acontis technologies GmbH

SOFTWARE

EC-Simulator

Swift Programming Interface

V3.3

AT3509 **Edition: February 27, 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 Code . . . 6
2.2 Wrapper . . . 6
2.2.1 Modules . . . 6
2.2.2 Return code vs. exception handling . . . 7
2.2.3 API with “out” or “ref” parameters . . . 7
2.3 Console Demo . . . 7
2.3.1 Windows . . . 7
2.3.2 Linux . . . 8
2.3.3 MacOS . . . 9

3 FAQ **11**

1 Introduction

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

1.1 Requirements

Swift v5.9 and above

MacOS (ARM64)

- macOS Ventura 13.5 and above

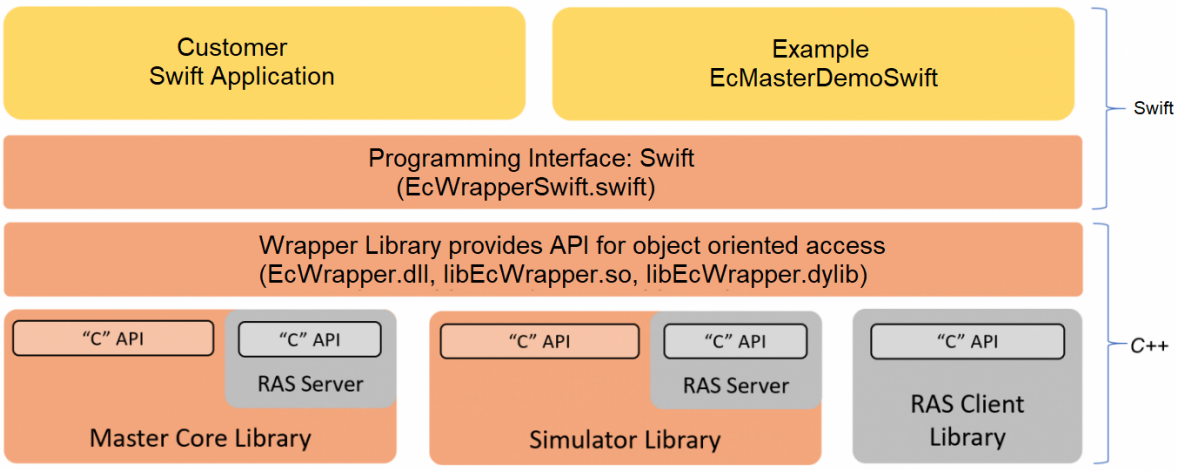
Windows (x86/x64)

- Microsoft Windows 10 and above
- Microsoft Visual C++ 2015 Runtime

Linux (x86/x64/ARM)

- Ubuntu 20.04 and above

1.2 Architecture



The architecture contains 4 basic layers:

Customer Swift Application or our examples (EcMasterDemoSwift, …)

- Demo application, written in Swift

Programming Interface (EcWrapperSwift)

- Provides an object oriented API written in Swift

Wrapper Library (EcWrapper)

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

Native Libraries

- Master Core Library
- Simulator Library
- RAS Client Library

2 Programmers Guide

2.1 Sample Code

The Swift demo application contains 1 module:

EcMasterDemoSwift.swift:
Console demo application

2.2 Wrapper

2.2.1 Modules

The Swift Wrapper contains 5 modules:

CEcWrapperSwift.swift

class CEcWrapperSwift
EC-Wrapper base class

class CEcMasterSwift
EC-Master

class CEcMasterMbxGatewayClientSwift
Mailbox Gateway Client for EC-Master

class CEcMasterMbxGatewayServerSwift
Mailbox Gateway Server for EC-Master

class CEcSimulatorSwift
EC-Simulator

class CEcSimulatorRasServerSwift
RAS Server for EC-Simulator

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

EcMotionSwift.swift

class CEcMotionSwift
EC-Motion interface

CEcWrapperSwiftTypes.swift
Swift types

CEcWrapper.swift
C Swift interface (internal)

CEcWrapperTypes.swift
C Swift types (internal)

2.2.2 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:

```
CEcWrapperSwift.EnableExceptionHandling = true // default is false
```

