAMES = 79
SET_SLAVE_MAX_STATE = 85
SET_SLVSTAT_PERIOD = 43
SET_SPLIT_FRAME_PROCESSING_ENABLED = 64
SET_STOP_TRANSITION_ON_PDI_WATCHDOG = 76
SET_TXFRAME_LOSS_SIMULATION = 16711683
SET_ZERO_INPUTS_ON_FRAME_LOSS = 87
SET_ZERO_INPUTS_ON_WKC_ERROR = 70
SET_ZERO_INPUTS_ON_WKC_ZERO = 69
SIMULATOR_GET_MBX_PROCESS_CTL = 3422552066
SIMULATOR_SET_MBX_PROCESS_CTL = 3422552065
SLAVE_LINKMESSAGES = 14
SLV_ALIAS_ENABLE = 327690
UNREGISTERCLIENT = 3
UNREG_DC_SLV_SYNC_NTFY = 196612

class EcWrapperPythonTypes.DN_EC_T_IPADDR
    sAddr: EcWrapperPythonTypes.DN_EC_T_INNER_IPADDR
    sAddr

class EcWrapperPythonTypes.DN_EC_T_JUNCTION_RED_CHANGE_DESC
```

```

SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
    Slave properties

bLineBreak: bool
    EC_TRUE for line break, EC_FALSE for line fixed

wPort: int
    Port

class EcWrapperPythonTypes.DN_EC_T_LINE_CROSSED_DESC
    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

    wInputPort: int
        Port where frame was received

class EcWrapperPythonTypes.DN_EC_T_LINKLAYER_TIMING (value) BCppDummy =

    BCppDummy

    TMR = 1
        Real-time Ethernet Driver Timer

    TTS = 1
        Real-time Ethernet Driver Time Triggered Send

    Undefined = 0
        Link Layer Timing is undefined must be TTS or TMR

class EcWrapperPythonTypes.DN_EC_T_LINKMODE (value) BCppDummy =

    BCppDummy

    INTERRUPT = 1
        Link is in interrupt mode and is triggered by interrupts

    POLLING = 2
        Link is in polling mode and is polled periodically

    UNDEFINED = 0
        Link is in an undefined state, must be polling or interrupt

class EcWrapperPythonTypes.DN_EC_T_LINK_PARMS
    ToNic ()

class EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_BCMNETXTREME
    dwRxBuffers: int
        Receive buffer count. Must be a power of 2, maximum 1024.

    dwTxBuffers: int
        Transmit buffer count. Must be a power of 2, maximum 1024.

class EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_DEFAULT

class EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_DPK
    bDontCheckLinkStatus: bool
        Don't check link status (forced)

    bEalInitDeinitByApp: bool
        DPDK EAL layer is initialized and deinitialized by user App

    bSetPromiscuousMode: bool
        Change promiscuous mode setting

    dwPortId: int
        DPDK port ID

    wRxBufferCnt: int
        defaults to DPDK internal setting

```

Type

Receive buffer count, 0

wTxBufferCnt: int

default to DPDK internal setting

Type

Transmit buffer count, 0

class `EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_EOM`**abyHostIpAddress: List[int]**

Host IP Address

abyTargetIpAddress: List[int]

IP address of the EOM coupler

abyTargetMac: List[int]

MAC address of the EOM coupler

eCommType: *EcWrapperPythonTypes.DN_EC_T_EOM_COMMUNICATION_TYPE*

Communication type

oHwLinkParms: *EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_NIC*

oHwLinkParms

class `EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_I8254X`**bDisableLocks: bool**

Disable locks

bNotUseDmaBuffers: bool

bNotUseDmaBuffers

dwAutoNegTimeout: int

Timeout [ms] for auto negotiation

wRxBufferCnt: int

default to 96

Type

Receive buffer count, 0

wRxBufferSize: int

buffer optimized for standard Ethernet frame.

Type

Receive buffer size for a single Ethernet frame. 0

wTxBufferCnt: int

default to 96

Type

Transmit buffer count, 0

wTxBufferSize: int

buffer optimized for standard Ethernet frame.

Type

Transmit buffer size for a single Ethernet frame. 0

class `EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_INTELGBE`**bDisableLocks: bool**

Disable locks

bNoPhyCtrlOnConnect: bool

No PHY control (e.g. PHY reset, PHY PM settings, Gbits Ctrl) on link connection detected

Type

EC_TRUE

bNotUseDmaBuffers: bool
bNotUseDmaBuffers

dwAutoNegTimeout: int
Timeout [ms] for auto negotiation

wRxBufferCnt: int
default to 96

Type
Receive buffer count, 0

wRxBufferSize: int
buffer optimized for standard Ethernet frame.

Type
Receive buffer size for a single Ethernet frame. 0

wTxBufferCnt: int
default to 96

Type
Transmit buffer count, 0

wTxBufferSize: int
buffer optimized for standard Ethernet frame.

Type
Transmit buffer size for a single Ethernet frame. 0

class `EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_MULTIPLIER`

dwPort: int
Used CU2508 downlink port

eMultiplierType: *EcWrapperPythonTypes.DN_EC_T_MULTIPLIER_TYPE*
Type of the Multiplier Ethernet port

oHwLinkParms: *EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_NIC*
oHwLinkParms

class `EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_NDIS`

abyIpAddress: List[int]
IP address of network adapter

bDisableForceBroadcast: bool
FF:FF:FF:FF

Type
Don't change target MAC address to FF

Type
FF

bDisablePromiscuousMode: bool
Disable adapter promiscuous mode

szAdapterName: str
ServiceName of network adapter, see HKLM/SOFTWARE/Microsoft/Windows NT/CurrentVersion/NetworkCards in registry (zero terminated)

class `EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_NIC`

class `EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_REMOTE`

abyDstIpAddress: List[int]
Destination adapter IP address (connect)

abyMac: List[int]
MAC address

```

abySrcIpAddress: List[int]
    Source adapter IP address (listen)

dwRxBufferCnt: int
    Frame buffer count for interrupt service thread (IST)

dwSocketType: int
    Socket type. Must be set to 1 (emrassocktype_tcp).

wDstPort: int
    Destination port number (connect)

wSrcPort: int
    Source port number (listen)

class EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_SIMULATOR
PerfMeasInternalParms:
EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_INTERNAL_PARMS
    Internal performance measurement parameters

abyMac: List[int]
    MAC address

aoLinkParms: List[EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_NIC]
    aoLinkParms

bConnectHcGroups: bool
    Connect hot connect groups in topology (floating group heads to free ports)

bDisableProcessDataImage: bool
    Don't allocate Process Data Image at simulator (legacy support, CiA402 simulation)

bJobsExecutedByApp: bool
    implicitly by emllSimulator

    Type
        EC_FALSE

    Type
        esExecJob explicitly called by application, EC_TRUE

bStartRasServer: bool
    bStartRasServer

dwBusCycleTimeNsec: int
    Cycle time of simulator job task

dwRasPriority: int
    RAS server threads priority

dwRasStackSize: int
    RAS server threads stack size

dwRxBufferCnt: int
    Frame buffer count for IST

dwSimulatorAddress: int
    Reserved

oDeviceConnection:
EcWrapperPythonTypes.DN_EC_T_SIMULATOR_DEVICE_CONNECTION_DESC
    See EC_SIMULATOR_DEVICE_CONNECTION_TYPE...

oRasCpuAffinityMask: int
    RAS server threads CPU affinity mask

qwOemKey: int
    64 bit OEM key (optional)

```

```

szEniFilename: str
    szEniFilename

szLicenseKey: str
    License key (zero terminated string)

wRasServerPort: int
    RAS server port

class EcWrapperPythonTypes.DN_EC_T_LINK_PARMS SOCKRAW
bDisableForceBroadcast: bool
    FF:FF:FF:FF

    Type
        Don't change target MAC address to FF

    Type
        FF

bReplacePaddingWithNopCmd: bool
    Prevent adding Ethernet padding to work-around EtherCAT corruption bugs from native Linux driver(s)

bSetCoalescingParms: bool
    Set Coalescing parameters to enhance the link layer performance

bSetPromiscuousMode: bool
    Enable promiscuous mode at network adapter

bUsePacketMmapRx: bool
    Use PACKET_MMAP PACKET_RX_RING for receive

szAdapterName: str
    Native ETH device name, e.g. 'eth0' (zero terminated)

class EcWrapperPythonTypes.DN_EC_T_LINK_PARMS TAP
abyMac: List[int]
    MAC address

dwRxBufferCnt: int
    Frame buffer count for IST

szAdapterGuid: str
    GUID of virtual interface to connect (zero terminated)

szAdapterName: str
    Native tap device name, e.g. 'tap0' (zero terminated)

class EcWrapperPythonTypes.DN_EC_T_LINK_PARMS VLAN
eVlanMode: EcWrapperPythonTypes.DN_EC_T_VLAN_MODE
    Vlan switch operation mode

oHwLinkParms: EcWrapperPythonTypes.DN_EC_T_LINK_PARMS_NIC
    oHwLinkParms

wVlanId: int
    VLAN Identifier (VID)

wVlanPrio: int
    VLAN Priority code point (PCP)

class EcWrapperPythonTypes.DN_EC_T_LINK_PARMS WINPCAP
abyIpAddress: List[int]
    IP address

bFilterInput: bool
    Filter input if EC_TRUE. This is needed on some system if the winpcap library notify the sent frames
    to the network adapter.

```

```
szAdapterId: str
    {XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}
```

Type

Adapter ID, format

```
class EcWrapperPythonTypes.DN_EC_T_LINK_VAR_INFO
```

```
dwApplVarBitOffs: int
    [in] Bit offset of variable
```

```
pbyApplVar: object
```

[in] Pointer to application variable where process variable should be linked to

```
pvContext: object
```

[in] Arbitrarily application-defined parameter

```
wApplVarBitSize: int
```

[in] Size of variable (in bits)

```
wApplVarType: int
```

[in] Type of application variable

```
class EcWrapperPythonTypes.DN_EC_T_LOG_PARMS
```

```
dwLogLevel: int
```

[in] Log level. See /ref EC_LOG_LEVELS 'EC_LOG_LEVEL...'

```
class EcWrapperPythonTypes.DN_EC_T_MAILBOX_STATISTICS
```

See also Object 0x2006 Mailbox Statistics Object

```
Aoe: EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER_DUPLEX
    AoE mailbox transfer statistics
```

```
Coe: EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER_DUPLEX
    CoE mailbox transfer statistics
```

```
Eoe: EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER_DUPLEX
    EoE mailbox transfer statistics
```

```
Foe: EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER_DUPLEX
    FoE mailbox transfer statistics
```

```
RawMbx: EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER_DUPLEX
    Raw mailbox transfer statistics
```

```
Soe: EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER_DUPLEX
    SoE mailbox transfer statistics
```

```
VoE: EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER_DUPLEX
    VoE mailbox transfer statistics
```

```
aRes: EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER_DUPLEX
    aRes
```

```
class EcWrapperPythonTypes.DN_EC_T_MASTER_INFO
```

ecatGetMasterInfo

```
BusDiagnosisInfo: EcWrapperPythonTypes.DN_EC_T_BUS_DIAGNOSIS_INFO
    Bus diagnostics
```

```
MailboxStatistics: EcWrapperPythonTypes.DN_EC_T_MAILBOX_STATISTICS
    Mailbox statistics
```

```
RedundancyDiagnosisInfo:
```

```
EcWrapperPythonTypes.DN_EC_T_REDUNDANCY_DIAGNOSIS_INFO
    Redundancy diagnosis info
```

```
dwMasterStateSummary: int
```

Master state summary

```

dwMasterVersion: int
    Master version

dwMasterVersionType: int
    Master version type. See /ref EC_VERSION_TYPES 'EC_VERSION_TYPE'

wMasterStateSummaryDiagBitOffset: int
    Bit offset of Master state summary in diagnosis image

wMasterStateSummaryDiagBitSize: int
    Bit offset size of Master state summary in diagnosis image

class EcWrapperPythonTypes.DN_EC_T_MASTER_RED_PARMS
    Master Redundancy initialization parameters, see ecatInitMaster

bEnabled: bool
    [in] Set to EC_TRUE if using Master Redundancy

bUpdateSlavePdIn: bool
    [in] Set to EC_TRUE to update Slave INPUT Process Data Image at INACTIVE Master (from CMF)

bUpdateSlavePdOut: bool
    [in] Set to EC_TRUE to update Slave OUTPUT Process Data Image at INACTIVE Master (from CSF)

dwMaxAcycFramesPerCycle: int
    [in] Maximum acyclic Master Red frames sent per cycle

wMasterPdInSize: int
    [in] INACTIVE to ACTIVE Master Process Data (in bytes)

wMasterPdOutSize: int
    [in] ACTIVE to INACTIVE Master Process Data (in bytes)

class EcWrapperPythonTypes.DN_EC_T_MBOXRCV

class EcWrapperPythonTypes.DN_EC_T_MBOX_FOE_ABORT_DESC
    EtherCAT FoE error

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

    achErrorString: str
        FoE error string

    dwErrorCode: int
        Error code

class EcWrapperPythonTypes.DN_EC_T_MBOX_SDO_ABORT_DESC
    EtherCAT Sdo abort

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

    bySubIndex: int
        SDO object sub index

    dwErrorCode: int
        Error code

    wObjIndex: int
        SDO object index

class EcWrapperPythonTypes.DN_EC_T_MBXRCV_INVALID_DATA_DESC
    Invalid mailbox data received error

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

class EcWrapperPythonTypes.DN_EC_T_MBXTFER_STATUS (value)
    EtherCAT mailbox status values

```

```
BCppDummy = 4294967295
    BCppDummy

Idle = 0
    Mailbox transfer object not in use

Pend = 1
    Mailbox transfer in process

TferDone = 2
    Mailbox transfer completed

TferReqError = 3
    Mailbox transfer request error

TferWaitingForContinue = 4
    Mailbox transfer waiting for continue, object owned by application

class EcWrapperPythonTypes.DN_EC_T_MBXTFER_TYPE (value)
    EtherCAT mailbox type values

AOE_READ = 17
    AoE read

AOE_READWRITE = 19
    AoE read/write

AOE_WRITE = 18
    AoE write

AOE_WRITECONTROL = 20
    AoE write control

BCppDummy = 4294967295
    BCppDummy

COE_EMERGENCY = 5
    CoE emergency request

COE_GETENTRYDESC = 4
    CoE Get object entry description

COE_GETOBDESC = 3
    CoE Get object description

COE_GETODLIST = 2
    CoE Get object dictionary list

COE_RX_PDO = 6
    CoE RxPDO

COE_SDO_DOWNLOAD = 0
    CoE SDO download

COE_SDO_UPLOAD = 1
    CoE SDO upload

EOE_RECEIVE_FRAME = 28
    EoE receive frame

EOE_SEND_FRAME = 27
    EoE send frame

EOE_SET_IP = 29
    EoE set IP address

FOE_DOWNLOAD_REQ = 26
    FoE download request
```

```
FOE_FILE_DOWNLOAD = 8
    FoE download

FOE_FILE_UPLOAD = 7
    FoE upload

FOE_SEG_DOWNLOAD = 22
    FoE segmented download

FOE_SEG_UPLOAD = 23
    FoE segmented upload

FOE_UPLOAD_REQ = 25
    FoE upload request

RAWMBX = 21
    Raw mbx

S2SMBX = 24
    S2S mbx

SOE_EMERGENCY = 14
    SoE emergency

SOE_NOTIFICATION = 13
    SoE notification

SOE_READREQUEST = 9
    SoE read request

SOE_READRESPONSE = 10
    SoE read response

SOE_WRITEREQUEST = 11
    SoE write request

SOE_WRITERESPONSE = 12
    SoE write response

VOE_MBX_READ = 15
    VoE read

VOE_MBX_WRITE = 16
    VoE write

class EcWrapperPythonTypes.DN_EC_T_MBX_DATA_COE
    bCompleteAccess: bool
        Complete access

    bySubIndex: int
        Object subindex

    wIndex: int
        Object index

    wStationAddress: int
        Station address of the slave

class EcWrapperPythonTypes.DN_EC_T_MBX_DATA_FOE
    dwBusyDone: int
        0 ... dwBusyEntire

        Type
        [out] If slave is busy

    dwBusyEntire: int
        Slave is busy
```

Type

[out] If dwBusyEntire > 0

dwFileSize: int

[out] File size

dwRequestedBytes: int

[out] Amount of bytes to be provided by the application

dwTransferredBytes: int

[out] Amount of transferred bytes

szBusyComment: str

[out] Busy Comment from slave

wStationAddress: int

[out] Station address of the slave

class `EcWrapperPythonTypes.DN_EC_T_MBX_DATA_FOE_REQ`**dwPassword: int**

[out] FoE read/write request password

szFileName: str

[out] Name of the file to be read/write

wStationAddress: int

[out] Station address of the slave

class `EcWrapperPythonTypes.DN_EC_T_MBX_PARMS`**Coe: *EcWrapperPythonTypes.DN_EC_T_MBX_PARMS_COE***

[in] CoE parameters

Foe: *EcWrapperPythonTypes.DN_EC_T_MBX_PARMS_FOE*

[in] FoE parameters

dwMemoryPoolSize: int

dwMemoryPoolSize

class `EcWrapperPythonTypes.DN_EC_T_MBX_PARMS_COE`**bDisableNotifications: bool**

[in] Disable all CoE related EC_NOTIFY_MBOXRCV notifications

bDisableODStorage: bool

[in] Disable storage of CoE objects in the internal object dictionary

class `EcWrapperPythonTypes.DN_EC_T_MBX_PARMS_FOE`**bDisableFileStorage: bool**

[in] Disable storage of FoE transfers as a file on the file system

bDisableNotifications: bool

[in] Disable all FoE related EC_NOTIFY_MBOXRCV notifications

dwMaxQueuedMbxTransfers: int

defaults to 32.

Type

[in] Maximum number of queued single FoE mailbox transfers that be used as a file write buffer. 0

class `EcWrapperPythonTypes.DN_EC_T_MONITOR_INIT_PARMS`**MbxParams: *EcWrapperPythonTypes.DN_EC_T_MBX_PARMS***

[in] Mailbox monitoring parameters

PerfMeasInternalParams:***EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_INTERNAL_PARMS***

[in] Internal performance measurement parameters

WorkerThreadParms: *EcWrapperPythonTypes.DN_EC_T_WORKER_THREAD_PARMS*
[in] Internal worker thread parameters

bApiLockByApp: **bool**
application is responsible for locking.

Type

[in] Lock pending API against `emonDeinitMonitor()`. `EC_FALSE` (default)

Type

locked internally. `EC_TRUE`

bProcessRestructuredCyclicCmds: **bool**
[in] Support processing of restructured cyclic command layout. All cyclic commands are processed as long as they are within the process data boundary of the ENI.

dwBacktraceFrames: **int**
dwBacktraceFrames

dwBusCycleTimeNsec: **int**
[in] Bus cycle time [ns]

dwCommunicationTimeoutMsec: **int**
disables monitoring.

Type

[in] Timeout [ms] for communication on the Ethernet TAP. 0

Type

defaults to 3 sec, `EC_WAITINFINITE`

dwLogLevel: *EcWrapperPythonTypes.DN_EC_LOG_LEVEL*
dwLogLevel

dwMaxBusSlaves: **int**
[in] Maximum pre-allocated bus slave objects

eEthTapType: *EcWrapperPythonTypes.DN_EC_T_ETHERNET_TAP_TYPE*
[in] Type of Ethernet TAP

oLinkParms: *EcWrapperPythonTypes.DN_EC_T_LINK_PARMS*
oLinkParms

szFileStoragePath: **str**
defaults to “.”

Type

[in] Path used to store records and files, e.g. FoE transfers. `EC_NULL`

wMainTapPortIn: **int**
X1.0 = 10, X1.1 = 11, X2.0 = 20, ..., X4.1 = 41. For other supported TAPs, this is the port number starting from 0. If both `wMainTapPortIn` and `wMainTapPortOut` are 0, no filtering is performed

Type

[in] TAP IN port. For ET2000

wMainTapPortOut: **int**
[in] TAP OUT port. See `wMainTapPortIn`

class `EcWrapperPythonTypes.DN_EC_T_MONITOR_STATUS`

bNextCyclicEntryReceived: **bool**
[out] Indicates whether all frames from the next EtherCAT cycle have been received and have not yet been processed

bNextFramesReceived: **bool**
[out] Indicates whether further unprocessed frames from the next EtherCAT cycle were received

bPdCommunicationDetected: **bool**
[out] Process Data communication detected

```

bPdoAssigned: bool
    [out] PDO assignment of all slaves detected and validated

bPdoMapped: bool
    [out] PDO mapping (SyncSM/FMMU) of all slaves detected and validated

bTopologyKnown: bool
    [out] Topology successfully detected and validated

dwCyclesProcessed: int
    [out] Number of EtherCAT cycles processed

eEthTapType: EcWrapperPythonTypes.DN_EC_T_ETHERNET_TAP_TYPE
    [out] Type of Ethernet TAP. Detected TAP if EC_T_MONITOR_INIT_PARMS::eEthTapType = eEth-
    Tap_AutoDetect is set.

wDeviceStatesDetected: int
    [out] Bit mask that indicates which slave states already detected/monitored, see /ref DEVICE_STATES

wEthTapPositionAutoIncAddr: int
    [out] Ethernet tap position as auto increment address

class EcWrapperPythonTypes.DN_EC_T_MSU_INFO
    dwBitOffsIn: int
        [out] Process Data Image INPUTs bit offset

    dwBitOffsOut: int
        [out] Process Data Image OUTPUTs bit offset

    dwBitSizeIn: int
        [out] Process Data Image INPUTs bit length

    dwBitSizeOut: int
        [out] Process Data Image OUTPUTs bit length

    wMsuId: int
        Slave/ProcessData/RxPdo[1..4]@Su,      Slave/ProcessData/TxPdo[1..4]@Su,      comment      at
        Cyclic/Frame/Cmd)

        Type
            [out] Master Sync Unit ID (ENI)

    wWkcStateDiagOffsIn: int
        0 = Process Data valid, 1 = Process Data invalid)