2.2.3 API with “out” or “ref” parameters

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

```
// This function has an "out" parameter "oSbStatus"
func GetScanBusStatus(oSbStatus: inout DN_EC_T_SB_STATUS_NTFY_DESC?) -> ECError {
    // ...
    return
}

// Create "out" parameter
var oStatus: DN_EC_T_SB_STATUS_NTFY_DESC?
// Call function
SwiftWrapper.GetScanBusStatus(oSbStatus: &oStatus)
// Now, the "oStatus" object can be used
print(oStatus!.dwResultCode)
```

2.3 Console Demo

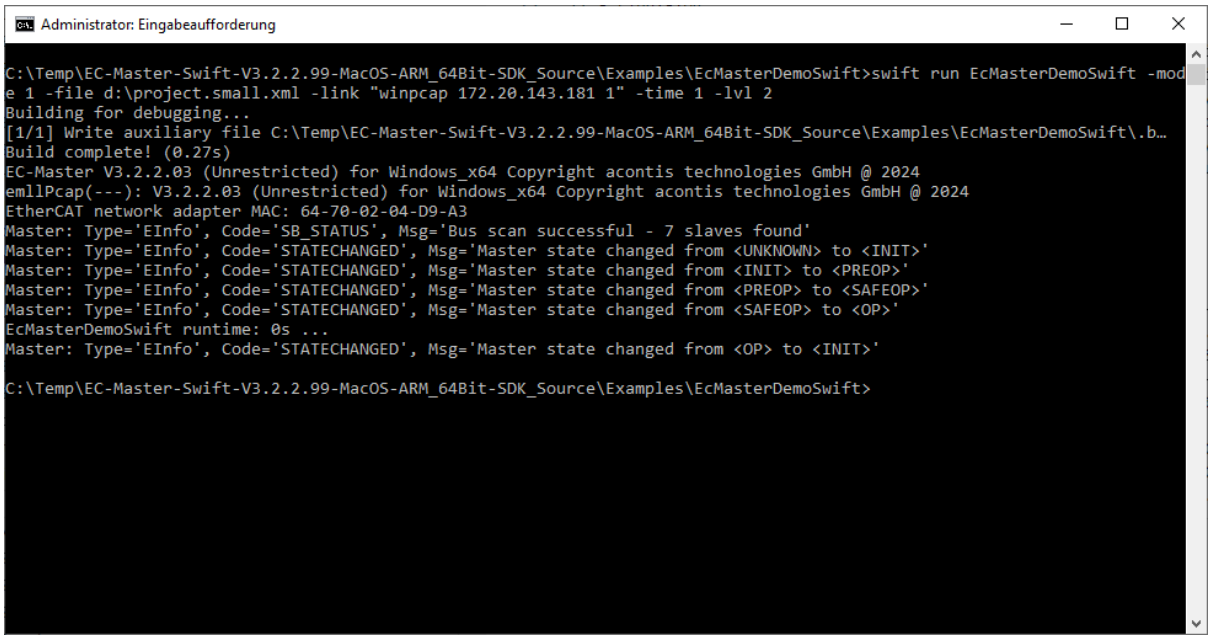
2.3.1 Windows

Open Windows Console Window, setup the environment and run the demo in the specific mode

```
cd C:\Temp\EC-Master-Swift-V3.2.2.01-02-MacOS-ARM_64Bit-SDK_Source\Examples\
↪EcMasterDemoSwift
set PATH=C:\Temp\V3.2.2.01-Windows-x86_64Bit\EC-Master-Windows-x86_64Bit\Bin\
↪Windows\x64;%PATH%
set ECWRAPPERSWIFT_INSTALLDIR=C:\Temp\V3.2.2.01-Windows-x86_64Bit\EC-Master-
↪Windows-x86_64Bit\Bin\Windows\x64\

// Master
swift run EcMasterDemoSwift -mode 1 -file d:\project.xml -link "winpcap 172.20.
↪143.181 1" -time 1
// Master+NDIS
swift run EcMasterDemoSwift -mode 1 -file d:\project.xml -link "ndis 172.20.
↪143.181 1" -time 1
// Master+RAS
swift run EcMasterDemoSwift -mode 1 -file d:\project.xml -link "winpcap 172.20.
↪143.181 1" -port 6000 -time 0
// Simulator
swift run EcMasterDemoSwift -mode 1 -file d:\project.xml -link "simulator d:\
↪project.xml 1 1" -time 1
// RAS-Client
swift run EcMasterDemoSwift -mode 2 -rem 127.0.0.1 -time 1
// Motion+Sim
swift run EcMasterDemoSwift -mode 1 -file d:\motion.xml -link "simulator d:\
↪motion.xml 1 1" -motion 1001 -port 6000 -time 0
```

… and the demo is running.



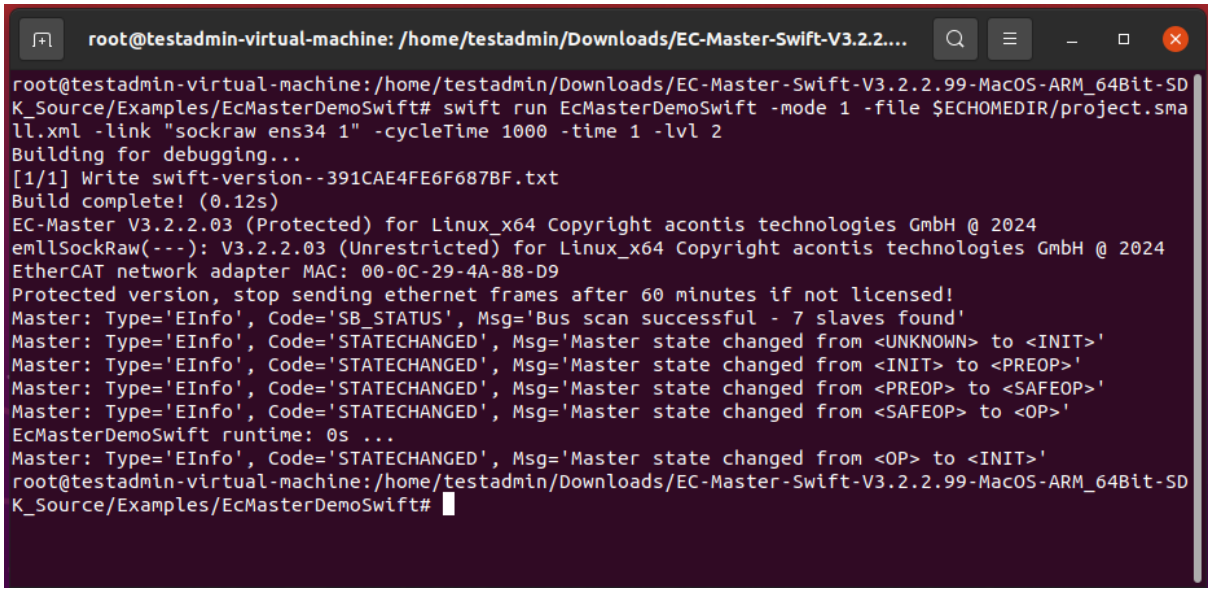
2.3.2 Linux

Open Terminal, setup the environment and run the demo in the specific mode

```
sudo -s // optional: required for sockraw link layer
cd /home/testadmin/Downloads/EC-Master-Swift-V3.2.2.01-02-MacOS-ARM_64Bit-SDK_
↪Source/Examples/EcMasterDemoSwift
export PATH=/home/testadmin/Downloads/swift-5.10.1-RELEASE-ubuntu20.04/usr/bin:
↪"${PATH}"
export PATH=/home/testadmin/Downloads/V3.2.2.01-Linux-x86_64Bit/EC-Master-V3.2.
↪2.01-Linux-x86_64Bit-Eval/Bin/Linux/x64:"${PATH}"
export LD_LIBRARY_PATH=/home/testadmin/Downloads/V3.2.2.01-Linux-x86_64Bit/EC-
↪Master-V3.2.2.01-Linux-x86_64Bit-Eval/Bin/Linux/x64:$LD_LIBRARY_PATH
export ECWRAPPERSWIFT_INSTALLDIR=/home/testadmin/Downloads/V3.2.2.01-Linux-x86_
↪64Bit/EC-Master-V3.2.2.01-Linux-x86_64Bit-Eval/Bin/Linux/x64/

// Master
swift run EcMasterDemoSwift -mode 1 -file /tmp/project.xml -link "sockraw␣
↪ens34 1" -time 1
// Simulator
swift run EcMasterDemoSwift -mode 1 -file /tmp/project.xml -link "simulator /
↪tmp/project.xml 1 1" -time 1
// RAS-Client
swift run EcMasterDemoSwift -mode 2 -rem 172.17.5.112 -time 1
```