        Type
            [out] INPUTs WkcState bit offset in Diagnosis Image. (Bit values

    wWkcStateDiagOffsOut: int
        0 = Process Data valid, 1 = Process Data invalid)

        Type
            [out] OUTPUTs WkcState bit offset in Diagnosis Image. (Bit values

class EcWrapperPythonTypes.DN_EC_T_MULTIPLIER_TYPE (value) Cu2508 = 0
    Beckhoff CU2508 port multiplier

    Et2000 = 1
        Beckhoff ET2000 industrial Ethernet multi-channel probe

class EcWrapperPythonTypes.DN_EC_T_NOTIFYPARMS
    dwInBufSize: int
        [in] Size of notification input parameters in bytes

    dwOutBufSize: int
        [in] Size of buffer at pbyOutBuf in bytes

```

pCallerData: object

[in] Client depending caller data parameter. This pointer is one of the parameters when the client registers.

pbyInBuf: object

[in] Notification input parameters

pbyOutBuf: object

[out] Buffer for notification output (result)

pdwNumOutData: int

amount not set by notification.

Type

[out] Amount of bytes written to pbyOutBuf by notification. EC_NULL

class EcWrapperPythonTypes.DN_EC_T_OBJ2002

Object 0x2002 Bus Diagnosis Object

dwAcyclicDatagrams: int

dwAcyclicDatagrams

dwAcyclicFrames: int

dwAcyclicFrames

dwCRC32ConfigChecksum: int

dwCRC32ConfigChecksum

dwClearCounters: int

dwClearCounters

dwCyclicDatagrams: int

dwCyclicDatagrams

dwCyclicFrames: int

dwCyclicFrames

dwLostFrames: int

dwLostFrames

dwNumCfgSlaves: int

dwNumCfgSlaves

dwNumDCSlavesFound: int

dwNumDCSlavesFound

dwNumMbxSlaves: int

dwNumMbxSlaves

dwNumSlavesFound: int

dwNumSlavesFound

dwRXFrames: int

dwRXFrames

dwTXFrames: int

dwTXFrames

wSubIndex0: int

wSubIndex0

class EcWrapperPythonTypes.DN_EC_T_OBJ2003

Object 0x2003 Redundancy Diagnosis Object

byLineBreak: int

byLineBreak

byRedEnabled: int

byRedEnabled

```
wNumOfMainSlaves: int
    wNumOfMainSlaves

wNumOfRedSlaves: int
    wNumOfRedSlaves

wSubIndex0: int
    wSubIndex0

class EcWrapperPythonTypes.DN_EC_T_OBJ2005
    Object 0x2005 MAC Address Object

    abyCfgDestination: List[int]
        abyCfgDestination

    abyCfgSource: List[int]
        abyCfgSource

    abyHardware: List[int]
        abyHardware

    abyRedHardware: List[int]
        abyRedHardware

    wSubIndex0: int
        wSubIndex0

class EcWrapperPythonTypes.DN_EC_T_OBJ2020
    Object 0x2020 Master Initialization parameters

    bAllocSendFrameActive: bool
        bAllocSendFrameActive

    bPollingModeActive: bool
        bPollingModeActive

    dwApplicationVersion: int
        dwApplicationVersion

    dwBusCycleTimeUsec: int
        dwBusCycleTimeUsec

    dwEcatCmdMaxRetries: int
        dwEcatCmdMaxRetries

    dwEcatCmdTimeout: int
        dwEcatCmdTimeout

    dwEoeTimeout: int
        dwEoeTimeout

    dwFoeBusyTimeout: int
        dwFoeBusyTimeout

    dwMasterVersion: int
        dwMasterVersion

    dwMaxAcycCmdsPerFrame: int
        dwMaxAcycCmdsPerFrame

    dwMaxAcycFramesQueued: int
        dwMaxAcycFramesQueued

    dwMaxBusSlave: int
        dwMaxBusSlave

    dwMaxSlavesProcessedPerCycle: int
        dwMaxSlavesProcessedPerCycle
```

```
    dwStateChangeDebug: int
        dwStateChangeDebug

    szDriverIdent: str
        szDriverIdent

    wSubIndex0: int
        wSubIndex0

class EcWrapperPythonTypes.DN_EC_T_OBJ2200
    Object 0x2200 Bus Load Statistics

    dwBytesPerCycleAct: int
        dwBytesPerCycleAct

    dwBytesPerCycleMax: int
        dwBytesPerCycleMax

    dwBytesPerCycleMin: int
        dwBytesPerCycleMin

    dwBytesPerSecondAct: int
        dwBytesPerSecondAct

    dwBytesPerSecondMax: int
        dwBytesPerSecondMax

    dwBytesPerSecondMin: int
        dwBytesPerSecondMin

    wClearCounters: int
        wClearCounters

    wSubIndex0: int
        wSubIndex0

class EcWrapperPythonTypes.DN_EC_T_OBJ3XXX
    Object 0x3000 - 0x3FFF Slave Objects

    bDC64Support: bool
        bDC64Support

    bDCSupport: bool
        bDCSupport

    bEnableLinkMsgs: bool
        bEnableLinkMsgs

    bEntryValid: bool
        bEntryValid

    bErrFlagSet: bool
        bErrFlagSet

    bMailboxSupport: bool
        bMailboxSupport

    bSlaveIsOptional: bool
        bSlaveIsOptional

    bSlaveIsPresent: bool
        bSlaveIsPresent

    bSyncPulseActive: bool
        bSyncPulseActive

    byESCType: int
        byESCType
```

```
byEcatProcUnitErrorCounter: int  
    byEcatProcUnitErrorCounter  
  
byFmmusSupported: int  
    byFmmusSupported  
  
byFwdRxErrorCounter0: int  
    byFwdRxErrorCounter0  
  
byFwdRxErrorCounter1: int  
    byFwdRxErrorCounter1  
  
byFwdRxErrorCounter2: int  
    byFwdRxErrorCounter2  
  
byFwdRxErrorCounter3: int  
    byFwdRxErrorCounter3  
  
byLostLinkCounter0: int  
    byLostLinkCounter0  
  
byLostLinkCounter1: int  
    byLostLinkCounter1  
  
byLostLinkCounter2: int  
    byLostLinkCounter2  
  
byLostLinkCounter3: int  
    byLostLinkCounter3  
  
byPDIErrorCounter: int  
    byPDIErrorCounter  
  
byPortDescriptor: int  
    byPortDescriptor  
  
byRamSizeKb: int  
    byRamSizeKb  
  
bySyncManagersSupported: int  
    bySyncManagersSupported  
  
dwCurState: int  
    dwCurState  
  
dwDCSync0Period: int  
    dwDCSync0Period  
  
dwDCSync1Period: int  
    dwDCSync1Period  
  
dwErrorCode: int  
    dwErrorCode  
  
dwHotConnectGroupId: int  
    dwHotConnectGroupId  
  
dwPdOffsIn: int  
    dwPdOffsIn  
  
dwPdOffsOut: int  
    dwPdOffsOut  
  
dwPdSizeIn: int  
    dwPdSizeIn  
  
dwPdSizeOut: int  
    dwPdSizeOut
```

```
    dwProductCode: int
        dwProductCode

    dwReqState: int
        dwReqState

    dwRevisionNo: int
        dwRevisionNo

    dwSBErrorCode: int
        dwSBErrorCode

    dwSerialNo: int
        dwSerialNo

    dwSystemTimeDifference: int
        dwSystemTimeDifference

    dwVendorID: int
        dwVendorID

    szDeviceName: str
        szDeviceName

    wAliasAddr: int
        wAliasAddr

    wAutoIncAddr: int
        wAutoIncAddr

    wConfigPhysAddr: int
        wConfigPhysAddr

    wMbxSupportedProtocols: int
        wMbxSupportedProtocols

    wPhysAddr: int
        wPhysAddr

    wPortState: int
        wPortState

    wRxErrorCounter0: int
        wRxErrorCounter0

    wRxErrorCounter1: int
        wRxErrorCounter1

    wRxErrorCounter2: int
        wRxErrorCounter2

    wRxErrorCounter3: int
        wRxErrorCounter3

    wSubIndex0: int
        wSubIndex0

class EcWrapperPythonTypes.DN_EC_T_OBJ8XXX
    Object 0x8000 - 0x8FFF Slave Objects (configured slaves) 'Modular Device Profiles'

    byFlags: int
        byFlags

    byLinkPreset: int
        byLinkPreset

    dwDeviceType: int
        dwDeviceType
```

```
dwProductCode: int  
    dwProductCode  
  
dwRevision: int  
    dwRevision  
  
dwSerial: int  
    dwSerial  
  
dwVendorID: int  
    dwVendorID  
  
szName: str  
    szName  
  
szType: str  
    szType  
  
wFixedStationAddr: int  
    wFixedStationAddr  
  
wMailboxInSize: int  
    wMailboxInSize  
  
wMailboxOutSize: int  
    wMailboxOutSize  
  
wSubIndex0: int  
    wSubIndex0
```

```
class EcWrapperPythonTypes.DN_EC_T_OBJ9XXX  
    Modular Device Profiles: EtherCAT Master - internal slave object element (connected slaves)
```

```
dwProductCode: int  
    dwProductCode  
  
dwRevision: int  
    dwRevision  
  
dwSerial: int  
    dwSerial  
  
dwVendorID: int  
    dwVendorID  
  
wDLStatus: int  
    wDLStatus  
  
wFixedStationAddr: int  
    wFixedStationAddr  
  
wSubIndex0: int  
    wSubIndex0
```

```
class EcWrapperPythonTypes.DN_EC_T_OBJAXXX  
    Modular Device Profiles: EtherCAT Master - internal slave object element (slave diagnosis)
```

```
bDisableAutomaticLinkControl: int  
    bDisableAutomaticLinkControl  
  
byLinkConnStatus: int  
    byLinkConnStatus  
  
byLinkControl: int  
    byLinkControl  
  
dwAbnormalStateCounter: int  
    dwAbnormalStateCounter
```

```
    dwCRCErrrorCounterPort0: int
        dwCRCErrrorCounterPort0

    dwCRCErrrorCounterPort1: int
        dwCRCErrrorCounterPort1

    dwCRCErrrorCounterPort2: int
        dwCRCErrrorCounterPort2

    dwCRCErrrorCounterPort3: int
        dwCRCErrrorCounterPort3

    dwCyclicWCErrrorCounter: int
        dwCyclicWCErrrorCounter

    dwSlaveNotPresentCounter: int
        dwSlaveNotPresentCounter

    wALControl: int
        wALControl

    wALStatus: int
        wALStatus

    wFixedAddressConnPort0: int
        wFixedAddressConnPort0

    wFixedAddressConnPort1: int
        wFixedAddressConnPort1

    wFixedAddressConnPort2: int
        wFixedAddressConnPort2

    wFixedAddressConnPort3: int
        wFixedAddressConnPort3

    wLastALStatusCode: int
        wLastALStatusCode

    wSubIndex0: int
        wSubIndex0

class EcWrapperPythonTypes.DN_EC_T_OBJF000
    Modular Device Profiles: EtherCAT Master - modular device profile

    dwGeneralCfg: int
        dwGeneralCfg

    dwGeneralInfo: int
        dwGeneralInfo

    wIndexDistance: int
        wIndexDistance

    wMaxModuleCnt: int
        wMaxModuleCnt

    wSubIndex0: int
        wSubIndex0

class EcWrapperPythonTypes.DN_EC_T_OBJF02X
    Modular Device Profiles: EtherCAT Master - configured address list

    awStationAddr: List[int]
        awStationAddr

    wSubIndex0: int
        wSubIndex0
```

```

class EcWrapperPythonTypes.DN_EC_T_OBJF04X
    Modular Device Profiles: EtherCAT Master - detected address list

    awStationAddr: List[int]
        awStationAddr

    wSubIndex0: int
        wSubIndex0

class EcWrapperPythonTypes.DN_EC_T_PACKETCAPTURE_INFO
    dwCyclesProcessed: int
        [out] Number of EtherCAT cycles processed

    eStatus: EcWrapperPythonTypes.DN_EC_T_PACKETCAPTURE_STATUS
        [out] Status of packet capture processing

    qwBytesProcessed: int
        [out] Number of processed bytes from the current packet capture file

    qwFileSize: int
        [out] File size[bytes] of the current packet capture

    qwFrameNumberCur: int
        [out] Last processed frame number from the current packet capture file

    qwFrameNumberTotal: int
        [out] Total number of processed frames from all capture files

    qwTimeStamp: int
        [out] Time stamp [ns] of the last processed frame from the current packet capture file

    szFileName: str
        [out] File name of current processed capture

class EcWrapperPythonTypes.DN_EC_T_PACKETCAPTURE_PARMS
    bReadMultipleFiles: bool
        bReadMultipleFiles

    dwMaxFileSize: int
        [in] Creates a new file every time the number of bytes written exceeds this limit. Disabled with a value
        set to 0.

    dwMaxFrameCnt: int
        [in] Creates a new file every time the number of frames written exceeds this limit. Disabled with a value
        set to 0.

    dwRingBufferFileCnt: int
        dwRingBufferFileCnt

    szFileName: str
        [in] File name. Supported formats are .pcap or .pcapng.

class EcWrapperPythonTypes.DN_EC_T_PACKETCAPTURE_STATUS (value)
    Dummy = 4294

    Dummy

    Finished = 3
        Packet capture processing finished

    NotLoaded = 1
        No packet capture loaded

    Running = 2
        Packet capture processing running

    Unknown = 0
        Unknown packet capture status
  
```

```

class EcWrapperPythonTypes.DN_EC_T_PDIWATCHDOG_DESC
    PDI Watchdog expired

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

class EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_APP_PARMS

class EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_COUNTER_PARMS
    qwFrequency: int
        [in] Frequency in Hz used by the timer in GetCounterTicks

class EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_FLAG (value)
    An enumeration.

    LONG_TIMER = 2

    OFFSET = 1

class EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_HISTOGRAM
    aBinsObj: List[int]
        aBinsObj

    dwBinCount: int
        length of aBins

    qwMaxTicks: int
        Results above qwMaxTicks are stored in the last bin

    qwMinTicks: int
        Results below qwMinTicks are stored in the first bin

class EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_HISTOGRAM_PARMS
    dwBinCount: int
        [in] Amount of bins to use for the histogram

    qwMaxTicks: int
        [in] Results above qwMaxTicks are stored in the last bin

    qwMinTicks: int
        [in] Results below qwMinTicks are stored in the first bin

class EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_INFO
    dwBinCountHistogram: int
        length of Histogram Bins

    dwFlags: int
        Flags associated with the benchmark (See EC_T_PERF_MEAS_FLAG...)

    eUserJob: EcWrapperPythonTypes.DN_EC_T_USER_JOB
        UserJob associated with the benchmark

    qwFrequency: int
        Frequency in Hz used by the timer

    szName: str
        Name of the benchmark

class EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_INFO_PARMS
    dwFlags: int
        [in] Flags associated with the benchmark (See EC_T_PERF_MEAS_FLAG...)

    szName: str
        [in] Performance counter name

class EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_INTERNAL_PARMS
    CounterParms: EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_COUNTER_PARMS
        [in] Timer function settings. When not provided OsMeasGetCounterTicks is used.

```

HistogramParms: *EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_HISTOGRAM_PARMS*
 [in] Histogram settings. When not provided the histogram is disabled.

bEnabled: bool
 [in] Enable/disable internal performance counters

class *EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_VAL*

qwAvgTicks: int
 [ticks]

qwCurrTicks: int
 [ticks]

qwMaxTicks: int
 [ticks]

qwMinTicks: int
 [ticks]

class *EcWrapperPythonTypes.DN_EC_T_PORT_OPERATION_NOTIFY_DESC*

SlaveProp: *EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP*
 Slave properties

dwResult: int
 Result of request

dwTferId: int
 Transfer ID. For every new port operation a unique ID has to be assigned. This ID can be used after completion to identify the transfer.

wPortStateNew: int
 New state of the slave ports

wPortStateOld: int
 Old state of the slave ports

class *EcWrapperPythonTypes.DN_EC_T_PROCESS_VAR_INFO*

emGetSlaveInpVarInfo / emGetSlaveOutpVarInfo

bIsInputData: bool
 [out] Determines whether the found process variable is an input variable or an output variable

nBitOffs: int
 [out] Bit offset in the process data image

nBitSize: int
 [out] Size in bits of the found process variable

szName: str
 [out] Name of the found process variable

wDataType: int
 [out] Data type of the found process variable (according to ETG.1000, section 5). See also *EcType.h*, *DEFTYPE_BOOLEAN*.

wFixedAddr: int
 [out] Station address of the slave that is owner of this variable

class *EcWrapperPythonTypes.DN_EC_T_PROCESS_VAR_INFO_EX*

CopyInfo: *EcWrapperPythonTypes.DN_EC_T_CYC_COPY_INFO*
 [out] Copy Info if applied to the variable

bIsInputData: bool
 [out] Determines whether the found process variable is an input variable or an output variable

dwTaskId: int
 [out] ID of task where process variable is located

```

nBitOffs: int
    [out] Bit offset in the process data image

nBitSize: int
    [out] Size in bits of the found process variable

szName: str
    [out] Name of the found process variable

wDataType: int
    [out] Data type of the found process variable (according to ETG.1000, section 5). See also EcType.h,
    DEFTYPE_BOOLEAN.

wFixedAddr: int
    [out] Station address of the slave that is owner of this variable

wIndex: int
    [out] Object index

wMasterSyncUnit: int
    Slave/ProcessData/RxPdo[1..4]@Su,      Slave/ProcessData/TxPdo[1..4]@Su,      comment      at
    Cyclic/Frame/Cmd

    Type
        [out] Master Sync Unit ID (ENI)

wPdoIndex: int
    [out] Index of PDO (process data object)

wRes1: int
    wRes1

wSubIndex: int
    [out] Object sub index

wWkcStateDiagOffs: int
    [out] Bit offset in the diagnostic image (API GetDiagnosisImagePtr)

class EcWrapperPythonTypes.DN_EC_T_PROFILE_CHANNEL_INFO
    szDisplayName: str
        [out] Display name

    wAddInfo: int
        'high word of CoE object 0x1000'

        Type
            [out] AddInfo

    wProfileNo: int
        'low word of CoE object 0x1000'

        Type
            [out] ProfileNo

class EcWrapperPythonTypes.DN_EC_T_PTS_SRV_START_PARMS
    Start parameter for the Pass-Through-Server

    dwPtsThreadPriority: int
        dwPtsThreadPriority

    oIpAddr: EcWrapperPythonTypes.DN_EC_T_IPADDR
        oIpAddr

    wPort: int
        wPort

class EcWrapperPythonTypes.DN_EC_T_RAS_CONNECTION_INFO
    dwAccessControlActive: int
        dwAccessControlActive

```

```
    dwAccessLevel: int
        dwAccessLevel

class EcWrapperPythonTypes.DN_EC_T_RAS_EXCEPTIONDESC

class EcWrapperPythonTypes.DN_EC_T_RAWCMDRESPONSE_NTIFY_DESC
    dwAddr: int
        [in] Address Field

    dwCmdIdx: int
        [in] Command Index Field

    dwInvokeId: int
        [in] Invoke Id from callee. Only lower 16 bits are relevant

    dwLength: int
        [in] Length of data portion (11 relevant bits)

    dwResult: int
        [in] EC_E_NOERROR on success, error code otherwise

    dwWkc: int
        [in] Received working counter

    pbyData: object
        [in] Pointer to data portion within a PDU. The callback function has to store the data into application
        memory, the data pointer will be invalid after returning from the callback.

class EcWrapperPythonTypes.DN_EC_T_REDUNDANCY_DIAGNOSIS_INFO
    See also Object 0x2003 Redundancy Diagnosis Object

    bLineBreakDetected: bool
        Line Break Detected

    bRedEnabled: bool
        Cable Redundancy Enabled

    dwMainSlaveCnt: int
        Main Line Slave Count

    dwRedSlaveCnt: int
        Red Line Slave Count

    dwRes: List[int]
        dwRes

class EcWrapperPythonTypes.DN_EC_T_REFCLKLOCK_PRESENCE_NTIFY_DESC
    Distributed clocks (DC) Reference Clock Presence

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        [in] Slave properties

    bPresent: bool
        [in] Reference clock present

class EcWrapperPythonTypes.DN_EC_T_REGISTERRESULTS
    Client register result

    dwClntId: int
        [out] Client ID

    dwPDInSize: int
        [out] Size of process data input memory (in bytes)

    dwPDOuSize: int
        [out] Size of process data output memory (in bytes)

    pbyPDIn: object
        [out] Pointer to process data input memory
```

```
pbyPDDOut: object  
    [out] Pointer to process data output memory  
  
class EcWrapperPythonTypes.DN_EC_T_RELEASE_FORCED_PROCESSDATA_NOTIFY_DESC  
    bOutput: bool  
        Input Bits  
        Type  
            EC_TRUE  
        Type  
            Output Bits, EC_FALSE  
  
    dwOffset: int  
        Offset of the forced Bits  
  
    wBitLength: int  
        Bit length  
  
class EcWrapperPythonTypes.DN_EC_T_S2SMBX_ERROR_DESC  
    S2S Mailbox Error  
  
    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP  
        Slave properties of requesting slave  
  
    dwErrorCode: int  
        Error code  
  
    wTargetFixedAddress: int  
        Fixed address of the target slave  
  
class EcWrapperPythonTypes.DN_EC_T_SB_MISMATCH_DESC  
    bIdentificationError: bool  
        [in] Identification command sent to slave but failed  
  
    dwBusProdCode: int  
        [in] Unexpected slave (bus) product code  
  
    dwBusRevisionNo: int  
        [in] Unexpected slave (bus) revision number  
  
    dwBusSerialNo: int  
        [in] Unexpected slave (bus) serial number  
  
    dwBusVendorId: int  
        [in] Unexpected slave (bus) vendor ID  
  
    dwCfgProdCode: int  
        [in] Missing slave (config) Product code  
  
    dwCfgRevisionNo: int  
        [in] Missing slave (config) Revision Number  
  
    dwCfgSerialNo: int  
        [in] Missing slave (config) Serial Number  
  
    dwCfgVendorId: int  
        [in] Missing slave (config) Vendor ID  
  
    wBusAIncAddress: int  
        [in] Unexpected slave (bus) auto-inc address  
  
    wBusFixedAddress: int  
        [in] Unexpected slave (bus) station address  
  
    wCfgAIncAddress: int  
        [in] Missing slave (config) Auto-Increment Address
```

```

wCfgFixedAddress: int
    [in] Missing slave (config) station Address

wIdentificationAdo: int
    [in] Identification register

wIdentificationVal: int
    [in] Last identification value read from slave according to the last used identification method

wIdentificationValExpected: int
    [in] Identification expected value

wPrevAIncAddress: int
    [in] Previous slave auto-increment address

wPrevFixedAddress: int
    [in] Previous slave station address

wPrevPort: int
    [in] Previous slave station address

class EcWrapperPythonTypes.DN_EC_T_SB_SLAVEINFO_DESC
    dwProductCode: int
        [out] Product Code stored in the EEPROM at offset 0x000A

    dwRevisionNumber: int
        [out] Revision number stored in the EEPROM at offset 0x000C (Not read by default!)

    dwScanBusStatus: int
        [out] Scan bus status (determined in the latest scan bus) emNotify EC_NOTIFY_SB_STATUS

    dwSerialNumber: int
        [out] Serial number stored in the EEPROM at offset 0x000E (Not read by default!)

    dwVendorId: int
        [out] Vendor Identification stored in the EEPROM at offset 0x0008

class EcWrapperPythonTypes.DN_EC_T_SB_SLAVEINFO_REQ_DESC
    eInfoEntry: EcWrapperPythonTypes.DN_EC_T_eINFOENTRY
        [in] Info Entry to read

    wAutoIncAddress: int
        [in] Auto-Increment address of the slave

class EcWrapperPythonTypes.DN_EC_T_SB_SLAVEINFO_RES_DESC
    dwInfoLength: int
        [in, out] Length of Info Field (buffer, actually read length)

    eInfoEntry: EcWrapperPythonTypes.DN_EC_T_eINFOENTRY
        [out] Info entry read

    pbyInfo: object
        [out] Pointer to Info (-1 if no info found in XML file)

class EcWrapperPythonTypes.DN_EC_T_SB_STATUS_NTIFY_DESC
    dwResultCode: int
        no bus scan executed, #EC_E_BUSCONFIG_MISMATCH: bus configuration mismatch result of scan-
        bus

        Type
            [in] #EC_E_NOERROR

        Type
            success, #EC_E_NOTREADY

    dwSlaveCount: int
        [in] Number of slaves connected to the bus

class EcWrapperPythonTypes.DN_EC_T_SELFTESTSCAN_PARMS

```

```

bDetectBadConnections: bool
    [in] Execute the bad connection detection after self-test

bMeasureRoundtripTimeForSingleFrame: bool
    [in] Execute roundtrip time calculation for single frame

dwFrameCount: int
    [in] Total number of frames sent during the self-test. Default value is 1500. A value of 0 sets the default
    value.

dwFrameSizeMax: int
    [in] Max frame size [bytes]. Default value is 1514. A value of 0 sets the default value.

dwFrameSizeMin: int
    [in] Min frame size [bytes]. Default value is 60. A value of 0 sets the default value.

dwFrameSizeStep: int
    [in] Size [bytes] by which the frame increases or decreases continuously during the self-test. Default
    value is 1. A value of 0 sets the default value.

dwTimeout: int
    [in] Timeout [ms], 0 or EC_NOWAIT defaults to 500ms

qwFrameRoundtripTimeAvg: int
    [out] Roundtrip time average [us]. Time taken from sending to receiving the frame (master application
    level).

qwFrameRoundtripTimeMax: int
    [out] Roundtrip time maximum [us]. Time taken from sending to receiving the frame (master application
    level).

qwFrameRoundtripTimeMin: int
    [out] Roundtrip time minimum [us]. Time taken from sending to receiving the frame (master application
    level).

class EcWrapperPythonTypes.DN_EC_T_SIMULATOR_DEVICE_CONNECTION_DESC
    byPort: int
        Port A-D

        Type
            0...3

    dwInstanceID: int
        EC-Simulator Instance ID

    dwType: int
        EC_SIMULATOR_DEVICE_CONNECTION_TYPE...

    wCfgFixedAddress: int
        EC-Simulator Configuration (ENI/EXI)

class EcWrapperPythonTypes.DN_EC_T_SIMULATOR_INIT_PARMS
    PerfMeasInternalParms:
        EcWrapperPythonTypes.DN_EC_T_PERF_MEAS_INTERNAL_PARMS
        [in] Internal performance measurement parameters

    aoLinkParms: List[EcWrapperPythonTypes.DN_EC_T_LINK_PARMS]
        aoLinkParms

    bApiLockByApp: bool
        application is responsible for locking.

        Type
            [in] Lock pending API against esDeinitSimulator(). EC_FALSE (default)

        Type
            locked internally. EC_TRUE

```

```

bConnectHcGroups: bool
    [in] Connect hot connect groups in topology (floating group heads to free ports)

bDisableProcessDataImage: bool
    [in] Disable Process Data Image (Process Data Variables not supported)

dwBusCycleTimeNsec: int
    [in] Bus cycle time in nanoseconds [ns]

dwLogLevel: EcWrapperPythonTypes.DN_EC_LOG_LEVEL
    dwLogLevel

dwSimulatorAddress: int
    [in] Reserved

oDeviceConnection:
    EcWrapperPythonTypes.DN_EC_T_SIMULATOR_DEVICE_CONNECTION_DESC
    oDeviceConnection

class EcWrapperPythonTypes.DN_EC_T_SIM_SLAVE_CFG_INFO
    aPortConnection:
    List [EcWrapperPythonTypes.DN_EC_T_SIMULATOR_DEVICE_CONNECTION_DESC]
        /EtherCATConfig/ExtendedConfig/Slaves/Slave/PortConnection)

        Type
            [out] Explicit port connection (optional) (EXI)

    bIgnoreCoeDownloadError: bool
        /EtherCATConfig/ExtendedConfig/Slaves/Slave/Mailbox/CoE@IgnoreDownloadError)

        Type
            [out] Slave ignores CoE download errors (EXI)

    bPowerOff: bool
        /EtherCATConfig/ExtendedConfig/Slaves/Slave@PowerOff)

        Type
            [out] Slave powered off on startup (EXI)

    bSimulated: bool
        /EtherCATConfig/ExtendedConfig/Slaves/Slave/Simulated)

        Type
            [out] Slave is simulated (EXI)

    szApplicationName: str
        /EtherCATConfig/ExtendedConfig/Slaves/Slave/Application/Name)

        Type
            [out] Configured slave application name (EXI)

    szApplicationParms: str
        /EtherCATConfig/ExtendedConfig/Slaves/Slave/Application/Parameter)

        Type
            [out] Configured slave application parameters (EXI)

    wFixedAddress: int
        /EtherCATConfig/Config/Slave/Info/PhysAddr)

        Type
            [out] Slave's station address (ENI/EXI)

class EcWrapperPythonTypes.DN_EC_T_SIM_SLAVE_INFO
    oCfg: EcWrapperPythonTypes.DN_EC_T_SIM_SLAVE_CFG_INFO
        [out] Config Info (ENI/EXI)

    oStatus: EcWrapperPythonTypes.DN_EC_T_SIM_SLAVE_STATUS_INFO
        [out] Status Info

```

```

class EcWrapperPythonTypes.DN_EC_T_SIM_SLAVE_STATUS_INFO
    aPortConnection:
    List [EcWrapperPythonTypes.DN_EC_T_SIMULATOR_DEVICE_CONNECTION_DESC]
        [out] See esConnectPorts()

    bIsPowerOn: bool
        [out] Slave is powered on. See esPowerSlave().

    bIsPresent: bool
        [out] Slave is present in topology segment and connected to the network. See esConnectPorts(), Hot
        Connect.

    wAlStatusCodeReq: int
        [out] AL Status (0x0134) requested from Simulator application. See esSetSimSlaveState().

    wAlStatusReq: int
        [out] AL Status (0x0130) requested from Simulator application. See esSetSimSlaveState().

class EcWrapperPythonTypes.DN_EC_T_SLAVEREGISTER_TRANSFER_NTIFY_DESC
    bRead: bool
        Write register transfer

        Type
            EC_TRUE

        Type
            Read register, EC_FALSE

    dwResult: int
        Result of Slave register transfer

    dwTferId: int
        Transfer ID. For every new slave register transfer a unique ID has to be assigned. This ID can be used
        after completion to identify the transfer.

    pbyData: object
        Pointer to the data read

    wFixedAddr: int
        Station address of slave

    wLen: int
        Length of slave register transfer

    wRegisterOffset: int
        Register offset

    wWkc: int
        Received working counter

class EcWrapperPythonTypes.DN_EC_T_SLAVES_ERROR_DESC
    Slaves error status descriptor

    SlaveError: List [EcWrapperPythonTypes.DN_EC_T_SLAVES_ERROR_DESC_ENTRY]
        Slave error descriptions

    wCount: int
        Number of slave errors

    wRes: int
        wRes

class EcWrapperPythonTypes.DN_EC_T_SLAVES_ERROR_DESC_ENTRY
    Slaves error status descriptor

    wRes: int
        wRes

```

```

wStationAddress: int
    Slave station address

wStatus: int
    Slave status (AL Status)

wStatusCode: int
    Slave status code (AL Control Status)

class EcWrapperPythonTypes.DN_EC_T_SLAVES_PRESENCE_NOTIFY_DESC
    SlavePresence:
    List [EcWrapperPythonTypes.DN_EC_T_SLAVE_PRESENCE_NOTIFY_DESC]
        Slave presence descriptions

    wCount: int
        Number of slave presence notifications

class EcWrapperPythonTypes.DN_EC_T_SLAVES_STATECHANGED_NOTIFY_DESC
    SlaveStates:
    List [EcWrapperPythonTypes.DN_EC_T_SLAVES_STATECHANGED_NOTIFY_DESC_ENTRY]
        Slave state changed descriptor

    wCount: int
        Number of slave state changes

class EcWrapperPythonTypes.DN_EC_T_SLAVES_STATECHANGED_NOTIFY_DESC_ENTRY
    Slaves state change descriptor

    byState: int
        New slave state

    wStationAddress: int
        Slave station address

class EcWrapperPythonTypes.DN_EC_T_SLAVES_UNEXPECTED_STATE_DESC
    Slaves in unexpected state descriptor

    SlaveStates:
    List [EcWrapperPythonTypes.DN_EC_T_SLAVES_UNEXPECTED_STATE_DESC_ENTRY]
        Slave state change descriptions

    wCount: int
        Number of unexpected slave state changes

    wRes: int
        wRes

class EcWrapperPythonTypes.DN_EC_T_SLAVES_UNEXPECTED_STATE_DESC_ENTRY
    Slaves in unexpected state descriptor

    curState: EcWrapperPythonTypes.DN_EC_T_STATE
        Current state

    expState: EcWrapperPythonTypes.DN_EC_T_STATE
        Expected state

    wStationAddress: int
        Slave station address

class EcWrapperPythonTypes.DN_EC_T_SLAVE_ERROR_INFO_DESC
    EtherCAT Slave error status info descriptor

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

    wStatus: int
        Slave Status (AL Status)

```

```

wStatusCode: int
    Error status code (AL STATUS CODE)

class EcWrapperPythonTypes.DN_EC_T_SLAVE_IDENTIFICATION_NOTIFY_DESC
    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

    dwResult: int
        Result of request

    dwTferId: int
        Transfer ID. For every new port operation a unique ID has to be assigned. This ID can be used after
        completion to identify the transfer.

    wAdo: int
        Slave address offset used for identification. Given by API.

    wValue: int
        Slave identification value. Given by API.

class EcWrapperPythonTypes.DN_EC_T_SLAVE_IDENT_TIMEOUT_DESC
    Slave identification timeout

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

    SlavePropParent: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties of parent slave

    wPortAtParent: int
        Port at parent slave

class EcWrapperPythonTypes.DN_EC_T_SLAVE_NOTSUPPORTED_DESC
    Slave not supported

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

class EcWrapperPythonTypes.DN_EC_T_SLAVE_PRESENCE_NOTIFY_DESC
    Slaves presence descriptor

    bPresent: int
        absent

        Type
            EC_TRUE

        Type
            present, EC_FALSE

    wStationAddress: int
        Slave station address

class EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
    EtherCAT slave properties

    achName: str
        Configured name of the slave device (NULL terminated string)

    wAutoIncAddr: int
        Configured auto increment address or INVALID_AUTO_INC_ADDR

    wStationAddress: int
        Configured station address or INVALID_FIXED_ADDR

class EcWrapperPythonTypes.DN_EC_T_SLAVE_SELECTION (value) eSlaveSelect
    eSlaveSelectionDummy

```

```

eSlaveSelectionMasterSyncUnit = 2
    [in] Select slave and his topological followers

eSlaveSelectionSingle = 0
    [in] Select only one slave

eSlaveSelectionTopoFollowers = 1
    [in] Select slave and his topological followers

class EcWrapperPythonTypes.DN_EC_T_SLAVE_STATECHANGED_NOTIFY_DESC
    Slave state change descriptor

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

    newState: EcWrapperPythonTypes.DN_EC_T_STATE
        New slave state

class EcWrapperPythonTypes.DN_EC_T_SLAVE_UNEXPECTED_STATE_DESC
    Slave in unexpected state descriptor

    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties

    curState: EcWrapperPythonTypes.DN_EC_T_STATE
        Current state

    expState: EcWrapperPythonTypes.DN_EC_T_STATE
        Expected state

class EcWrapperPythonTypes.DN_EC_T_SLVSTATISTICS_DESC
    abyFwdRxErrorCnt: List[int]
        [out] Forwarded RX Error Counters per Slave Port

    abyInvalidFrameCnt: List[int]
        [out] Invalid Frame Counters per Slave Port

    abyLostLinkCnt: List[int]
        [out] Lost Link Counters per Slave Port

    abyRxErrorCnt: List[int]
        [out] RX Error Counters per Slave Port

    byPdiErrorCnt: int
        [out] PDI Error Counter

    byProcessingUnitErrorCnt: int
        [out] Processing Unit Error Counter

    qwChangeTime: int
        [out] Timestamp of the last counter change [ns]

    qwReadTime: int
        [out] Timestamp of the last read [ns]

    wAlStatusCode: int
        [out] AL Status Code

class EcWrapperPythonTypes.DN_EC_T_SOE_EMERGENCY
    EtherCAT SoE emergency request

    abyData: List[int]
        Emergency error data

    wHeader: int
        SoE Header

    wStationAddress: int
        Station address of the slave initiated the emergency request
  
```

```

class EcWrapperPythonTypes.DN_EC_T_SOE_NOTIFICATION
    EtherCAT SoE notification

    abyData: List[int]
        Error data

    wHeader: int
        SoE Header

    wIdn: int
        IDN number

    wStationAddress: int
        Station address of the slave

class EcWrapperPythonTypes.DN_EC_T_STATE (value)
    EtherCAT state

    BCppDummy = 4294967295
        BCppDummy

    BOOTSTRAP = 3
        EtherCAT state BOOTSTRAP

    INIT = 1
        EtherCAT state INIT

    OP = 8
        EtherCAT state OP (operational)

    PREOP = 2
        EtherCAT state PREOP (pre-operational)

    SAFEOP = 4
        EtherCAT state SAFEOP (safe operational)

    UNKNOWN = 0
        Unknown state

class EcWrapperPythonTypes.DN_EC_T_STATECHANGE
    EtherCAT state change

    newState: EcWrapperPythonTypes.DN_EC_T_STATE
        New operational state

    oldState: EcWrapperPythonTypes.DN_EC_T_STATE
        Old operational state

class EcWrapperPythonTypes.DN_EC_T_STATISTIC
    See also Object 0x2006 Mailbox Statistics Object

    dwLast: int
        Last

    dwTotal: int
        Total

class EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER
    See also Object 0x2006 Mailbox Statistics Object

    Bytes: EcWrapperPythonTypes.DN_EC_T_STATISTIC
        Number of bytes transferred

    Cnt: EcWrapperPythonTypes.DN_EC_T_STATISTIC
        Number of transfers

class EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER_DUPLEX
    See also Object 0x2006 Mailbox Statistics Object
  
```

Read: `EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER`
 Number of read transfers

Write: `EcWrapperPythonTypes.DN_EC_T_STATISTIC_TRANSFER`
 Number of write transfers

class `EcWrapperPythonTypes.DN_EC_T_TAP_LINK_STATUS_NOTIFY_DESC`
bLinkConnected: `bool`
 Link status of EC-Monitor - Ethernet Tap connection

class `EcWrapperPythonTypes.DN_EC_T_TLS_CERT_TYPE (value)` **BCppDummy =**
 BCppDummy
Data = 2
 TLS certificate type is a buffer
Filename = 1
 TLS certificate type is a filename
Unknown = 0
 Unknown

class `EcWrapperPythonTypes.DN_EC_T_TLS_PRIVKEY_TYPE (value)` **BCppDummy =**
 BCppDummy
Data = 2
 TLS private key type is a buffer
Filename = 1
 TLS private key type is a filename
Unknown = 0
 Unknown

class `EcWrapperPythonTypes.DN_EC_T_TRACE_DATA_INFO`
dwOffset: `int`
 [out] Trace data offset in bytes
pbyData: `object`
 [out] Process data output buffer, containing trace data
wSize: `int`
 [out] Trace data size in bytes

class `EcWrapperPythonTypes.DN_EC_T_TX_PDO_NOTIFY_DESC`
dwNumber: `int`
 dwNumber
pbyData: `object`
 pbyData
wLen: `int`
 wLen
wPhysAddr: `int`
 wPhysAddr

class `EcWrapperPythonTypes.DN_EC_T_USER_JOB (value)`
 EtherCAT user controlled execution job
BCppDummy = 4294967295
 BCppDummy
MasterTimer = 4
 Run internal master and slave state machines for generic management

MasterTimerMinimal = 8

no state change possible

Type

Run minimal master timer routine

MonitorTimer = 33

Run EC-Monitor timer routine (generic management)

ProcessAcycRxFrames = 10

Receive frames and process received data related to acyclic frames

ProcessAllRxFrames = 1

Receive frames and process all received data. Polling mode only.

ProcessRxFramesByTaskId = 9

Cyclic/TaskId)

Type

Receive frames and process received data related to a specific task id (ENI

SendAcycFrames = 6

Send acyclic frames

SendAllCycFrames = 2

Send all cyclic frames

SendCycFramesByTaskId = 7

Cyclic/TaskId)

Type

Send cyclic frames related to a specific task id (ENI

SimulatorTimer = 32

Run EC-Simulator timer routine (generic management)

StampSendAllCycFrames = 22

Obsolete

StampSendCycFramesByTaskId = 27

Obsolete

StartTask = 12

Start new task

StopTask = 13

Stop currently running task

SwitchEoeFrames = 11

Switch queued EoE frames (see EC_IOCTL_SET_EOE_DEFERRED_SWITCHING_ENABLED)

Undefined = 0

Undefined

class EcWrapperPythonTypes.DN_EC_T_VAR_DIRECTION (*value*)

All = 3

INPUTs and OUTPUTs

BCppDummy = 4294967295

BCppDummy

INPUT = 1

INPUTs

OUTPUT = 2

OUTPUTs

Undefined = 0

Undefined Direction

```

class EcWrapperPythonTypes.DN_EC_T_VAR_SOURCE (value) All = 1
    Slaves and Master/Monitor/Simulator
    AllSlaves = 2
        All Slaves
    BCppDummy = 4294967295
        BCppDummy
    Master = 4
        Master
    Monitor = 5
        Monitor
    Simulator = 6
        Simulator
    Slave = 3
        Slave
    Undefined = 0
        Undefined Source

class EcWrapperPythonTypes.DN_EC_T_VLAN_MODE (value) Default = 0
    default Vlan tag for programmable Switch
    TailTag = 1
        Tail tag for Micrel KSZ8863

class EcWrapperPythonTypes.DN_EC_T_WKCERR_DESC
    EtherCAT command WKC error descriptor
    SlaveProp: EcWrapperPythonTypes.DN_EC_T_SLAVE_PROP
        Slave properties, content is undefined in case of cyclic WKC_ERROR
    byCmd: int
        EtherCAT command type
    byRsvd: List[int]
        byRsvd
    dwAddr: int
        Logical address or physical address (ADP/ADO)
    dwTaskId: int
        Cyclic/TaskId
        Type
            Cyclic Task ID (ENI)
    wMsuId: int
        Slave/ProcessData/RxPdo[1..4]@Su,      Slave/ProcessData/TxPdo[1..4]@Su,      comment      at
        Cyclic/Frame/Cmd)
        Type
            Master Sync Unit ID (ENI)
    wWkcAct: int
        Working counter actual value
    wWkcSet: int
        Working counter set value

class EcWrapperPythonTypes.DN_EC_T_WORKER_THREAD_PARMS
    cpuAffinityMask: int
        [in] CPU affinity to use for the worker thread

```

dwPrio: int
[in] Priority to use for the worker thread

class EcWrapperPythonTypes.DN_EC_T_eINFOENTRY (*value*)

BCppDummy =

BCppDummy

alias_address = 9
alias_address

cfgphy_address = 10
cfgphy_address

dc64support = 8
dc64support

dcsupport = 7
dcsupport

device_name = 11
device_name

esctype = 51
esctype

ismailbox_slave = 12
ismailbox_slave

isoptional = 49
isoptional

ispresent = 50
ispresent

mbx_insize = 46
mbx_insize

mbx_insize2 = 48
mbx_insize2

mbx_outsize = 45
mbx_outsize

mbx_outsize2 = 47
mbx_outsize2

pdoffs_in = 1
pdoffs_in

pdoffs_in2 = 21
pdoffs_in2

pdoffs_in3 = 31
pdoffs_in3

pdoffs_in4 = 41
pdoffs_in4

pdoffs_out = 3
pdoffs_out

pdoffs_out2 = 23
pdoffs_out2

pdoffs_out3 = 33
pdoffs_out3

pdoffs_out4 = 43
pdoffs_out4

```

pdssize_in = 2
    pds_size_in
pdssize_in2 = 22
    pds_size_in2
pdssize_in3 = 32
    pds_size_in3
pdssize_in4 = 42
    pds_size_in4
pdssize_out = 4
    pds_size_out
pdssize_out2 = 24
    pds_size_out2
pdssize_out3 = 34
    pds_size_out3
pdssize_out4 = 44
    pds_size_out4
phys_address = 5
    phys_address
portstate = 6
    portstate
unknown = 0
    unknown

```

```
class EcWrapperPythonTypes.DN_ESC_SII_REG (value)
```

```
ALIASADDRESS
```

```

    ALIASADDRESS
BOOT_RECV_MBX = 20
    BOOT_RECV_MBX
BOOT_RECV_MBX_OFFSET = 20
    BOOT_RECV_MBX_OFFSET
BOOT_RECV_MBX_SIZE = 21
    BOOT_RECV_MBX_SIZE
BOOT_SEND_MBX = 22
    BOOT_SEND_MBX
BOOT_SEND_MBX_OFFSET = 22
    BOOT_SEND_MBX_OFFSET
BOOT_SEND_MBX_SIZE = 23
    BOOT_SEND_MBX_SIZE
CHECKSUM = 7
    CHECKSUM
EXTENDEDPDICONFIG = 3
    EXTENDED_PDICONFIG
FIRSTCATEGORYHDR = 64
    FIRSTCATEGORYHDR
MBX_PROTOCOL = 28
    MBX_PROTOCOL
PDICONFIG = 1
    PDICONFIG

```

```

PDICONTROL = 0
    PDICONTROL

PRODUCTCODE = 10
    PRODUCTCODE

REVISIONNUMBER = 12
    REVISIONNUMBER

REVISIONNUMBER_HI = 13
    REVISIONNUMBER_HI

REVISIONNUMBER_LO = 12
    REVISIONNUMBER_LO

SERIALNUMBER = 14
    SERIALNUMBER

STD_RECV_MBX = 24
    STD_RECV_MBX

STD_RECV_MBX_OFFSET = 24
    STD_RECV_MBX_OFFSET

STD_RECV_MBX_SIZE = 25
    STD_RECV_MBX_SIZE

STD_SEND_MBX = 26
    STD_SEND_MBX

STD_SEND_MBX_OFFSET = 26
    STD_SEND_MBX_OFFSET

STD_SEND_MBX_SIZE = 27
    STD_SEND_MBX_SIZE

SYNCIMPULSELENGTH = 2
    SYNCIMPULSELENGTH

VENDORID = 8
    VENDORID

class EcWrapperPythonTypes.DN_ETHERNET_ADDRESS
    b: List[int]
        MAC address

class EcWrapperPythonTypes.DN_MC_T_AXIS_PROFILE (value)                                DS402 = 1
    DS402

    NONE = 0
        NONE

    SERCOS = 2
        SERCOS

class EcWrapperPythonTypes.DN_MC_T_AXIS_TYPE (value)                                AXIS_TYPE_REAL_ALL
    AXIS_TYPE_REAL_ALL

    AXIS_TYPE_VIRTUAL = 0
        AXIS_TYPE_VIRTUAL

class EcWrapperPythonTypes.DN_MC_T_BUFFER_MODE (value)                                ABORTING = 0
    ABORTING

    BLENDING_HIGH = 5
        BLENDING_HIGH

```

```

    BLENDING_HIGH_STOP = 8
    BLENDING_HIGH_STOP

    BLENDING_LOW = 2
    BLENDING_LOW

    BLENDING_NEXT = 4
    BLENDING_NEXT

    BLENDING_NEXT_STOP = 7
    BLENDING_NEXT_STOP

    BLENDING_PREVIOUS = 3
    BLENDING_PREVIOUS

    BLENDING_PREVIOUS_STOP = 6
    BLENDING_PREVIOUS_STOP

    BUFFERED = 1
    BUFFERED

class EcWrapperPythonTypes.DN_MC_T_CAM_INTERPOL_TYPE (value)                                CAM_INTERPOL
    CAM_INTERPOL_TYPE_CUB

    CAM_INTERPOL_TYPE_LIN = 1
    CAM_INTERPOL_TYPE_LIN

class EcWrapperPythonTypes.DN_MC_T_CAM_VAR_TYPE (value)                                CAM_VAR_TYPE
    CAM_VAR_TYPE_INT

    CAM_VAR_TYPE_REAL = 2
    CAM_VAR_TYPE_REAL

class EcWrapperPythonTypes.DN_MC_T_CIA402_OPMODE (value)                                MODE_OP_CSP
    MODE_OP_CSP

    MODE_OP_CST = 10
    MODE_OP_CST

    MODE_OP_CSV = 9
    MODE_OP_CSV

    MODE_OP_HOMING = 6
    MODE_OP_HOMING

    MODE_OP_INTER_POS = 7
    MODE_OP_INTER_POS

    MODE_OP_PROF_POS = 1
    MODE_OP_PROF_POS

    MODE_OP_PROF_TOR = 4
    MODE_OP_PROF_TOR

    MODE_OP_PROF_VEL = 3
    MODE_OP_PROF_VEL

    MODE_OP_VELOCITY = 2
    MODE_OP_VELOCITY

class EcWrapperPythonTypes.DN_MC_T_CIA402_STATE (value)                                DEV_STATE_MA
    DEV_STATE_MALFCT_REACTION

    DEV_STATE_MALFUNCTION = 7
    DEV_STATE_MALFUNCTION

```

```

DEV_STATE_NOT_READY = 0
    DEV_STATE_NOT_READY

DEV_STATE_OP_ENABLED = 4
    DEV_STATE_OP_ENABLED

DEV_STATE_QUICK_STOP = 5
    DEV_STATE_QUICK_STOP

DEV_STATE_READY_TO_SWITCHON = 2
    DEV_STATE_READY_TO_SWITCHON

DEV_STATE_SWITCHED_ON = 3
    DEV_STATE_SWITCHED_ON

DEV_STATE_SWITCHON_DIS = 1
    DEV_STATE_SWITCHON_DIS

class EcWrapperPythonTypes.DN_MC_T_DIRECTION (value) DIR_CURRENT
    DIR_CURRENT
    DIR_NEGATIVE = 3
        DIR_NEGATIVE
    DIR_POSITIVE = 1
        DIR_POSITIVE
    DIR_SHORTEST = 2
        DIR_SHORTEST

class EcWrapperPythonTypes.DN_MC_T_EXECUTION_MODE (value) IMMEDIATELY
    IMMEDIATELY
    QUEUED = 1
        QUEUED

class EcWrapperPythonTypes.DN_MC_T_FBID (value) ID_ABORT_TRI
    ID_ABORT_TRIGGER
    ID_ACCELERATION_PROFILE = 16
        ID_ACCELERATION_PROFILE
    ID_CAMTABLE_SELECT = 37
        ID_CAMTABLE_SELECT
    ID_CAM_IN = 38
        ID_CAM_IN
    ID_CAM_OUT = 39
        ID_CAM_OUT
    ID_CHECK_TARGETPOS_REACHED = 42
        ID_CHECK_TARGETPOS_REACHED
    ID_DIGITAL_CAM_SWITCH = 34
        ID_DIGITAL_CAM_SWITCH
    ID_GEAR_IN = 40
        ID_GEAR_IN
    ID_GEAR_OUT = 41
        ID_GEAR_OUT
    ID_HALT = 4
        ID_HALT

```

```
ID_HALT_RECOVERY = 43
    ID_HALT_RECOVERY

ID_HALT_SUPERIMPOSED = 9
    ID_HALT_SUPERIMPOSED

ID_HOME = 2
    ID_HOME

ID_IDLE = 0
    ID_IDLE

ID_MOVE_ABSOLUTE = 5
    ID_MOVE_ABSOLUTE

ID_MOVE_ADDITIVE = 7
    ID_MOVE_ADDITIVE

ID_MOVE_CONTINUOUS_ABSOLUTE = 11
    ID_MOVE_CONTINUOUS_ABSOLUTE

ID_MOVE_CONTINUOUS_RELATIVE = 12
    ID_MOVE_CONTINUOUS_RELATIVE

ID_MOVE_RELATIVE = 6
    ID_MOVE_RELATIVE

ID_MOVE_SUPERIMPOSED = 8
    ID_MOVE_SUPERIMPOSED

ID_MOVE_VELOCITY = 10
    ID_MOVE_VELOCITY

ID_POSITION_PROFILE = 14
    ID_POSITION_PROFILE

ID_POWER = 1
    ID_POWER

ID_READ_ACTUAL_POSITION = 26
    ID_READ_ACTUAL_POSITION

ID_READ_ACTUAL_TORQUE = 28
    ID_READ_ACTUAL_TORQUE

ID_READ_ACTUAL_VELOCITY = 27
    ID_READ_ACTUAL_VELOCITY

ID_READ_AXIS_ERROR = 32
    ID_READ_AXIS_ERROR

ID_READ_AXIS_INFO = 31
    ID_READ_AXIS_INFO

ID_READ_BOOL_PARAMETER = 20
    ID_READ_BOOL_PARAMETER

ID_READ_DIGITAL_INPUT = 23
    ID_READ_DIGITAL_INPUT

ID_READ_DIGITAL_OUTPUT = 24
    ID_READ_DIGITAL_OUTPUT

ID_READ_MOTION_STATE = 30
    ID_READ_MOTION_STATE

ID_READ_PARAMETER = 19
    ID_READ_PARAMETER
```

```
ID_READ_STATUS = 29
    ID_READ_STATUS

ID_RESET = 33
    ID_RESET

ID_SET_OVERRIDE = 18
    ID_SET_OVERRIDE

ID_SET_POSITION = 17
    ID_SET_POSITION

ID_STOP = 3
    ID_STOP

ID_TORQUE_CONTROL = 13
    ID_TORQUE_CONTROL

ID_TOUCH_PROBE = 35
    ID_TOUCH_PROBE

ID_VELOCITY_PROFILE = 15
    ID_VELOCITY_PROFILE

ID_WRITE_BOOL_PARAMETER = 22
    ID_WRITE_BOOL_PARAMETER

ID_WRITE_DIGITAL_OUTPUT = 25
    ID_WRITE_DIGITAL_OUTPUT

ID_WRITE_PARAMETER = 21
    ID_WRITE_PARAMETER
```

```
class EcWrapperPythonTypes.DN_MC_T_FBSTATE (value)
```

```
ACTIVE = 2
```

```
    ACTIVE

    BUSY = 1
        BUSY

    COMMAND_ABORTED = 5
        COMMAND_ABORTED

    DONE = 3
        DONE

    ERROR = 4
        ERROR

    IDLE = 0
        IDLE
```

```
class EcWrapperPythonTypes.DN_MC_T_OPMODE (value)
```

```
CSP = 1
```

```
    CSP

    CST = 3
        CST

    CSV = 2
        CSV

    HOMING = 5
        HOMING

    NOTSET = 0
        NOTSET

    PP = 4
        PP
```

```

class EcWrapperPythonTypes.DN_MC_T_PLCOOPEN_STATE (value) PLCOOPEN_STAT
    continous motion
    PLCOOPEN_STATE_DISABLED = 11
        disabled
    PLCOOPEN_STATE_DISC_MOTION = 5
        discrete motion
    PLCOOPEN_STATE_ERROR_STOP = 4
        error stop
    PLCOOPEN_STATE_HOMING = 2
        homing
    PLCOOPEN_STATE_STAND_STILL = 1
        stand still
    PLCOOPEN_STATE_STOPPING = 3
        stopping
    PLCOOPEN_STATE_SYNC_MOTION = 7
        synchronized motion
    PLCOOPEN_STATE_TORQUE_CONTROL = 10
        torque control
    PLCOOPEN_STATE_UNKNOWN = 0
        unknown

class EcWrapperPythonTypes.DN_MC_T_PN (value) PN_ACTUAL_PO
    PN_ACTUAL_POSITION
    PN_ACTUAL_VELOCITY = 10
        PN_ACTUAL_VELOCITY
    PN_CMDANDED_ACCELERATION = 1000
        PN_CMDANDED_ACCELERATION
    PN_CMDANDED_JERK = 1001
        PN_CMDANDED_JERK
    PN_CMDANDED_POSITION = 1
        PN_CMDANDED_POSITION
    PN_CMDANDED_VELOCITY = 11
        PN_CMDANDED_VELOCITY
    PN_ENA_LIMIT_NEG = 5
        PN_ENA_LIMIT_NEG
    PN_ENA_LIMIT_POS = 4
        PN_ENA_LIMIT_POS
    PN_SW_LIMIT_NEG = 3
        PN_SW_LIMIT_NEG
    PN_SW_LIMIT_POS = 2
        PN_SW_LIMIT_POS

class EcWrapperPythonTypes.DN_MC_T_PROFILE_STATE (value) Error = 0
    Error
    Halt = 8
        Halt

```

```

Idle = 0
    Idle

NewSetPoint = 1
    NewSetPoint

StartHoming = 6
    StartHoming

SwitchToHomingMode = 5
    SwitchToHomingMode

WaitForAck = 2
    WaitForAck

WaitHaltDone = 9
    WaitHaltDone

WaitHomingDone = 7
    WaitHomingDone

WaitQStop = 4
    WaitQStop

WaitReached = 3
    WaitReached

class EcWrapperPythonTypes.DN_MC_T_RECOVERY_MODE (value)
    RECOVERY_ABORT_MOVEMENT
    RECOVERY_NO_ACTION = 0
    RECOVERY_NO_ACTION

class EcWrapperPythonTypes.DN_MC_T_SERCOS_OPMODE (value)
    OPMODE_POS_FB1
    OPMODE_POS_FB1FB2 = 5
    OPMODE_POS_FB1FB2

    OPMODE_POS_FB1FB2_LAGLESS = 13
    OPMODE_POS_FB1FB2_LAGLESS

    OPMODE_POS_FB1_LAGLESS = 11
    OPMODE_POS_FB1_LAGLESS

    OPMODE_POS_FB2 = 4
    OPMODE_POS_FB2

    OPMODE_POS_FB2_LAGLESS = 12
    OPMODE_POS_FB2_LAGLESS

    OPMODE_TORQUE = 1
    OPMODE_TORQUE

    OPMODE_VEL = 2
    OPMODE_VEL

class EcWrapperPythonTypes.DN_MC_T_SERCOS_STATE (value)
    SER_STATE_MALFUNCTION

    SER_STATE_NOT_READY = 0
    SER_STATE_NOT_READY

    SER_STATE_OP_ENABLED = 4
    SER_STATE_OP_ENABLED

```

```

SER_STATE_READY_TO_SWITCHON = 2
SER_STATE_READY_TO_SWITCHON

SER_STATE_SWITCHED_ON = 3
SER_STATE_SWITCHED_ON

class EcWrapperPythonTypes.DN_MC_T_SETVALUE_SOURCE (value) MASTER_CAMM

    MASTER_CAMM

MASTER_GEAR = 4
MASTER_GEAR

NONE = 0
NONE

TRAJGEN_DRV = 3
TRAJGEN_DRV

TRAJGEN_ECM = 1
TRAJGEN_ECM

TRAJGEN_EXT = 2
TRAJGEN_EXT

class EcWrapperPythonTypes.DN_MC_T_SIMU_STATE (value) SIM_STATE_OP

    SIM_STATE_OP_ENABLED

class EcWrapperPythonTypes.DN_MC_T_SOURCE (value) SOURCE_ACTUA

    SOURCE_ACTUAL_VALUE

SOURCE_SET_VALUE = 1
SOURCE_SET_VALUE

class EcWrapperPythonTypes.DN_MC_T_START_MODE (value) SM_ABSOLUTE

    SM_ABSOLUTE

SM_RAMP_IN = 2
SM_RAMP_IN

SM_RAMP_IN_NEG = 4
SM_RAMP_IN_NEG

SM_RAMP_IN_POS = 3
SM_RAMP_IN_POS

SM_RELATIVE = 1
SM_RELATIVE

class EcWrapperPythonTypes.DN_MC_T_TRAJECT_STATE (value) Acc = 1

    Acc

Dec = 2
Dec

Idle = 0
Idle

Vel = 3
Vel

class EcWrapperPythonTypes.DN_NotifyCode (value)
    Specifies EtherCAT errors codes

    ALL_DEVICES_OPERATIONAL = 65569

    AOE_MBXSNW_WKC_ERROR = 65581

```

```
BAD_CONNECTION = 65578
CLIENTREGISTRATION_DROPPED = 65562
COE_MBXSND_WKC_ERROR = 65544
COE_TX_PDO = 131073
COMMUNICATION_TIMEOUT = 65579
CONFIG_LOAD = 111
CYCCMD_WKC_ERROR = 65537
DCL_STATUS = 8
DCM_SYNC = 9
DCX_SYNC = 10
DC_SLV_SYNC = 5
DC_STATUS = 4
EEPROM_CHECKSUM_ERROR = 65571
EOE_MBXSND_WKC_ERROR = 65543
ETH_LINK_CONNECTED = 2
ETH_LINK_NOT_CONNECTED = 65552
FOE_MBSLAVE_ERROR = 65564
FOE_MBXSND_WKC_ERROR = 65545
FRAMELOSS_AFTER_SLAVE = 65576
FRAME_RESPONSE_ERROR = 65546
FSOE_CONNECTION_STATECHANGED = 112
HC_DETECTADDGROUPS = 262146
HC_PROBEALLGROUPS = 262147
HC_TOPOCHGDONE = 262148
JUNCTION_RED_CHANGE = 65573
LINE_CROSSED = 65572
MASTER_INITCMD_RESPONSE_ERROR = 65548
MASTER_INITCMD_WKC_ERROR = 65538
MASTER_RED_FOREIGN_SRC_MAC = 105
MASTER_RED_STATECHANGED = 104
MBOXRCV = 131072
MBSLAVE_COE_SDO_ABORT = 65561
MBSLAVE_INITCMD_TIMEOUT = 65550
MBXRCV_INVALID_DATA = 65565
NOT_ALL_DEVICES_OPERATIONAL = 65551
PDIWATCHDOG = 65566
PORT_OPERATION = 108
RAWCMD_DONE = 100
RED_LINEBRK = 65554
```

```
RED_LINEFIXED = 65563
REFCLOCK_PRESENCE = 103
RELEASE_FORCED_PROCESSDATA = 110
S2SMBX_ERROR = 65577
SB_DUPLICATE_HC_NODE = 196611
SB_EEPROM_ACCESS_DENIED = 196612
SB_MISMATCH = 196610
SB_SLAVE_IDENT_TIMEOUT = 196613
SB_STATUS = 3
SLAVES_ERROR_STATUS = 65575
SLAVES_PRESENCE = 102
SLAVES_STATECHANGED = 22
SLAVES_UNEXPECTED_STATE = 65574
SLAVE_ERROR_STATUS_INFO = 65556
SLAVE_IDENTIFICATION = 109
SLAVE_INITCMD_RESPONSE_ERROR = 65547
SLAVE_INITCMD_WKC_ERROR = 65539
SLAVE_NOTSUPPORTED = 65567
SLAVE_NOT_ADDRESSABLE = 65557
SLAVE_PRESENCE = 101
SLAVE_REGISTER_TRANSFER = 106
SLAVE_STATECHANGED = 21
SLAVE_UNEXPECTED_STATE = 65568
SOE_MBXSNW_WKC_ERROR = 65559
SOE_WRITE_ERROR = 65560
STATECHANGED = 1
STATUS_SLAVE_ERROR = 65555
TAP_LINK_STATUS = 65580
UNDEFINED = 0
VOE_MBXSNW_WKC_ERROR = 65570
```

```
class EcWrapperPythonTypes.DN_NotifyType (value)
```

An enumeration.

```
EApp = 6
EBusScan = 4
EError = 1
EHotConnect = 5
EInfo = 2
EMailBox = 3
EUnknown = 0
```

```
class EcWrapperPythonTypes.DN_RasNotifyCode (value)
    Specifies EtherCAT errors codes

    ACKERROR = 1114114

    CONNECTION = 1048577

    EXCEPTION = 1114118

    MARSHALERROR = 1114113

    MBXNOTIFYMEMORYSMALL = 1114117

    NONOTIFYMEMORY = 1114115

    REGISTER = 1048578

    STDNOTIFYMEMORYSMALL = 1114116

    UNREGISTER = 1048579

class EcWrapperPythonTypes.EC_T_DEFAULT_VALUE (value)
    An enumeration.

    BCppDummy = 4294967295

    EC_T_DEFAULT_VALUE_ATECAT_SIGNATURE = 1

    EC_T_DEFAULT_VALUE_EC_VERSION = 0

    EC_T_DEFAULT_VALUE_EL9010_SLAVE_ID = 6

    EC_T_DEFAULT_VALUE_FRAMELOSS_SLAVE_ID = 7

    EC_T_DEFAULT_VALUE_INVALID_SLAVE_ID = 2

    EC_T_DEFAULT_VALUE_JUNCTION_RED_FLAG = 8

    EC_T_DEFAULT_VALUE_MASTER_RED_SLAVE_ID = 5

    EC_T_DEFAULT_VALUE_MASTER_SLAVE_ID = 3

    EC_T_DEFAULT_VALUE_MAX_NUMOF_MASTER_INSTANCES = 4

class EcWrapperPythonTypes.EC_T_MOTION_CMD (value)
    An enumeration.

    EC_T_MOTION_CMD_AXIS_INIT_CREATE = 0

    EC_T_MOTION_CMD_AXIS_INIT_DELETE = 1

    EC_T_MOTION_CMD_AXIS_INIT_ECAT_CREATE = 150

    EC_T_MOTION_CMD_AXIS_INIT_ECAT_DELETE = 151

    EC_T_MOTION_CMD_AXIS_INIT_ECAT_SET_COEIDXOPMODE = 156

    EC_T_MOTION_CMD_AXIS_INIT_ECAT_SET_COESUBIDXOPMODE = 157

    EC_T_MOTION_CMD_AXIS_INIT_ECAT_SET_PCONTROLWORD = 159

    EC_T_MOTION_CMD_AXIS_INIT_ECAT_SET_PROFILE = 154

    EC_T_MOTION_CMD_AXIS_INIT_ECAT_SET_PSTATUSWORD = 158

    EC_T_MOTION_CMD_AXIS_INIT_ECAT_SET_SERCOSDRIVENO = 155

    EC_T_MOTION_CMD_AXIS_INIT_ECAT_SET_SLAVEID = 152

    EC_T_MOTION_CMD_AXIS_INIT_ECAT_SET_STATIONADDRESS = 153

    EC_T_MOTION_CMD_AXIS_INIT_INPUTS_CREATE = 100

    EC_T_MOTION_CMD_AXIS_INIT_INPUTS_DELETE = 101

    EC_T_MOTION_CMD_AXIS_INIT_INPUTS_SET_PACTUALPOSITION = 102
```

```
EC_T_MOTION_CMD_AXIS_INIT_INPUTS_SET_PACTUALTORQUE = 103
EC_T_MOTION_CMD_AXIS_INIT_INPUTS_SET_PDIGITALINPUTS = 104
EC_T_MOTION_CMD_AXIS_INIT_OUTPUTS_CREATE = 50
EC_T_MOTION_CMD_AXIS_INIT_OUTPUTS_DELETE = 51
EC_T_MOTION_CMD_AXIS_INIT_OUTPUTS_SET_PDIGITALOUTPUTS = 59
EC_T_MOTION_CMD_AXIS_INIT_OUTPUTS_SET_PMODEOFOPERATION = 57
EC_T_MOTION_CMD_AXIS_INIT_OUTPUTS_SET_PMODEOFOPERATIONDISPLAY = 58
EC_T_MOTION_CMD_AXIS_INIT_OUTPUTS_SET_PTARGETPOSITION = 52
EC_T_MOTION_CMD_AXIS_INIT_OUTPUTS_SET_PTARGETTORQUE = 55
EC_T_MOTION_CMD_AXIS_INIT_OUTPUTS_SET_PTARGETVELOCITY = 53
EC_T_MOTION_CMD_AXIS_INIT_OUTPUTS_SET_PTORQUEOFFSET = 56
EC_T_MOTION_CMD_AXIS_INIT_OUTPUTS_SET_PVELOCITYOFFSET = 54
EC_T_MOTION_CMD_AXIS_INIT_SET_AXISTYPE = 2
EC_T_MOTION_CMD_AXIS_INIT_SET_CYCLETIME = 3
EC_T_MOTION_CMD_AXIS_INIT_SET_INCFACTOR = 5
EC_T_MOTION_CMD_AXIS_INIT_SET_INCPERMM = 4
EC_T_MOTION_CMD_AXIS_INIT_SET_TORQUEGAIN = 7
EC_T_MOTION_CMD_AXIS_INIT_SET_VELOCITYGAIN = 6
EC_T_MOTION_CMD_AXIS_REF_CREATE = 200
EC_T_MOTION_CMD_AXIS_REF_DELETE = 201
EC_T_MOTION_CMD_AXIS_REF_EXEC_INIT = 202
EC_T_MOTION_CMD_AXIS_REF_EXEC_INITECAT = 205
EC_T_MOTION_CMD_AXIS_REF_EXEC_INITINPUTS = 203
EC_T_MOTION_CMD_AXIS_REF_EXEC_INITOUTPUTS = 204
EC_T_MOTION_CMD_AXIS_REF_EXEC_SETMODEOFOPERATION = 206
EC_T_MOTION_CMD_HALT_CREATE = 400
EC_T_MOTION_CMD_HALT_DELETE = 401
EC_T_MOTION_CMD_HALT_EXEC_ONCYCLE = 411
EC_T_MOTION_CMD_HALT_GET_BUSY = 404
EC_T_MOTION_CMD_HALT_GET_DECELERATION = 407
EC_T_MOTION_CMD_HALT_GET_DONE = 403
EC_T_MOTION_CMD_HALT_GET_EXECUTE = 405
EC_T_MOTION_CMD_HALT_GET_JERK = 409
EC_T_MOTION_CMD_HALT_SET_AXIS = 402
EC_T_MOTION_CMD_HALT_SET_DECELERATION = 408
EC_T_MOTION_CMD_HALT_SET_EXECUTE = 406
EC_T_MOTION_CMD_HALT_SET_JERK = 410
EC_T_MOTION_CMD_HOME_CREATE = 300
EC_T_MOTION_CMD_HOME_DELETE = 301
```

```
EC_T_MOTION_CMD_HOME_EXEC_ONCYCLE = 311
EC_T_MOTION_CMD_HOME_GET_BUSY = 304
EC_T_MOTION_CMD_HOME_GET_DONE = 303
EC_T_MOTION_CMD_HOME_GET_ENABLESETPOSITION = 309
EC_T_MOTION_CMD_HOME_GET_EXECUTE = 305
EC_T_MOTION_CMD_HOME_GET_POSITION = 307
EC_T_MOTION_CMD_HOME_SET_AXIS = 302
EC_T_MOTION_CMD_HOME_SET_ENABLESETPOSITION = 310
EC_T_MOTION_CMD_HOME_SET_EXECUTE = 306
EC_T_MOTION_CMD_HOME_SET_POSITION = 308
EC_T_MOTION_CMD_MOVE_ABSOLUTE_CREATE = 450
EC_T_MOTION_CMD_MOVE_ABSOLUTE_DELETE = 451
EC_T_MOTION_CMD_MOVE_ABSOLUTE_EXEC_ONCYCLE = 471
EC_T_MOTION_CMD_MOVE_ABSOLUTE_GET_ACCELERATION = 463
EC_T_MOTION_CMD_MOVE_ABSOLUTE_GET_BUSY = 454
EC_T_MOTION_CMD_MOVE_ABSOLUTE_GET_CONTINUOUSUPDATE = 457
EC_T_MOTION_CMD_MOVE_ABSOLUTE_GET_DECELERATION = 465
EC_T_MOTION_CMD_MOVE_ABSOLUTE_GET_DIRECTION = 469
EC_T_MOTION_CMD_MOVE_ABSOLUTE_GET_DONE = 453
EC_T_MOTION_CMD_MOVE_ABSOLUTE_GET_EXECUTE = 455
EC_T_MOTION_CMD_MOVE_ABSOLUTE_GET_JERK = 467
EC_T_MOTION_CMD_MOVE_ABSOLUTE_GET_POSITION = 459
EC_T_MOTION_CMD_MOVE_ABSOLUTE_GET_VELOCITY = 461
EC_T_MOTION_CMD_MOVE_ABSOLUTE_SET_ACCELERATION = 464
EC_T_MOTION_CMD_MOVE_ABSOLUTE_SET_AXIS = 452
EC_T_MOTION_CMD_MOVE_ABSOLUTE_SET_CONTINUOUSUPDATE = 458
EC_T_MOTION_CMD_MOVE_ABSOLUTE_SET_DECELERATION = 466
EC_T_MOTION_CMD_MOVE_ABSOLUTE_SET_DIRECTION = 470
EC_T_MOTION_CMD_MOVE_ABSOLUTE_SET_EXECUTE = 456
EC_T_MOTION_CMD_MOVE_ABSOLUTE_SET_JERK = 468
EC_T_MOTION_CMD_MOVE_ABSOLUTE_SET_POSITION = 460
EC_T_MOTION_CMD_MOVE_ABSOLUTE_SET_VELOCITY = 462
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_CREATE = 600
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_DELETE = 601
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_EXEC_ONCYCLE = 623
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_GET_ACCELERATION = 615
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_GET_BUSY = 604
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_GET_CONTINUOUSUPDATE = 607
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_GET_DECELERATION = 617
```

```
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_GET_DIRECTION = 621
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_GET_ENDVELOCITY = 611
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_GET_EXECUTE = 605
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_GET_INENDVELOCITY = 603
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_GET_JERK = 619
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_GET_POSITION = 609
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_GET_VELOCITY = 613
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_SET_ACCELERATION = 616
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_SET_AXIS = 602
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_SET_CONTINUOUSUPDATE = 608
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_SET_DECELERATION = 618
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_SET_DIRECTION = 622
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_SET_ENDVELOCITY = 612
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_SET_EXECUTE = 606
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_SET_JERK = 620
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_SET_POSITION = 610
EC_T_MOTION_CMD_MOVE_CONT_ABSOLUTE_SET_VELOCITY = 614
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_CREATE = 650
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_DELETE = 651
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_EXEC_ONCYCLE = 671
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_GET_ACCELERATION = 665
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_GET_BUSY = 654
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_GET_CONTINUOUSUPDATE = 657
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_GET_DECELERATION = 667
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_GET_DISTANCE = 659
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_GET_ENDVELOCITY = 661
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_GET_EXECUTE = 655
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_GET_INENDVELOCITY = 653
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_GET_JERK = 669
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_GET_VELOCITY = 663
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_SET_ACCELERATION = 666
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_SET_AXIS = 652
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_SET_CONTINUOUSUPDATE = 658
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_SET_DECELERATION = 668
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_SET_DISTANCE = 660
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_SET_ENDVELOCITY = 662
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_SET_EXECUTE = 656
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_SET_JERK = 670
EC_T_MOTION_CMD_MOVE_CONT_RELATIVE_SET_VELOCITY = 664
```

```
EC_T_MOTION_CMD_MOVE_RELATIVE_CREATE = 500
EC_T_MOTION_CMD_MOVE_RELATIVE_DELETE = 501
EC_T_MOTION_CMD_MOVE_RELATIVE_EXEC_ONCYCLE = 519
EC_T_MOTION_CMD_MOVE_RELATIVE_GET_ACCELERATION = 513
EC_T_MOTION_CMD_MOVE_RELATIVE_GET_BUSY = 504
EC_T_MOTION_CMD_MOVE_RELATIVE_GET_CONTINUOUSUPDATE = 507
EC_T_MOTION_CMD_MOVE_RELATIVE_GET_DECELERATION = 515
EC_T_MOTION_CMD_MOVE_RELATIVE_GET_DISTANCE = 509
EC_T_MOTION_CMD_MOVE_RELATIVE_GET_DONE = 503
EC_T_MOTION_CMD_MOVE_RELATIVE_GET_EXECUTE = 505
EC_T_MOTION_CMD_MOVE_RELATIVE_GET_JERK = 517
EC_T_MOTION_CMD_MOVE_RELATIVE_GET_VELOCITY = 511
EC_T_MOTION_CMD_MOVE_RELATIVE_SET_ACCELERATION = 514
EC_T_MOTION_CMD_MOVE_RELATIVE_SET_AXIS = 502
EC_T_MOTION_CMD_MOVE_RELATIVE_SET_CONTINUOUSUPDATE = 508
EC_T_MOTION_CMD_MOVE_RELATIVE_SET_DECELERATION = 516
EC_T_MOTION_CMD_MOVE_RELATIVE_SET_DISTANCE = 510
EC_T_MOTION_CMD_MOVE_RELATIVE_SET_EXECUTE = 506
EC_T_MOTION_CMD_MOVE_RELATIVE_SET_JERK = 518
EC_T_MOTION_CMD_MOVE_RELATIVE_SET_VELOCITY = 512
EC_T_MOTION_CMD_MOVE_VELOCITY_CREATE = 550
EC_T_MOTION_CMD_MOVE_VELOCITY_DELETE = 551
EC_T_MOTION_CMD_MOVE_VELOCITY_EXEC_ONCYCLE = 569
EC_T_MOTION_CMD_MOVE_VELOCITY_GET_ACCELERATION = 561
EC_T_MOTION_CMD_MOVE_VELOCITY_GET_BUSY = 554
EC_T_MOTION_CMD_MOVE_VELOCITY_GET_CONTINUOUSUPDATE = 557
EC_T_MOTION_CMD_MOVE_VELOCITY_GET_DECELERATION = 563
EC_T_MOTION_CMD_MOVE_VELOCITY_GET_DIRECTION = 567
EC_T_MOTION_CMD_MOVE_VELOCITY_GET_EXECUTE = 555
EC_T_MOTION_CMD_MOVE_VELOCITY_GET_INVELOCITY = 553
EC_T_MOTION_CMD_MOVE_VELOCITY_GET_JERK = 565
EC_T_MOTION_CMD_MOVE_VELOCITY_GET_VELOCITY = 559
EC_T_MOTION_CMD_MOVE_VELOCITY_SET_ACCELERATION = 562
EC_T_MOTION_CMD_MOVE_VELOCITY_SET_AXIS = 552
EC_T_MOTION_CMD_MOVE_VELOCITY_SET_CONTINUOUSUPDATE = 558
EC_T_MOTION_CMD_MOVE_VELOCITY_SET_DECELERATION = 564
EC_T_MOTION_CMD_MOVE_VELOCITY_SET_DIRECTION = 568
EC_T_MOTION_CMD_MOVE_VELOCITY_SET_EXECUTE = 556
EC_T_MOTION_CMD_MOVE_VELOCITY_SET_JERK = 566
```

```
EC_T_MOTION_CMD_MOVE_VELOCITY_SET_VELOCITY = 560
EC_T_MOTION_CMD_POWER_CREATE = 250
EC_T_MOTION_CMD_POWER_DELETE = 251
EC_T_MOTION_CMD_POWER_EXEC_ONCYCLE = 262
EC_T_MOTION_CMD_POWER_GET_ENABLE = 256
EC_T_MOTION_CMD_POWER_GET_ENABLENEGATIVE = 260
EC_T_MOTION_CMD_POWER_GET_ENABLEPOSITIVE = 258
EC_T_MOTION_CMD_POWER_GET_FAULTREACTION = 255
EC_T_MOTION_CMD_POWER_GET_STATUS = 253
EC_T_MOTION_CMD_POWER_GET_VALID = 254
EC_T_MOTION_CMD_POWER_SET_AXIS = 252
EC_T_MOTION_CMD_POWER_SET_ENABLE = 257
EC_T_MOTION_CMD_POWER_SET_ENABLENEGATIVE = 261
EC_T_MOTION_CMD_POWER_SET_ENABLEPOSITIVE = 259
EC_T_MOTION_CMD_READ_ACTUAL_POSITION_CREATE = 1100
EC_T_MOTION_CMD_READ_ACTUAL_POSITION_DELETE = 1101
EC_T_MOTION_CMD_READ_ACTUAL_POSITION_EXEC_ONCYCLE = 1107
EC_T_MOTION_CMD_READ_ACTUAL_POSITION_GET_ENABLE = 1105
EC_T_MOTION_CMD_READ_ACTUAL_POSITION_GET_POSITION = 1104
EC_T_MOTION_CMD_READ_ACTUAL_POSITION_GET_VALID = 1103
EC_T_MOTION_CMD_READ_ACTUAL_POSITION_SET_AXIS = 1102
EC_T_MOTION_CMD_READ_ACTUAL_POSITION_SET_ENABLE = 1106
EC_T_MOTION_CMD_READ_ACTUAL_VELOCITY_CREATE = 1150
EC_T_MOTION_CMD_READ_ACTUAL_VELOCITY_DELETE = 1151
EC_T_MOTION_CMD_READ_ACTUAL_VELOCITY_EXEC_ONCYCLE = 1157
EC_T_MOTION_CMD_READ_ACTUAL_VELOCITY_GET_ENABLE = 1155
EC_T_MOTION_CMD_READ_ACTUAL_VELOCITY_GET_VALID = 1153
EC_T_MOTION_CMD_READ_ACTUAL_VELOCITY_GET_VELOCITY = 1154
EC_T_MOTION_CMD_READ_ACTUAL_VELOCITY_SET_AXIS = 1152
EC_T_MOTION_CMD_READ_ACTUAL_VELOCITY_SET_ENABLE = 1156
EC_T_MOTION_CMD_READ_AXIS_INFO_CREATE = 1250
EC_T_MOTION_CMD_READ_AXIS_INFO_DELETE = 1251
EC_T_MOTION_CMD_READ_AXIS_INFO_EXEC_ONCYCLE = 1264
EC_T_MOTION_CMD_READ_AXIS_INFO_GET_COMMANDEDPOSITION = 1259
EC_T_MOTION_CMD_READ_AXIS_INFO_GET_DRIVEERRORACK = 1262
EC_T_MOTION_CMD_READ_AXIS_INFO_GET_DRIVEERRORACKREQ = 1258
EC_T_MOTION_CMD_READ_AXIS_INFO_GET_DRIVESTATE = 1257
EC_T_MOTION_CMD_READ_AXIS_INFO_GET_ENABLE = 1260
EC_T_MOTION_CMD_READ_AXIS_INFO_GET_PLCOOPENSTATE = 1256
```

```
EC_T_MOTION_CMD_READ_AXIS_INFO_GET_SIMULATION = 1254
EC_T_MOTION_CMD_READ_AXIS_INFO_GET_VALID = 1253
EC_T_MOTION_CMD_READ_AXIS_INFO_GET_VERBOSE = 1255
EC_T_MOTION_CMD_READ_AXIS_INFO_SET_AXIS = 1252
EC_T_MOTION_CMD_READ_AXIS_INFO_SET_DRIVEERRORACK = 1263
EC_T_MOTION_CMD_READ_AXIS_INFO_SET_ENABLE = 1261
EC_T_MOTION_CMD_READ_BOOL_PARAMETER_CREATE = 800
EC_T_MOTION_CMD_READ_BOOL_PARAMETER_DELETE = 801
EC_T_MOTION_CMD_READ_BOOL_PARAMETER_EXEC_ONCYCLE = 810
EC_T_MOTION_CMD_READ_BOOL_PARAMETER_GET_BUSY = 804
EC_T_MOTION_CMD_READ_BOOL_PARAMETER_GET_ENABLE = 806
EC_T_MOTION_CMD_READ_BOOL_PARAMETER_GET_PARAMETERNUMBER = 808
EC_T_MOTION_CMD_READ_BOOL_PARAMETER_GET_VALID = 803
EC_T_MOTION_CMD_READ_BOOL_PARAMETER_GET_VALUE = 805
EC_T_MOTION_CMD_READ_BOOL_PARAMETER_SET_AXIS = 802
EC_T_MOTION_CMD_READ_BOOL_PARAMETER_SET_ENABLE = 807
EC_T_MOTION_CMD_READ_BOOL_PARAMETER_SET_PARAMETERNUMBER = 809
EC_T_MOTION_CMD_READ_DIGITAL_INPUT_CREATE = 950
EC_T_MOTION_CMD_READ_DIGITAL_INPUT_DELETE = 951
EC_T_MOTION_CMD_READ_DIGITAL_INPUT_EXEC_ONCYCLE = 960
EC_T_MOTION_CMD_READ_DIGITAL_INPUT_GET_BUSY = 954
EC_T_MOTION_CMD_READ_DIGITAL_INPUT_GET_ENABLE = 956
EC_T_MOTION_CMD_READ_DIGITAL_INPUT_GET_INPUTNUMBER = 958
EC_T_MOTION_CMD_READ_DIGITAL_INPUT_GET_VALID = 953
EC_T_MOTION_CMD_READ_DIGITAL_INPUT_GET_VALUE = 955
EC_T_MOTION_CMD_READ_DIGITAL_INPUT_SET_AXIS = 952
EC_T_MOTION_CMD_READ_DIGITAL_INPUT_SET_ENABLE = 957
EC_T_MOTION_CMD_READ_DIGITAL_INPUT_SET_INPUTNUMBER = 959
EC_T_MOTION_CMD_READ_DIGITAL_OUTPUT_CREATE = 1000
EC_T_MOTION_CMD_READ_DIGITAL_OUTPUT_DELETE = 1001
EC_T_MOTION_CMD_READ_DIGITAL_OUTPUT_EXEC_ONCYCLE = 1010
EC_T_MOTION_CMD_READ_DIGITAL_OUTPUT_GET_BUSY = 1004
EC_T_MOTION_CMD_READ_DIGITAL_OUTPUT_GET_ENABLE = 1006
EC_T_MOTION_CMD_READ_DIGITAL_OUTPUT_GET_OUTPUTNUMBER = 1008
EC_T_MOTION_CMD_READ_DIGITAL_OUTPUT_GET_VALID = 1003
EC_T_MOTION_CMD_READ_DIGITAL_OUTPUT_GET_VALUE = 1005
EC_T_MOTION_CMD_READ_DIGITAL_OUTPUT_SET_AXIS = 1002
EC_T_MOTION_CMD_READ_DIGITAL_OUTPUT_SET_ENABLE = 1007
EC_T_MOTION_CMD_READ_DIGITAL_OUTPUT_SET_OUTPUTNUMBER = 1009
```

```
EC_T_MOTION_CMD_READ_ERROR_CREATE = 1300
EC_T_MOTION_CMD_READ_ERROR_DELETE = 1301
EC_T_MOTION_CMD_READ_ERROR_EXEC_ONCYCLE = 1306
EC_T_MOTION_CMD_READ_ERROR_GET_ENABLE = 1304
EC_T_MOTION_CMD_READ_ERROR_GET_VALID = 1303
EC_T_MOTION_CMD_READ_ERROR_SET_AXIS = 1302
EC_T_MOTION_CMD_READ_ERROR_SET_ENABLE = 1305
EC_T_MOTION_CMD_READ_MOTION_STATE_CREATE = 1200
EC_T_MOTION_CMD_READ_MOTION_STATE_DELETE = 1201
EC_T_MOTION_CMD_READ_MOTION_STATE_EXEC_ONCYCLE = 1211
EC_T_MOTION_CMD_READ_MOTION_STATE_GET_ACCELERATING = 1205
EC_T_MOTION_CMD_READ_MOTION_STATE_GET_CONSTANTVELOCITY = 1204
EC_T_MOTION_CMD_READ_MOTION_STATE_GET_DECELERATING = 1206
EC_T_MOTION_CMD_READ_MOTION_STATE_GET_DIRECTIONNEGATIVE = 1208
EC_T_MOTION_CMD_READ_MOTION_STATE_GET_DIRECTIONPOSITIVE = 1207
EC_T_MOTION_CMD_READ_MOTION_STATE_GET_ENABLE = 1209
EC_T_MOTION_CMD_READ_MOTION_STATE_GET_VALID = 1203
EC_T_MOTION_CMD_READ_MOTION_STATE_SET_AXIS = 1202
EC_T_MOTION_CMD_READ_MOTION_STATE_SET_ENABLE = 1210
EC_T_MOTION_CMD_READ_PARAMETER_CREATE = 750
EC_T_MOTION_CMD_READ_PARAMETER_DELETE = 751
EC_T_MOTION_CMD_READ_PARAMETER_EXEC_ONCYCLE = 760
EC_T_MOTION_CMD_READ_PARAMETER_GET_BUSY = 754
EC_T_MOTION_CMD_READ_PARAMETER_GET_ENABLE = 756
EC_T_MOTION_CMD_READ_PARAMETER_GET_PARAMETERNUMBER = 758
EC_T_MOTION_CMD_READ_PARAMETER_GET_VALID = 753
EC_T_MOTION_CMD_READ_PARAMETER_GET_VALUE = 755
EC_T_MOTION_CMD_READ_PARAMETER_SET_AXIS = 752
EC_T_MOTION_CMD_READ_PARAMETER_SET_ENABLE = 757
EC_T_MOTION_CMD_READ_PARAMETER_SET_PARAMETERNUMBER = 759
EC_T_MOTION_CMD_RESET_CREATE = 1350
EC_T_MOTION_CMD_RESET_DELETE = 1351
EC_T_MOTION_CMD_RESET_EXEC_ONCYCLE = 1357
EC_T_MOTION_CMD_RESET_GET_BUSY = 1354
EC_T_MOTION_CMD_RESET_GET_DONE = 1353
EC_T_MOTION_CMD_RESET_GET_EXECUTE = 1355
EC_T_MOTION_CMD_RESET_SET_AXIS = 1352
EC_T_MOTION_CMD_RESET_SET_EXECUTE = 1356
EC_T_MOTION_CMD_SET_POSITION_CREATE = 700
```

```
EC_T_MOTION_CMD_SET_POSITION_DELETE = 701
EC_T_MOTION_CMD_SET_POSITION_EXEC_ONCYCLE = 713
EC_T_MOTION_CMD_SET_POSITION_GET_BUSY = 704
EC_T_MOTION_CMD_SET_POSITION_GET_DONE = 703
EC_T_MOTION_CMD_SET_POSITION_GET_EXECUTE = 705
EC_T_MOTION_CMD_SET_POSITION_GET_EXECUTIONMODE = 711
EC_T_MOTION_CMD_SET_POSITION_GET_POSITION = 707
EC_T_MOTION_CMD_SET_POSITION_GET_RELATIVE = 709
EC_T_MOTION_CMD_SET_POSITION_SET_AXIS = 702
EC_T_MOTION_CMD_SET_POSITION_SET_EXECUTE = 706
EC_T_MOTION_CMD_SET_POSITION_SET_EXECUTIONMODE = 712
EC_T_MOTION_CMD_SET_POSITION_SET_POSITION = 708
EC_T_MOTION_CMD_SET_POSITION_SET_RELATIVE = 710
EC_T_MOTION_CMD_STOP_CREATE = 350
EC_T_MOTION_CMD_STOP_DELETE = 351
EC_T_MOTION_CMD_STOP_EXEC_ONCYCLE = 362
EC_T_MOTION_CMD_STOP_GET_BUSY = 354
EC_T_MOTION_CMD_STOP_GET_COMMANDABORTED = 355
EC_T_MOTION_CMD_STOP_GET_DECELERATION = 358
EC_T_MOTION_CMD_STOP_GET_DONE = 353
EC_T_MOTION_CMD_STOP_GET_EXECUTE = 356
EC_T_MOTION_CMD_STOP_GET_JERK = 360
EC_T_MOTION_CMD_STOP_SET_AXIS = 352
EC_T_MOTION_CMD_STOP_SET_DECELERATION = 359
EC_T_MOTION_CMD_STOP_SET_EXECUTE = 357
EC_T_MOTION_CMD_STOP_SET_JERK = 361
EC_T_MOTION_CMD_WRITE_BOOL_PARAMETER_CREATE = 900
EC_T_MOTION_CMD_WRITE_BOOL_PARAMETER_DELETE = 901
EC_T_MOTION_CMD_WRITE_BOOL_PARAMETER_EXEC_ONCYCLE = 910
EC_T_MOTION_CMD_WRITE_BOOL_PARAMETER_GET_BUSY = 903
EC_T_MOTION_CMD_WRITE_BOOL_PARAMETER_GET_EXECUTE = 904
EC_T_MOTION_CMD_WRITE_BOOL_PARAMETER_GET_PARAMETERNUMBER = 906
EC_T_MOTION_CMD_WRITE_BOOL_PARAMETER_GET_VALUE = 908
EC_T_MOTION_CMD_WRITE_BOOL_PARAMETER_SET_AXIS = 902
EC_T_MOTION_CMD_WRITE_BOOL_PARAMETER_SET_EXECUTE = 905
EC_T_MOTION_CMD_WRITE_BOOL_PARAMETER_SET_PARAMETERNUMBER = 907
EC_T_MOTION_CMD_WRITE_BOOL_PARAMETER_SET_VALUE = 909
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_CREATE = 1050
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_DELETE = 1051
```

```
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_EXEC_ONCYCLE = 1062
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_GET_BUSY = 1054
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_GET_DONE = 1053
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_GET_EXECUTE = 1055
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_GET_EXECUTIONMODE = 1060
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_GET_OUTPUTNUMBER = 1056
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_GET_VALUE = 1058
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_SET_AXIS = 1052
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_SET_EXECUTIONMODE = 1061
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_SET_OUTPUTNUMBER = 1057
EC_T_MOTION_CMD_WRITE_DIGITAL_OUTPUT_SET_VALUE = 1059
EC_T_MOTION_CMD_WRITE_PARAMETER_CREATE = 850
EC_T_MOTION_CMD_WRITE_PARAMETER_DELETE = 851
EC_T_MOTION_CMD_WRITE_PARAMETER_EXEC_ONCYCLE = 860
EC_T_MOTION_CMD_WRITE_PARAMETER_GET_BUSY = 853
EC_T_MOTION_CMD_WRITE_PARAMETER_GET_EXECUTE = 854
EC_T_MOTION_CMD_WRITE_PARAMETER_GET_PARAMETERNUMBER = 856
EC_T_MOTION_CMD_WRITE_PARAMETER_GET_VALUE = 858
EC_T_MOTION_CMD_WRITE_PARAMETER_SET_AXIS = 852
EC_T_MOTION_CMD_WRITE_PARAMETER_SET_EXECUTE = 855
EC_T_MOTION_CMD_WRITE_PARAMETER_SET_PARAMETERNUMBER = 857
EC_T_MOTION_CMD_WRITE_PARAMETER_SET_VALUE = 859
```

```
class EcWrapperPythonTypes.ECoEObjCode (value)
```

```
OB Desc Object Code
```

```
ARRAY = 8
```

```
RECORD = 9
```

```
VARIABLE = 7
```

```
class EcWrapperPythonTypes.ECoeObEntryAccess (value)
```

```
OB Entry Access
```

```
R_OP = 4
```

```
R_PREOP = 1
```

```
R_SAFEOP = 2
```

```
W_OP = 32
```

```
W_PREOP = 8
```

```
W_SAFEOP = 16
```

```
class EcWrapperPythonTypes.ECoeObEntryDescInfo (value)
```

```
OB Entry Description Information
```

```
DEFAULTVALUE = 16
```

```
MAXVALUE = 64
```

```
MINVALUE = 32
```

```
OBJACCESS = 1
OBJCATEGORY = 2
PDOMAPPING = 4
UNITTYPE = 8

class EcWrapperPythonTypes.EEscPort (value)
    ESC Ports (see defines ESC_PORT_A..D)

    PORT_A = 0
    PORT_B = 1
    PORT_C = 2
    PORT_D = 3
    PORT_INVALID = 255

class EcWrapperPythonTypes.ELinkMode (value)
    Link Layer Modi

    INTERRUPT = 1
    POLLING = 2
    UNDEFINED = 0

class EcWrapperPythonTypes.ELinkType (value)
    Link Layer Type

    BCMNETXTREME = 14
    CCAT = 8
    DPDK = 15
    EOM = 5
    I8254X = 3
    I8255X = 4
    INTELGBE = 12
    MULTIPLIER = 16
    NDIS = 9
    RTL8139 = 6
    RTL8169 = 7
    Remote = 11
    SOCKRAW = 2
    Simulator = 10
    TAP = 13
    UNDEFINED = 0
    VLAN = 17
    WINPCAP = 1

class EcWrapperPythonTypes.EMailBoxFlags (value)
    MailBoxFlags

    NONE_ = 0
    SDO_COMPLETE = 1
```

```
class EcWrapperPythonTypes.EMbxProtocol (value)
    Supported mailbox protocols

    AOE = 1
    COE = 4
    EOE = 2
    FOE = 8
    SOE = 16
    VOE = 32

class EcWrapperPythonTypes.EcDeviceState (value)
    Specifies all possible Device states

    BOOTSTRAP = 3
    INIT = 1
    OP = 8
    PREOP = 2
    SAFEOP = 4
    Unknown = 65519

class EcWrapperPythonTypes.EcRunMode (value)
    Run modes

    Daq = 11
    DaqReader = 12
    LibAccess = 13
    Master = 1
    MbxGateway = 4
    MbxGatewaySrv = 5
    Monitor = 9
    MonitorRasServer = 10
    None_ = 0
    RasClient = 3
    RasServer = 2
    SimulatorHil = 7
    SimulatorRasServer = 8
    SimulatorSil = 6

class EcWrapperPythonTypes.uint32Enum (value)
    Base class for creating enumerated constants of unsigned 32 bit integers
```

3.4 Error Codes

class EcWrapperPythonTypes.**ECError** (*value*)

Specifies EtherCAT errors codes

EC_ACCESSDENIED = 2551250966

Access Denied (e.g. master internal software error)

Type

0x98110016

Type

ISW

EC_ACYC_FRM_FREEQ_EMPTY = 2551250953

Cannot queue acyclic EtherCAT command (Acyclic command queue is full. Possible solution: Increase of configuration value dwMaxQueuedEthFrames)

Type

0x98110009

Type

ISW

EC_ADO_NOT_SUPPORTED = 2551251326

ADO for slave identification not supported (e.g. Request ID mechanism (ADO 0x134) not supported by slave)

Type

0x9811017E

Type

SLV

EC_ADS_IS_RUNNING = 2551251250

ADS adapter (Pass Through Server) is running (API call conflicts with ADS state (running))

Type

0x98110132

Type

PTS

EC_AI_ADDRESS = 2551250985

Auto increment address increment mismatch (e.g. Network information file and bus topology doesn't match any more. Error shows only, if an already recognized slave isn't present any more)

Type

0x98110029

Type

SLV

EC_AOE_ABORTED = 2551251293

Request aborted

Type

0x9811015D

Type

AoE

EC_AOE_ACCESSDENIED = 2551251297

Access denied

Type

0x98110161

Type

AoE

EC_AOE_AMS_SYNC_AMS = 2551382041

Sync AMS error

Type

0x98130019

Type

AoE

EC_AOE_AMS_SYNC_NO_INDEX_MAP = 2551382042

Sync no index map

Type

0x9813001A

Type

AoE

EC_AOE_AMS_SYNC_TIMEOUT = 2551382040

Sync Timeout

Type

0x98130018

Type

AoE

EC_AOE_AMS_SYNC_WIN32 = 2551382039

Sync Win 32 error

Type

0x98130017

Type

AoE

EC_AOE_BAD_TASK_ID = 2551382033

Bad task ID

Type

0x98130011

Type

AoE

EC_AOE_BUSY = 2551251270

Device busy

Type

0x98110146

Type

AoE

EC_AOE_CLIENTUNKNOWN = 2551251283

Notification client not registered

Type

0x98110153

Type

AoE

EC_AOE_ERROR = 2551251262

Common AoE device error

Type
0x9811013E

Type
AoE

EC_AOE_EXISTS = 2551251277
Object already exists

Type
0x9811014D

Type
AoE

EC_AOE_HOST_UNREACHABLE = 2551382044
Host unreachable

Type
0x9813001C

Type
AoE

EC_AOE_INCOMPATIBLE = 2551251276
Objects do not match

Type
0x9811014C

Type
AoE

EC_AOE_INTERNAL = 2551251298
Internal error

Type
0x98110162

Type
AoE

EC_AOE_INVALIDACCESS = 2551251266
Reading/writing not permitted

Type
0x98110142

Type
AoE

EC_AOE_INVALIDAMSFAGMENT = 2551382045
Invalid AMS fragment

Type
0x9813001D

Type
AoE

EC_AOE_INVALIDARRAYIDX = 2551251295
Invalid array index

Type
0x9811015F

Type
AoE

EC_AOE_INVALIDCLSID = 2551251290
Class ID invalid

Type
0x9811015A

Type
AoE

EC_AOE_INVALIDCONTEXT = 2551251271
Invalid context

Type
0x98110147

Type
AoE

EC_AOE_INVALIDDATA = 2551251268
Invalid parameter value(s)

Type
0x98110144

Type
AoE

EC_AOE_INVALIDGRP = 2551251264
Invalid index group

Type
0x98110140

Type
AoE

EC_AOE_INVALIDINTERFACE = 2551251289
Wrong interface required

Type
0x98110159

Type
AoE

EC_AOE_INVALIDOBJID = 2551251291
Object ID invalid

Type
0x9811015B

Type
AoE

EC_AOE_INVALIDOFFSET = 2551251265
Invalid index offset

Type
0x98110141

Type
AoE

EC_AOE_INVALIDPARM = 2551251273
Invalid parameter value(s)

Type
0x98110149

Type

AoE

EC_AOE_INVALIDSIZE = 2551251267

Parameter size not correct

Type

0x98110143

Type

AoE

EC_AOE_INVALIDSTATE = 2551251280

Server in invalid state

Type

0x98110150

Type

AoE

EC_AOE_INVALIDWATCHSIZE = 2551251285

Size for watch to big

Type

0x98110155

Type

AoE

EC_AOE_INVALID_AMS_ID = 2551251304

invalid AMS Net ID

Type

0x98110168

Type

AoE

EC_AOE_INVALID_AMS_LENGTH = 2551251303

Invalid AMS length

Type

0x98110167

Type

AoE

EC_AOE_INVALID_AMS_PORT = 2551251307

Invalid AMS port

Type

0x9811016B

Type

AoE

EC_AOE_INV_RESPONSE_SIZE = 2551251261

Invalid AoE response received

Type

0x9811013D

Type

AoE

EC_AOE_LOCKED_MEMORY = 2551382030

Allocation locked memory

Type
0x9813000E

Type
AoE

EC_AOE_LOW_INSTALL_LEVEL = 2551382037
Low installation level

Type
0x98130015

Type
AoE

EC_AOE_MAILBOX = 2551382031
Insert mailbox error

Type
0x9813000F

Type
AoE

EC_AOE_MAILBOX_FULL = 2551382047
Mailbox full

Type
0x9813001F

Type
AoE

EC_AOE_MBX_WKC_ERROR = 2551251234
Mailbox receive: working counter

Type
0x98110122

Type
AoE

EC_AOE_NOINTERFACE = 2551251288
Query interface failed

Type
0x98110158

Type
AoE

EC_AOE_NOMEMORY = 2551251272
Out of memory

Type
0x98110148

Type
AoE

EC_AOE_NOMOREHDLS = 2551251284
No more notification handles

Type
0x98110154

Type
AoE

EC_AOE_NOTFOUND = 2551251274

Not found

Type

0x9811014A

Type

AoE

EC_AOE_NOTIFYHANDINVALID = 2551251282

Notification handle invalid

Type

0x98110152

Type

AoE

EC_AOE_NOTINIT = 2551251286

Device not initialized

Type

0x98110156

Type

AoE

EC_AOE_NOTREADY = 2551251269

Device not in a ready state

Type

0x98110145

Type

AoE

EC_AOE_NO_DEBUG = 2551382038

No debug available

Type

0x98130016

Type

AoE

EC_AOE_NO_IO = 2551382034

No IO

Type

0x98130012

Type

AoE

EC_AOE_NO_LOCKED_MEMORY = 2551382046

No allocation locked memory

Type

0x9813001E

Type

AoE

EC_AOE_NO_MEMORY = 2551251308

No memory

Type

0x9811016C

Type

AoE

EC_AOE_NO_RUNTIME = 2551382029

No Rtime

Type

0x9813000D

Type

AoE

EC_AOE_PENDING = 2551251292

Request pending

Type

0x9811015C

Type

AoE

EC_AOE_PORT_CONNECTED = 2551251306

Port already connected

Type

0x9811016A

Type

AoE

EC_AOE_PORT_DISABLED = 2551251305

Port disabled

Type

0x98110169

Type

AoE

EC_AOE_PORT_NOT_CONNECTED = 2551251302

Port not connected

Type

0x98110166

Type

AoE

EC_AOE_SRVNOTSUPP = 2551251263

Service not supported by server

Type

0x9811013F

Type

AoE

EC_AOE_SYMBOLNOTACTIVE = 2551251296

Symbol not active -> release handle and try again

Type

0x98110160

Type

AoE

EC_AOE_SYMBOLNOTFOUND = 2551251278

Symbol not found

Type

0x9811014E

Type

AoE

EC_AOE_SYMBOLVERSIONINVALID = 2551251279

Symbol version invalid

Type

0x9811014F

Type

AoE

EC_AOE_SYNTAX = 2551251275

Syntax error in command or file

Type

0x9811014B

Type

AoE

EC_AOE_TARGET_MACHINE_NOT_FOUND = 2551251300

Target machine not found

Type

0x98110164

Type

AoE

EC_AOE_TARGET_PORT_NOT_FOUND = 2551251299

Target port not found

Type

0x98110163

Type

AoE

EC_AOE_TCP_SEND = 2551382043

TCP send error

Type

0x9813001B

Type

AoE

EC_AOE_TIMEOUT = 2551251287

Device has a timeout

Type

0x98110157

Type

AoE

EC_AOE_TRANSMODENOTSUPP = 2551251281

AdsTransMode not supported

Type

0x98110151

Type

AoE

EC_AOE_UNKNOWN_AMS_COMMAND = 2551382035

Unknown ADS command

Type

0x98130013

Type

AoE

EC_AOE_UNKNOWN_CMD_ID = 2551251301

Unknown command ID

Type

0x98110165

Type

AoE

EC_AOE_VENDOR_SPECIFIC = 2551251309

Vendor specific AoE device error

Type

0x9811016D

Type

AoE

EC_AOE_WARNING = 2551251294

Signal warning

Type

0x9811015E

Type

AoE

EC_AOE_WIN32 = 2551382036

Win 32 error

Type

0x98130014

Type

AoE

EC_AOE_WRONG_HMSG = 2551382032

Wrong receive HMSG

Type

0x98130010

Type

AoE

EC_BAD_CONNECTION = 2551382066

Bad connection

Type

0x98130032

EC_BUSCONFIG_MISMATCH = 2551250974

Bus configuration mismatch (e.g. Network information file and currently connected bus topology does not match)

Type

0x9811001E

Type

ENI

EC_BUSCONFIG_TOPOCHANGE = 2551251230

Bus configuration not detected, Topology changed (e.g. Topology changed while scanning bus)

Type

0x9811011E

EC_BUSY = 2551250952

Busy (e.g. EtherCAT stack is currently busy and not available to process the API request. The function may be called again later)

Type

0x98110008

Type

APP

EC_CANCEL = 2551250948

Cancel (e.g. EtherCAT stack should abort current mailbox transfer)

Type

0x98110004

Type

APP

EC_CFGFILENOTFOUND = 2551251056

Network configuration file not found (e.g. path to configuration file (XML) was wrong or the file is not available)

Type

0x98110070

Type

CFG

EC_CMD_MISSING = 2551250982

At least one EtherCAT command is missing in the received frame (e.g. received EtherCAT frame incomplete)

Type

0x98110026

Type

SLV

EC_COE_MBXRCV_WKC_ERROR = 2551250992

CoE mailbox receive: working counter (e.g. CoE mailbox couldn't be read from slave)

Type

0x98110030

Type

SLV

EC_COE_MBXSEND_WKC_ERROR = 2551250991

CoE mailbox send: working counter (e.g. CoE mailbox couldn't be read on slave, slave didn't read out mailbox since last write)

Type

0x9811002F

Type

SLV

EC_CONFIGDATAREAD = 2551250975

Error reading configuration file (e.g. Network information file could not be read)

Type

0x9811001F

Type

ENI

EC_CYCCMD_WKC_ERROR = 2551250983

Cyclic command WKC error

Type

0x98110027

EC_CYC_CMDS_OVERFLOW = 2551250988

Too many cyclic commands in XML configuration file (e.g. EC_T_INIT_MASTER_PARAMS.dwMaxAcycFramesQueued too small)

Type

0x9811002C

Type

ENI

EC_DATA_TYPE_CONVERSION_FAILED = 2551251321

Data type conversion failed

Type

0x98110179

EC_DC_REF_CLOCK_NOT_FOUND = 2551251257

Reference clock not found! May happen if reference clock is removed from network.

Type

0x98110139

Type

SLV

EC_DC_REF_CLOCK_SYNC_OUT_UNIT_DISABLED = 2551251256

DC (time loop control) unit of reference clock disabled

Type

0x98110138

EC_DC_SLAVES_BEFORE_REF_CLOCK = 2551251320

Slaves with DC configured present on bus before reference clock (e.g. The first DC Slave was not configured as potential reference clock)

Type

0x98110178

Type

ENI

EC_DLSTATUS_IRQ_TOPOCHANGED = 2551251248

Data link (DL) status interrupt because of changed topology (automatically handled by master)

Type

0x98110130

Type

SLV

EC_DUPLICATE = 2551250957

Duplicated fixed address detected (handled internally)

Type

0x9811000D

Type

ISW

EC_EEPROMASSIGNERROR = 2551251236
EEPROM assignment failed

Type
0x98110124

Type
SLV

EC_EEPROMREADERROR = 2551251057
Command error while EEPROM upload (read slave EEPROM)

Type
0x98110071

Type
SLV

EC_EEPROMRELOADERROR = 2551251216
Command error while EEPROM reload

Type
0x98110110

EC_EEPROMWRITEERROR = 2551251058
Command error while EEPROM download (write slave EEPROM)

Type
0x98110072

Type
SLV

EC_ENI_ENCRYPTED = 2551382026
Loading encrypted ENI needs OEM key

Type
0x9813000A

Type
OEM

EC_ENI_ENCRYPTION_WRONG_VERSION = 2551382025
ENI encryption algorithm version not supported

Type
0x98130009

Type
ENI, OEM

EC_ENI_NO_SAFEOP_OP_SUPPORT = 2551250976
Configuration doesn't support SAFEOP and OP requested state

Type
0x98110020

EC_EOE_MBX_WKC_ERROR = 2551251231
Mailbox receive: working counter

Type
0x9811011F

Type
EoE

EC_ERROR = 2551250944
Unspecific Error

Type

0x98110000

EC_ERROR2 = 2551382016

EC_ERROR2

EC_ERROR_LAST = 2551251327

EC_ERROR_LAST

EC_ERROR_LAST2 = 2551447551

EC_ERROR_LAST2

EC_EVAL_EXPIRED = 2551251000

Evaluation Time limit reached (e.g. License not provided and evaluation period (1 hour) of protected version exceeded)

Type

0x98110038

Type

CFG

EC_EVAL_VIOLATION = 2551250999

Configuration violates Evaluation limits (obsolete)

Type

0x98110037

Type

ENI

EC_FEATURE_DISABLED = 2551250972

Feature disabled (e.g. Application tried to perform a missing or disabled API function)

Type

0x9811001C

Type

APP

EC_FOE_ERRCODE_ACCESS = 2551251042

ERROR FoE: access denied (FoE Error Code 2 (0x8002) of slave)

Type

0x98110062

Type

SLV

EC_FOE_ERRCODE_BOOTSTRAPONLY = 2551251048

ERROR FoE: bootstrap only (FoE Error Code 8 (0x8008) of slave)

Type

0x98110068

Type

SLV

EC_FOE_ERRCODE_DISKFULL = 2551251043

ERROR FoE: disk full (FoE Error Code 3 (0x8003) of slave)

Type

0x98110063

Type

SLV

EC_FOE_ERRCODE_EXISTS = 2551251046

ERROR FoE: already exists (FoE Error Code 6 (0x8006) of slave)

Type
0x98110066

Type
SLV

EC_FOE_ERRCODE_FILE_HEAD_MISSING = 2551382017
ERROR FoE: File header does not exist (FoE Error Code 16 (0x8010) of slave)

Type
0x98130001

Type
SLV

EC_FOE_ERRCODE_FILE_INCOMPATIBLE = 2551382019
ERROR FoE: File incompatible (FoE Error Code 18 (0x8012) of slave)

Type
0x98130003

Type
SLV

EC_FOE_ERRCODE_FLASH_PROBLEM = 2551382018
ERROR FoE: Flash problem (FoE Error Code 17 (0x8011) of slave)

Type
0x98130002

Type
SLV

EC_FOE_ERRCODE_ILLEGAL = 2551251044
ERROR FoE: illegal (FoE Error Code 4 (0x8004) of slave)

Type
0x98110064

Type
SLV

EC_FOE_ERRCODE_INVALIDPASSWORD = 2551251050
ERROR FoE: no rights (FoE Error Code 10 (0x800A) of slave)

Type
0x9811006A

Type
SLV

EC_FOE_ERRCODE_INVALID_CHECKSUM = 2551251052
Wrong checksum

Type
0x9811006C

Type
FoE

EC_FOE_ERRCODE_INVALID_FIRMWARE = 2551251053
ERROR FoE: Firmware does not fit for Hardware (FoE Error Code 13 (0x800D) of slave)

Type
0x9811006D

Type
SLV

EC_FOE_ERRCODE_MAX_FILE_SIZE = 2551251322

ERROR FoE: File is bigger than max file size (e.g. Slave returned more data than the buffer provided by application can store.)

Type

0x9811017A

Type

APP

EC_FOE_ERRCODE_NOTDEFINED = 2551251040

ERROR FoE: not defined (FoE Error Code 0 (0x8000) of slave)

Type

0x98110060

Type

SLV

EC_FOE_ERRCODE_NOTFOUND = 2551251041

ERROR FoE: not found (FoE Error Code 1 (0x8001) of slave)

Type

0x98110061

Type

SLV

EC_FOE_ERRCODE_NOTINBOOTSTRAP = 2551251049

ERROR FoE: Downloaded file name is not valid in Bootstrap state (FoE Error Code 9 (0x8009) of slave)

Type

0x98110069

Type

SLV

EC_FOE_ERRCODE_NOUSER = 2551251047

ERROR FoE: no user (FoE Error Code 7 (0x8007) of slave)

Type

0x98110067

Type

SLV

EC_FOE_ERRCODE_NO_FILE = 2551251055

ERROR FoE: No file to read (FoE Error Code 15 (0x800F) of slave)

Type

0x9811006F

Type

SLV

EC_FOE_ERRCODE_PACKENO = 2551251045

ERROR FoE: packet number wrong (FoE Error Code 5 (0x8005) of slave)

Type

0x98110065

Type

SLV

EC_FOE_ERRCODE_PROGERROR = 2551251051

ERROR FoE: program error (FoE Error Code 11 (0x800B) of slave)

Type

0x9811006B

Type

SLV

EC_FOE_MBX_WKC_ERROR = 2551251232

Mailbox receive: working counter

Type

0x98110120

Type

FoE

EC_FRAMELOSS_AFTER_SLAVE = 2551251327

Frameloss after Slave (opening port destroys communication)

Type

0x9811017F

EC_FRAME_LOST = 2551250981

Frame lost, IDX mismatch (EtherCAT frame(s) lost on bus, means the response was not received. In case this error shows frequently a problem with the wiring could be the cause)

Type

0x98110025

Type

SLV

EC_IDENTIFICATIONFAILED = 2551250967

Identification failed (e.g. identification command failed)

Type

0x98110017

Type

ENI

EC_INSERTMAILBOX = 2551250963

Insert Mailbox error (internal limit MAX_QUEUED_COE_CMDS: 20)

Type

0x98110013

Type

CFG

EC_INVALIDCMD = 2551250964

Invalid Command (Unknown mailbox command code)

Type

0x98110014

Type

ISW

EC_INVALIDDATA = 2551250950

Invalid data (multiple error sources)

Type

0x98110006

Type

ISW

EC_INVALIDINDEX = 2551250946

Invalid index (e.g. CoE: invalid SDO index)

Type

0x98110002

Type

APP

EC_INVALIDOFFSET = 2551250947

Invalid offset (e.g. invalid offset while accessing Process Data Image)

Type

0x98110003

Type

ISW

EC_INVALIDPARM = 2551250955

Invalid parameter (e.g. API function called with erroneous parameter set)

Type

0x9811000B

Type

APP

EC_INVALIDSIZE = 2551250949

Invalid size

Type

0x98110005

Type

APP

EC_INVALIDSTATE = 2551250958

Invalid state (EtherCAT stack not initialized or not configured)

Type

0x9811000E

Type

ISW

EC_INVALID_DCL_MODE = 2551250984

IOCTL_EC_IOCTL_DC_LATCH_REQ_LTIMVALS invalid in DCL auto read mode (this function cannot be used if DC Latching is running in mode "Auto Read")

Type

0x98110028

Type

APP

EC_INVALID_SLAVE_STATE = 2551250986

Slave in invalid state, e.g. not in OP (API not callable in this state) (mailbox commands are not allowed in current slave state)

Type

0x9811002A

Type

APP

EC_JUNCTION_RED_LINE_BREAK = 2551382054

Junction redundancy line break

Type

0x98130026

EC_LICENSE_MISSING = 2551251001

License key invalid or missing

Type

0x98110039

EC_LINE_CROSSED = 2551251323

Line crossed (cabling wrong)

Type

0x9811017B

EC_LINE_CROSSED_SLAVE_INFO = 2551251324

Line crossed at slave (obsolete)

Type

0x9811017C

EC_LINK_DISCONNECTED = 2551250989

Ethernet link cable disconnected (e.g. EtherCAT bus segment not connected to network interface)

Type

0x9811002D

Type

SLV

EC_LOCK_CREATE_FAILED = 2551250968

Create lock failed (e.g. OsCreateLockTyped failed)

Type

0x98110018

Type

SYS

EC_MASTERCORE_INACCESSIBLE = 2551250990

Simulator core not accessible (e.g. Connection to remote server was terminated or simulator instance has been stopped on remote side)

Type

0x9811002E

Type

RAS

EC_MASTER_RED_STATE_ACTIVE = 2551382053

Master Redundancy State is ACTIVE (e.g. API not allowed in current Master Redundancy State)

Type

0x98130025

Type

APP

EC_MASTER_RED_STATE_INACTIVE = 2551382052

Master Redundancy State is INACTIVE (e.g. API not allowed in current Master Redundancy State)

Type

0x98130024

Type

APP

EC_MAX_BUS_SLAVES_EXCEEDED = 2551251311

Error: Maximum number of bus slave has been exceeded (The maximum number of preallocated bus slave objects is too small. The maximum number can be adjusted by the master initialization parameter EC_T_INITMASTERPARMS.dwMaxBusSlaves)

Type

0x9811016F

Type
CFG

EC_MBXERR_INVALIDCHANNEL = 2551251314

Mailbox error: Field contains wrong value (Slave error mailbox return value: 0x03)

Type
0x98110172

Type
SLV

EC_MBXERR_INVALIDHEADER = 2551251316

Mailbox error: The mailbox protocol header of the mailbox protocol is wrong (Slave error mailbox return value: 0x05)

Type
0x98110174

Type
SLV

EC_MBXERR_INVALIDSIZE = 2551251319

Mailbox error: The length of data is inconsistent (Slave error mailbox return value: 0x08)

Type
0x98110177

Type
SLV

EC_MBXERR_NOMOREMEMORY = 2551251318

Mailbox error: Mailbox protocol can not be processed because of limited resources (Slave error mailbox return value: 0x07)

Type
0x98110176

Type
SLV

EC_MBXERR_SERVICENOTSUPPORTED = 2551251315

Mailbox error: The addressed service in the mailbox protocol is not supported (Slave error mailbox return value: 0x04)

Type
0x98110173

Type
SLV

EC_MBXERR_SIZETOOSHORT = 2551251317

Mailbox error: Length of received mailbox data is too short (Slave error mailbox return value: 0x06)

Type
0x98110175

Type
SLV

EC_MBXERR_SYNTAX = 2551251312

Mailbox error: Syntax of 6 octet Mailbox header is wrong (Slave error mailbox return value: 0x01)

Type
0x98110170

Type
SLV

EC_MBXERR_UNSUPPORTEDPROTOCOL = 2551251313

Mailbox error: The Mailbox protocol is not supported (Slave error mailbox return value: 0x02)

Type

0x98110171

Type

SLV

EC_MBX_CMD_WKC_ERROR = 2551251259

Mailbox command working counter error (e.g. Mailbox init command Retry Count exceeded)

Type

0x9811013B

Type

SLV

EC_MBX_ERROR_TYPE = 2551251237

Unknown mailbox error code received in mailbox

Type

0x98110125

Type

SLV

EC_MCSM_FATAL_ERROR = 2551250979

Fatal internal McSm (master control state machine is in an undefined state)

Type

0x98110023

Type

ISW

EC_NOERROR = 0

No Error

Type

0x00000000

EC_NOMEMORY = 2551250954

No memory left (e.g. memory full / fragmented)

Type

0x9811000A

Type

CFG

EC_NOTFOUND = 2551250956

Not found (e.g. Network Information File ENI not found or API called with invalid slave ID)

Type

0x9811000C

Type

APP

EC_NOTREADY = 2551250951

Not ready (multiple error sources)

Type

0x98110007

Type

ISW

EC_NOTSUPPORTED = 2551250945

Feature not supported (e.g. function or property not available)

Type

0x98110001

Type

APP

EC_NO_AOE_SUPPORT = 2551251260

AoE: Protocol not supported (e.g. Application calls AoE-API although not implemented at slave)

Type

0x9811013C

Type

APP / SLV

EC_NO_COE_SUPPORT = 2551250994

CoE protocol not supported (e.g. Configuration error or slave information file doesn't match slave firmware)

Type

0x98110032

Type

ENI

EC_NO_EOE_SUPPORT = 2551250995

EoE protocol not supported (e.g. Configuration error or slave information file doesn't match slave firmware)

Type

0x98110033

Type

ENI

EC_NO_FOE_SUPPORT = 2551250996

FoE protocol not supported (e.g. Configuration error or slave information file doesn't match slave firmware)

Type

0x98110034

Type

ENI

EC_NO_FOE_SUPPORT_BS = 2551251215

ERROR FoE: Protocol not supported in boot strap (e.g. Application requested FoE in Bootstrap although slave does not support this)

Type

0x9811010F

Type

APP

EC_NO_MBX_SUPPORT = 2551250993

No mailbox support (e.g. Slave does not support mailbox access)

Type

0x98110031

Type

APP

EC_NO_SOE_SUPPORT = 2551250997

SoE protocol not supported (e.g. Configuration error or slave information file doesn't match slave firmware)

Type

0x98110035

Type

ENI

EC_NO_VOE_SUPPORT = 2551250998

VoE protocol not supported (e.g. Configuration error or slave information file doesn't match slave firmware)

Type

0x98110036

Type

ENI

EC_OEM_KEY_MISMATCH = 2551382027

OEM key mismatch

Type

0x9813000B

Type

RAS, APP

EC_OEM_KEY_MISSING = 2551382028

OEM key access needs OEM key set (e.g. Application must call esSetOemKey (HiL) or set EC_T_LINK_PARAMS_SIMULATOR::qwOemKey (SiL))

Type

0x9813000C

Type

APP

EC_OEM_SIGNATURE_MISMATCH = 2551382024

Manufacturer signature mismatch

Type

0x98130008

Type

ENI, OEM

EC_OPENFAILED = 2551250961

Open failed

Type

0x98110011

Type

ISW

EC_PDIWATCHDOG = 2551382065

PDI watchdog expired

Type

0x98130031

EC_PORTCLOSE = 2551251062

Port close failed

Type

0x98110076

EC_PORTOPEN = 2551251063

Port open failed

Type

0x98110077

EC_PRODKEY_INVALID = 2551250970

Product Key Invalid (e.g. application using protected version of the stack, which stops operation after the evaluation time limit reached if a license is not provided)

Type

0x9811001A

Type

CFG

EC_PTS_IS_NOT_RUNNING = 2551251249

Pass Through Server is not running (Pass-Through-Server was tried to be enabled/disabled or stopped without being started)

Type

0x98110131

Type

PTS

EC_PTS_IS_RUNNING = 2551251250

Pass Through Server is running (obsolete, replaced by EC_E_ADS_IS_RUNNING)

Type

0x98110132

Type

PTS

EC_PTS_LL_MODE_NOT_SUPPORTED = 2551251254

The Link Layer mode is not supported by the Pass Through Server (e.g. The Master is running in interrupt mode but the Pass-Through-Server only supports polling mode)

Type

0x98110136

Type

PTS

EC_PTS_NOT_ENABLED = 2551251253

The Pass Through Server is running but not enabled

Type

0x98110135

Type

PTS

EC_PTS SOCK_BIND_FAILED = 2551251252

The Pass Through Server could not bind the IP address with a socket (e.g. Possibly because the IPaddress (and Port) is already in use or the IP-address does not exist)

Type

0x98110134

Type

PTS

EC_PTS_THREAD_CREATE_FAILED = 2551251251

Could not start the Pass Through Server

Type

0x98110133

Type

PTS

EC_REDLINEBREAK = 2551251238

Redundancy line break (e.g. cable break between slaves or between EtherCAT network adapter and first slave)

Type

0x98110126

Type

SLV

EC_S2SMBX_DEST_SLAVE_NOT_FOUND = 2551382051

Destination Slave not found

Type

0x98130023

Type

S2S

EC_S2SMBX_NOT_CONFIGURED = 2551382048

Not Configured

Type

0x98130020

Type

S2S

EC_S2SMBX_NO_DESCRIPTOR = 2551382050

No Descriptor

Type

0x98130022

Type

S2S

EC_S2SMBX_NO_MEMORY = 2551382049

No Memory

Type

0x98130021

Type

S2S

EC_SDO_ABORTCODE_ACCESS = 2551251015

SDO: Unsupported access to an object (CoE abort code 0x06010000 of slave)

Type

0x98110047

Type

SLV

EC_SDO_ABORTCODE_BLK_SIZE = 2551251011

SDO: Invalid block size (block mode only) (CoE abort code 0x05040002 of slave)

Type

0x98110043

Type

SLV

EC_SDO_ABORTCODE_CA_TYPE_MISM = 2551382021

SDO: Complete access not supported for objects of variable length such as ENUM object types (CoE abort code 0x06010004 of slave)

Type

0x98130005

Type

SLV

EC_SDO_ABORTCODE_CCS_SCS = 2551251010

SDO: Client/server command specifier not valid or unknown (CoE abort code 0x05040001 of slave)

Type

0x98110042

Type

SLV

EC_SDO_ABORTCODE_CRC = 2551251013

SDO: CRC error (block mode only) (CoE abort code 0x05040004 of slave)

Type

0x98110045

Type

SLV

EC_SDO_ABORTCODE_DATA_LENGTH_NOT_MATCH = 2551251024

SDO: Data type does not match, length of service parameter does not match (CoE abort code 0x06070010 of slave)

Type

0x98110050

Type

SLV

EC_SDO_ABORTCODE_DATA_LENGTH_TOO_HIGH = 2551251025

SDO: Data type does not match, length of service parameter too high (CoE abort code 0x06070012 of slave)

Type

0x98110051

Type

SLV

EC_SDO_ABORTCODE_DATA_LENGTH_TOO_LOW = 2551251026

SDO: Data type does not match, length of service parameter too low (CoE abort code 0x06070013 of slave)

Type

0x98110052

Type

SLV

EC_SDO_ABORTCODE_DICTIONARY = 2551251036

SDO: Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error) (CoE abort code 0x08000023 of slave)

Type

0x9811005C

Type

SLV

EC_SDO_ABORTCODE_FIRST = 2551251008
EC_SDO_ABORTCODE_FIRST

EC_SDO_ABORTCODE_GENERAL = 2551251032
SDO: General error (CoE abort code 0x08000000 of slave)

Type
0x98110058

Type
SLV

EC_SDO_ABORTCODE_HARDWARE = 2551251023
SDO: Access failed due to a hardware error (CoE abort code 0x06060000 of slave)

Type
0x9811004F

Type
SLV

EC_SDO_ABORTCODE_INDEX = 2551251018
SDO: Object does not exist in the object dictionary (CoE abort code 0x06020000 of slave)

Type
0x9811004A

Type
SLV

EC_SDO_ABORTCODE_I_INCOMP = 2551251022
SDO: General internal incompatibility in the device (CoE abort code 0x06040047 of slave)

Type
0x9811004E

Type
SLV

EC_SDO_ABORTCODE_LAST = 2551251038
EC_SDO_ABORTCODE_LAST

EC_SDO_ABORTCODE_MEMORY = 2551251014
SDO: Out of memory (CoE abort code 0x05040005 of slave)

Type
0x98110046

Type
SLV

EC_SDO_ABORTCODE_MINMAX = 2551251031
SDO: Maximum value is less than minimum value (CoE abort code 0x06090036 of slave)

Type
0x98110057

Type
SLV

EC_SDO_ABORTCODE_MODULE_ID_LIST_NOT_MATCH = 2551251038
Detected Module Ident List (0xF030) and Configured Module Ident list (0xF050) does not match

Type
0x9811005E

EC_SDO_ABORTCODE_OBJ_TOO_BIG = 2551382022
SDO: Object length exceeds mailbox size (CoE abort code 0x06010005 of slave)

Type
0x98130006

Type
SLV

EC_SDO_ABORTCODE_OFFSET = 2551251027
SDO: Sub-index does not exist (CoE abort code 0x06090011 of slave)

Type
0x98110053

Type
SLV

EC_SDO_ABORTCODE_PDO_LEN = 2551251020
SDO: The number and length of the objects to be mapped would exceed PDO length (CoE abort code 0x06040042 of slave)

Type
0x9811004C

Type
SLV

EC_SDO_ABORTCODE_PDO_MAP = 2551251019
SDO: Object cannot be mapped to the PDO (CoE abort code 0x06040041 of slave)

Type
0x9811004B

Type
SLV

EC_SDO_ABORTCODE_PDO_MAPPED = 2551382023
SDO: Object mapped to RxPDO, SDO Download blocked (CoE abort code 0x06010006 of slave)

Type
0x98130007

Type
SLV

EC_SDO_ABORTCODE_P_INCOMP = 2551251021
SDO: General parameter incompatibility reason (CoE abort code 0x06040043 of slave)

Type
0x9811004D

Type
SLV

EC_SDO_ABORTCODE_READONLY = 2551251017
SDO: Attempt to write a read only object (CoE abort code 0x06010002 of slave)

Type
0x98110049

Type
SLV

EC_SDO_ABORTCODE_SEQNO = 2551251012
SDO: Invalid sequence number (block mode only) (CoE abort code 0x05040003 of slave)

Type
0x98110044

Type
SLV

EC_SDO_ABORTCODE_SI_NOT_WRITTEN = 2551382020
SDO: Sub Index cannot be written, SIO must be 0 for write access (CoE abort code 0x06010003 of slave)

Type
0x98130004

Type
SLV

EC_SDO_ABORTCODE_TIMEOUT = 2551251009
SDO: Protocol timed out (CoE abort code 0x05040000 of slave)

Type
0x98110041

Type
SLV

EC_SDO_ABORTCODE_TOGGLE = 2551251008
SDO: Toggle bit not alternated (CoE abort code 0x05030000 of slave)

Type
0x98110040

Type
SLV

EC_SDO_ABORTCODE_TRANSFER = 2551251033
SDO: Data cannot be transferred or stored to the application (CoE abort code 0x08000020 of slave)

Type
0x98110059

Type
SLV

EC_SDO_ABORTCODE_TRANSFER_DEVICE_STATE = 2551251035
SDO: Data cannot be transferred or stored to the application because of the present device state (CoE abort code 0x08000022 of slave)

Type
0x9811005B

Type
SLV

EC_SDO_ABORTCODE_TRANSFER_LOCAL_CONTROL = 2551251034
SDO: Data cannot be transferred or stored to the application because of local control (CoE abort code 0x08000021 of slave)

Type
0x9811005A

Type
SLV

EC_SDO_ABORTCODE_UNKNOWN = 2551251037
SDO: Unknown code (Unknown CoE abort code of slave)

Type
0x9811005D

Type
SLV

EC_SDO_ABORTCODE_VALUE_RANGE = 2551251028
SDO: Value range of parameter exceeded (only for write access) (CoE abort code 0x06090030 of slave)

Type
0x98110054

Type
SLV

EC_SDO_ABORTCODE_VALUE_TOO_HIGH = 2551251029
SDO: Value of parameter written too high (CoE abort code 0x06090031 of slave)

Type
0x98110055

Type
SLV

EC_SDO_ABORTCODE_VALUE_TOO_LOW = 2551251030
SDO: Value of parameter written too low (CoE abort code 0x06090032 of slave)

Type
0x98110056

Type
SLV

EC_SDO_ABORTCODE_WRITEONLY = 2551251016
SDO: Attempt to read a write only object (CoE abort code 0x06010001 of slave)

Type
0x98110048

Type
SLV

EC_SENDFAILED = 2551250962
Frame send failed

Type
0x98110012

Type
LLA

EC_SHADOW_MEMORY = 2551250973
Shadow memory requested in wrong mode

Type
0x9811001D

EC_SIGNATURE_MISMATCH = 2551382064
Signature mismatch

Type
0x98130030

EC_SLAVECTRLRESETERROR = 2551251217
Command error while Reset Slave Controller

Type
0x98110111

EC_SLAVE_ERROR = 2551250980
Slave error (e.g. A slave error was detected. See also EC_NOTIFY_STATUS_SLAVE_ERROR and EC_NOTIFY_SLAVE_ERROR_STATUS_INFO)

Type
0x98110024

Type
SLV

EC_SLAVE_NOT_ADDRESSABLE = 2551250987

Station address lost (or slave missing) - FPRD to AL_STATUS failed (e.g. Slave had a power cycle)

Type

0x9811002B

Type

SLV

EC_SLAVE_NOT_PRESENT = 2551251214

command not executed (slave not present on bus) (e.g. slave disappeared or was never present)

Type

0x9811010E

Type

APP / SLV

EC_SOCKET_DISCONNECTED = 2551251325

Socket disconnected (e.g. IP connection terminated or lost)

Type

0x9811017D

Type

RAS

EC_SOE_ERRCODE_NO_ELEM_ADR = 2551251213

No element addressed

Type

0x9811010D

Type

ERROR SoE

EC_SOE_ERRORCODE_ALREADY_ACTIVE = 2551251103

Command already active

Type

0x9811009F

Type

ERROR SoE

EC_SOE_ERRORCODE_ATTR_UNCHANGE = 2551251074

Attribute unchangeable

Type

0x98110082

Type

ERROR SoE

EC_SOE_ERRORCODE_ATTR_WR_PROT = 2551251075

Attribute currently write-protected

Type

0x98110083

Type

ERROR SoE

EC_SOE_ERRORCODE_BUFFER_FULL = 2551251206

RX buffer full (EtherCAT call with too small data-buffer)

Type

0x98110106

Type

ERROR SoE

EC_SOE_ERRORCODE_CMD_NOT_AVAIL = 2551251201

Command not available (in this phase)

Type

0x98110101

Type

ERROR SoE

EC_SOE_ERRORCODE_CMD_NOT_AVAIL1 = 2551251202

Command not available (invalid parameter)

Type

0x98110102

Type

ERROR SoE

EC_SOE_ERRORCODE_DATA_INCOR = 2551251098

Data item incorrect

Type

0x9811009A

Type

ERROR SoE

EC_SOE_ERRORCODE_DATA_MAX_LIMIT = 2551251097

Data item exceeds maximum input value limit

Type

0x98110099

Type

ERROR SoE

EC_SOE_ERRORCODE_DATA_MIN_LIMIT = 2551251096

Data item less than minimum input value limit

Type

0x98110098

Type

ERROR SoE

EC_SOE_ERRORCODE_DATA_NOT_EXIST = 2551251091

Data item does not exist

Type

0x98110093

Type

ERROR SoE

EC_SOE_ERRORCODE_DATA_OVERSIZE = 2551251093

Data item oversize in transmission

Type

0x98110095

Type

ERROR SoE

EC_SOE_ERRORCODE_DATA_UNCHANGE = 2551251094

Data item unchangeable

Type
0x98110096

Type
ERROR SoE

EC_SOE_ERRORCODE_DATA_UNDERSIZE = 2551251092

Data item undersize in transmission

Type
0x98110094

Type
ERROR SoE

EC_SOE_ERRORCODE_DATA_WR_PROT = 2551251095

Data item currently write-protected

Type
0x98110097

Type
ERROR SoE

EC_SOE_ERRORCODE_DEFAULT_LONG = 2551251209

Default value transmission too long

Type
0x98110109

Type
ERROR SoE

EC_SOE_ERRORCODE_DEFAULT_WP = 2551251210

Default value cannot be changed, read only

Type
0x9811010A

Type
ERROR SoE

EC_SOE_ERRORCODE_DRIVE_NO = 2551251203

Response drive number not identical with requested drive number

Type
0x98110103

Type
ERROR SoE

EC_SOE_ERRORCODE_FRAGMENT_LOST = 2551251205

At least one fragment lost

Type
0x98110105

Type
ERROR SoE

EC_SOE_ERRORCODE_GENERAL_ERROR = 2551251212

General error

Type
0x9811010C

Type
ERROR SoE

EC_SOE_ERRORCODE_IDN = 2551251204
Response IDN not identical with requested IDN

Type
0x98110104

Type
ERROR SoE

EC_SOE_ERRORCODE_INVALID_ACCESS = 2551251064
Invalid access to element 0

Type
0x98110078

Type
ERROR SoE

EC_SOE_ERRORCODE_INVL_ACC_ELEM1 = 2551251066
Invalid access to element 1

Type
0x9811007A

Type
ERROR SoE

EC_SOE_ERRORCODE_INVL_DRIVE_NO = 2551251211
Invalid drive number

Type
0x9811010B

Type
ERROR SoE

EC_SOE_ERRORCODE_INVL_INDIRECT = 2551251101
Invalid indirect

Type
0x9811009D

Type
ERROR SoE

EC_SOE_ERRORCODE_MAX_NOT_EXIST = 2551251086
Maximum input value does not exist

Type
0x9811008E

Type
ERROR SoE

EC_SOE_ERRORCODE_MAX_OVERSIZE = 2551251088
Maximum input value oversize in transmission

Type
0x98110090

Type
ERROR SoE

EC_SOE_ERRORCODE_MAX_UNCHANGE = 2551251089
Maximum input value unchangeable

Type
0x98110091

Type

ERROR SoE

EC_SOE_ERRORCODE_MAX_UNDERSIZE = 2551251087

Maximum input value undersize in transmission

Type

0x9811008F

Type

ERROR SoE

EC_SOE_ERRORCODE_MAX_WR_PROT = 2551251090

Maximum input value currently write-protected

Type

0x98110092

Type

ERROR SoE

EC_SOE_ERRORCODE_MIN_NOT_EXIST = 2551251081

Minimum input value does not exist

Type

0x98110089

Type

ERROR SoE

EC_SOE_ERRORCODE_MIN_OVERSIZE = 2551251083

Minimum input value oversize in transmission

Type

0x9811008B

Type

ERROR SoE

EC_SOE_ERRORCODE_MIN_UNCHANGE = 2551251084

Minimum input value unchangeable

Type

0x9811008C

Type

ERROR SoE

EC_SOE_ERRORCODE_MIN_UNDERSIZE = 2551251082

Minimum input value undersize in transmission

Type

0x9811008A

Type

ERROR SoE

EC_SOE_ERRORCODE_MIN_WR_PROT = 2551251085

Minimum input value currently write-protected

Type

0x9811008D

Type

ERROR SoE

EC_SOE_ERRORCODE_NAME_NOT_EXIST = 2551251067

Name does not exist

Type
0x9811007B

Type
ERROR SoE

EC_SOE_ERRORCODE_NAME_OVERSIZE = 2551251069
Name oversize in transmission

Type
0x9811007D

Type
ERROR SoE

EC_SOE_ERRORCODE_NAME_UNCHANGE = 2551251070
Name unchangeable

Type
0x9811007E

Type
ERROR SoE

EC_SOE_ERRORCODE_NAME_UNDERSIZE = 2551251068
Name undersize in transmission

Type
0x9811007C

Type
ERROR SoE

EC_SOE_ERRORCODE_NAME_WR_PROT = 2551251071
Name currently write-protected

Type
0x9811007F

Type
ERROR SoE

EC_SOE_ERRORCODE_NOT_EXIST = 2551251065
Does not exist

Type
0x98110079

Type
ERROR SoE

EC_SOE_ERRORCODE_NOT_INTERRUPT = 2551251200
Command not interruptible

Type
0x98110100

Type
ERROR SoE

EC_SOE_ERRORCODE_NO_DATA = 2551251207
No data state

Type
0x98110107

Type
ERROR SoE

EC_SOE_ERRORCODE_NO_DEFAULT_VALUE = 2551251208

No default value

Type

0x98110108

Type

ERROR SoE

EC_SOE_ERRORCODE_OVERS_TRANS = 2551251073

Attribute oversize in transmission

Type

0x98110081

Type

ERROR SoE

EC_SOE_ERRORCODE_PASWD_PROT = 2551251099

Data item protected by password

Type

0x9811009B

Type

ERROR SoE

EC_SOE_ERRORCODE_TEMP_UNCHANGE = 2551251100

Data item temporary unchangeable (in AT or MDT)

Type

0x9811009C

Type

ERROR SoE

EC_SOE_ERRORCODE_TEMP_UNCHANGE1 = 2551251102

Data item temporary unchangeable (parameter or opmode)

Type

0x9811009E

Type

ERROR SoE

EC_SOE_ERRORCODE_UNDERS_TRANS = 2551251072

Attribute undersize in transmission

Type

0x98110080

Type

ERROR SoE

EC_SOE_ERRORCODE_UNIT_NOT_EXIST = 2551251076

Unit does not exist

Type

0x98110084

Type

ERROR SoE

EC_SOE_ERRORCODE_UNIT_OVERSIZE = 2551251078

Unit oversize in transmission

Type

0x98110086

Type

ERROR SoE

EC_SOE_ERRORCODE_UNIT_UNCHANGE = 2551251079

Unit unchangeable

Type

0x98110087

Type

ERROR SoE

EC_SOE_ERRORCODE_UNIT_UNERSIZE = 2551251077

Unit undersize in transmission

Type

0x98110085

Type

ERROR SoE

EC_SOE_ERRORCODE_UNIT_WR_PROT = 2551251080

Unit currently write-protected

Type

0x98110088

Type

ERROR SoE

EC_SOE_MBX_WKC_ERROR = 2551251233

mailbox receive: working counter

Type

0x98110121

Type

SoE

EC_SYSDRIVERMISSING = 2551251218

Cannot open system driver (e.g. system driver was not loaded)

Type

0x98110112

Type

SYS

EC_TIMEOUT = 2551250960

Timeout

Type

0x98110010

EC_TIMEOUT_WAITING_FOR_DC = 2551382056

Timeout waiting for DC

Type

0x98130028

EC_TIMEOUT_WAITING_FOR_DCM = 2551382057

Timeout waiting for DCM

Type

0x98130029

EC_TIMER_LIST_FULL = 2551250959

Cannot add slave to timer list (slave timer list full)

Type

0x9811000F

Type

ISW

EC_UNKNOWN_MBX_PROTOCOL = 2551250965

Unknown Mailbox Protocol Command (Unknown Mailbox protocol or mailbox command with unknown protocol association)

Type

0x98110015

Type

ISW

EC_VALIDATION_ERROR = 2551382055

Validation error (validation data mismatch)

Type

0x98130027

EC_VERSION_MISMATCH = 2551250969

Version mismatch for loaded library

Type

0x98110019

Type

APP

EC_VOE_MBX_WKC_ERROR = 2551251235

VoE mailbox send: working counter (VoE mailbox couldn't be written)

Type

0x98110123

Type

SLV

EC_VOE_NO_MBX_RECEIVED = 2551251255

No VoE mailbox received yet from specific slave

Type

0x98110137

Type

SLV

EC_WRONG_FORMAT = 2551250971

Wrong configuration format (e.g. Network information file empty or malformed), SLV: Malformed EEPROM content

Type

0x9811001B

Type

ENI

EC_XML_ALSTATUS_READ_MISSING = 2551250978

AL_STATUS register read missing in XML file for at least one state (e.g. Read of AL Status register is missing in cyclic part of given network information file)

Type

0x98110022

Type

ENI

EC_XML_AOE_NETID_INVALID = 2551251310

AoE: Invalid NetID (e.g. Error from Configuration Tool)

Type

0x9811016E

Type

ENI

EC_XML_CYCCMDS_MISSING = 2551250977

Cyclic commands are missing (e.g. Network information file does not contain cyclic commands)

Type

0x98110021

Type

ENI

EC_XML_CYCCMDS_SIZEMISMATCH = 2551251059

Cyclic command wrong size (too long) (size in network configuration file (XML) does not match size of process data)

Type

0x98110073

Type

ENI

EC_XML_DC_CYCCMDS_MISSING = 2551251241

DC enabled and DC cyclic commands missing (e.g. access to 0x0900)

Type

0x98110129

EC_XML_INCONSISTENT = 2551382067

Inconsistent content

Type

0x98130033

Type

ENI

EC_XML_INVALID_CMD_WITH_RED = 2551251239

Invalid EtherCAT command in cyclic frame with redundancy (e.g. BRW commands are not allowed with redundancy)

Type

0x98110127

Type

ENI

EC_XML_INVALID_INP_OFF = 2551251060

Invalid input offset in cyclic command, please check InputOffs

Type

0x98110074

Type

ENI

EC_XML_INVALID_OUT_OFF = 2551251061

Invalid output offset in cyclic command, please check OutputOffs

Type

0x98110075

Type

ENI

EC_XML_PREV_PORT_MISSING = 2551251240

<PreviousPort>-tag is missing (e.g. 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)

Type

0x98110128

Type

ENI

EMRAS_ACCESSLESS = 2551251336

No access to this call at this access level (e.g. a higher SPoC access level is needed to use the called Remote API function)

Type

0x98110188

Type

RAS

EMRAS_ACCESS_NOT_FOUND = 2551251352

Access not configured for this call (e.g. SPoC access configuration missing)

Type

0x98110198

Type

RAS

EMRAS_CLIENTLOGON = 2551251348

Client logged on

Type

0x98110194

Type

RAS Server

EMRAS_CLNTDISC = 2551251351

Client disconnect

Type

0x98110197

Type

RAS

EMRAS_ERROR = 2551251328

Unspecific RAS Error

Type

0x98110180

EMRAS_ERROR_LAST = 2551251391

EMRAS_ERROR_LAST

EMRAS_INVALIDACCESSCONFIG = 2551251335

Access configuration is invalid (e.g. SPoC access configuration invalid)

Type

0x98110187

Type

RAS

EMRAS_INVALIDCOOKIE = 2551251329

Invalid Cookie (e.g.obsolete)

Type

0x98110181

Type

RAS

EMRAS_INVALIDDATARECEIVED = 2551251337

Invalid data received (communication corrupted)

Type

0x98110189

Type

RAS

EMRAS_INVALIDVERSION = 2551251334

Invalid Version (Connection reject because of using mismatching protocol versions on client and server side)

Type

0x98110186

Type

RAS

EMRAS_LOGONCANCELLED = 2551251332

Logon canceled (Server-side connection reject while opening a client connection.)

Type

0x98110184

Type

RAS

EMRAS_MULSRVDISMULCON = 2551251331

Connect 2nd server denied because Multi Server support is disabled (obsolete)

Type

0x98110183

Type

RAS

EMRAS_RECONEXPIRED = 2551251347

Reconnect expired (obsolete)

Type

0x98110193

Type

RAS

EMRAS_RECONNECT = 2551251349

obsolete

Type

0x98110195

Type

RAS

EMRAS_SERVERSTOPPED = 2551251345

Server stopped (e.g. connection dropped because of Remote API Server stop)

Type

0x98110191

Type

RAS

EMRAS SOCKCHANGE = 2551251350

Socket exchanged after reconnect (obsolete)

Type

0x98110196

Type

RAS

EMRAS_TLS_CERTIFICATE_ERROR = 2551251356

TLS certificate error

Type

0x9811019C

Type

RAS

EMRAS_TLS_HANDSHAKE_FAILED = 2551251358

TLS handshake failed

Type

0x9811019E

Type

RAS

EMRAS_TLS_PRIVATEKEY_ERROR = 2551251357

TLS private key error

Type

0x9811019D

Type

RAS

EMRAS_TOKEN_DENIED = 2551251355

Token denied

Type

0x9811019B

Type

RAS

EMRAS_TOKEN_INVALID = 2551251354

Token invalid

Type

0x9811019A

Type

RAS

EMRAS_TOKEN_MISSING = 2551251353

Token missing

Type

0x98110199

Type

RAS

EMRAS_WDEXPIRED = 2551251346

Watchdog expired (e.g. connection dropped because of missing keep-alive messages)

Type
0x98110192

Type
RAS

4 Examples

4.1 CoeSdoUpload

```
def CoeSdoUploadExample(self):
    wSlaveAddress = 1012 # slave with station address 1012
    wObIndex = 0x1018 # object index 0x1018
    byObSubIndex = 1 # subindex 1 (Vendor ID, which is from type UDINT with 4_
↳bytes)
    dwDataLen = 4 # 4 bytes
    pbyData = [0] * dwDataLen
    out_dwOutDataLen = CECWrapperPythonOutParam()
    dwTimeout = 5000 # timeout 5s
    dwFlags = EMailBoxFlags.NONE_ # no complete access

    dwSlaveId = self.m_oEcWrapper.GetSlaveId(wSlaveAddress)
    eRes = self.m_oEcWrapper.CoeSdoUpload(dwSlaveId, wObIndex, byObSubIndex,
↳pbyData, dwDataLen, out_dwOutDataLen, dwTimeout, dwFlags)
    if eRes != ECErrror.EC_NOERROR:
        self.LogMasterError("CoeSdoUpload failed: ", eRes)
        return

    dwOutDataLen = out_dwOutDataLen.value
    szValue = CECWrapperPython.ConvValueToString(DN_EC_T_DEFTYPE.ARRAY_OF_BYTE,
↳pbyData)
    self.LogInfo("CoeSdoUpload success: " + szValue) # 02 00 00 00
```

4.2 CoeSdoDownload

```
def CoeSdoDownloadExample(self):
    wSlaveAddress = 1012 # slave with station address 1012
    wObIndex = 0x40A2 # object index 0x40A2
    byObSubIndex = 2 # subindex 2
    dwValue = 123 # value 123
    dwTimeout = 5000 # timeout 5s
    dwFlags = EMailBoxFlags.NONE_ # no complete access

    # convert UNSIGNED32 value to 4 bytes
    out_pbyData = CECWrapperPythonOutParam()
    eRes = CECWrapperPython.ConvValueToBytes(DN_EC_T_DEFTYPE.UNSIGNED32, dwValue,
↳out_pbyData)
    if eRes != ECErrror.EC_NOERROR:
        LogMasterError("ConvValueToBytes failed: ", eRes)
        return

    pbyData = out_pbyData.value
    dwSlaveId = self.m_oEcWrapper.GetSlaveId(wSlaveAddress)
    eRes = self.m_oEcWrapper.CoeSdoDownload(dwSlaveId, wObIndex, byObSubIndex,
↳pbyData, len(pbyData), dwTimeout, dwFlags)
    if eRes != ECErrror.EC_NOERROR:
        self.LogMasterError("CoeSdoDownload failed: ", eRes)
        return

    self.LogInfo("CoeSdoDownload success")
```

4.3 ReadSlaveEEPROM

```
def ReadSlaveEEPROMExample(self):
    bFixedAddressing = True # enable fixed addressing
    wSlaveAddress = 1006 # slave with station address 1006
    wEEPROMStartOffset = 0x0 # offset 0
    dwReadLen = 10 # 10 bytes
    pwReadData = [0] * dwReadLen
    out_dwNumOutData = CECWrapperPythonOutParam()
    dwTimeout = 2000 # timeout 2s

    eRes = self.m_oEcWrapper.ReadSlaveEEPROM(bFixedAddressing, wSlaveAddress,
    ↪wEEPROMStartOffset, pwReadData, dwReadLen, out_dwNumOutData, dwTimeout)
    if eRes != ECError.EC_NOERROR:
        self.LogMasterError("ReadSlaveEEPROM failed: ", eRes)
        return

    dwNumOutData = out_dwNumOutData.value
    szValue = CECWrapperPython.ConvValueToString(DN_EC_T_DEFTYPE.ARRAY_OF_BYTE,
    ↪pwReadData)
    self.LogInfo("ReadSlaveEEPROM success: " + szValue)
```

4.4 FoeFileDownload

```
def FoeFileDownloadExample(self):
    wSlaveAddress = 1012 # slave with station address 1012
    szFileName = "foe.txt" # filename of file in slave
    pbyData = [ 1, 2, 3, 4 ] # 4 bytes
    dwPassword = 0 # password is 0
    dwTimeout = 2000 # timeout 2s

    dwSlaveId = self.m_oEcWrapper.GetSlaveId(wSlaveAddress)
    eRes = self.m_oEcWrapper.FoeFileDownload(dwSlaveId, szFileName,
    ↪len(szFileName), pbyData, len(pbyData), dwPassword, dwTimeout)
    if eRes != ECError.EC_NOERROR:
        self.LogMasterError("FoeFileDownload failed: ", eRes)
        return

    self.LogInfo("FoeFileDownload success")
```

5 FAQ

PyQt5 cannot be installed on Ubuntu 14.04 x64, because it requires Python 3.5. How can I install it?

It can be installed by calling

```
$ sudo apt-get install python3-pyqt5
```

I installed Python and the demo crashes with strange errors. What can I do?

This might be a problem of mixing x86 with x64 binaries. Verify that if you have installed the Python runtime for x64 bit, please install also EC-Master for x64 bit.

Python Module Index

e

`EcWrapperPythonTypes`, 124