… and the demo is running.



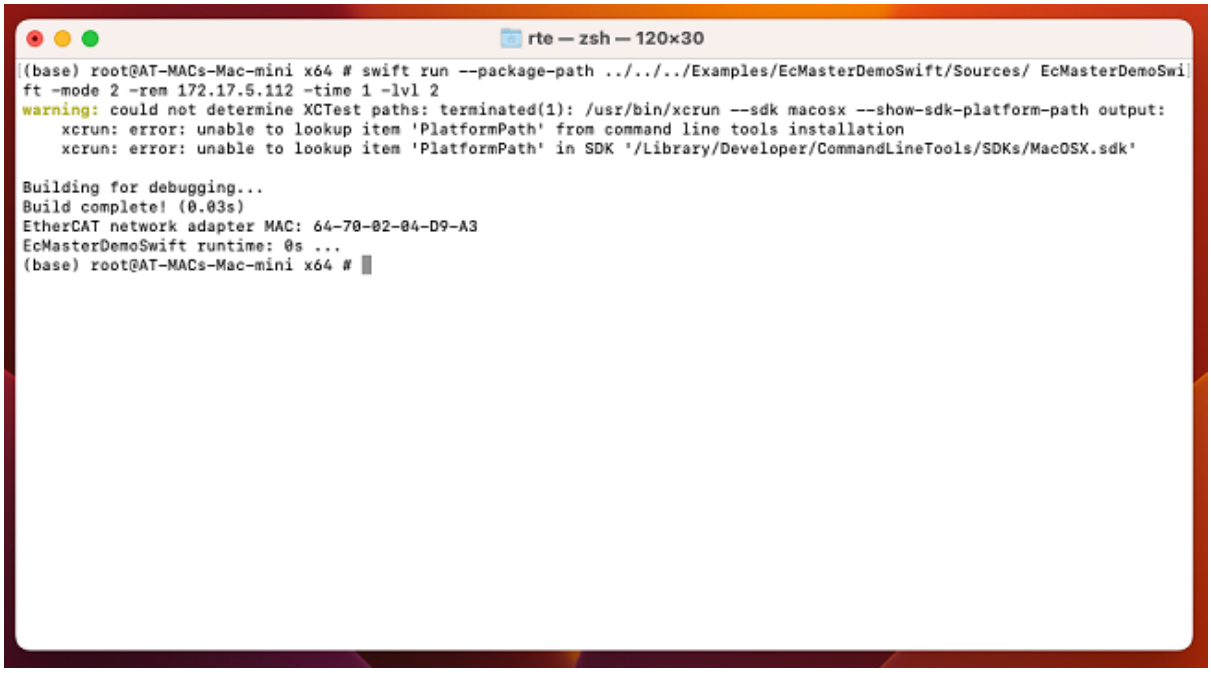
2.3.3 MacOS

Open Terminal, setup the environment and run the demo in the specific mode

```
sudo -s // optional: required for winpcap link layer, to avoid error message
 ↪"(cannot open BPF device) /dev/bpf0: Permission denied"
cd /Users/rte/Desktop/USERS/MGR/Test/EC-Master-Swift-V3.2.2.01-02-MacOS-ARM_
 ↪64Bit-SDK_Source/Bin/MacOS/x64 // Run from here with --package-path, because␣
 ↪otherwise link layers could not be found
export PATH=/Users/rte/Desktop/USERS/MGR/Test/EC-Master-Swift-V3.2.2.01-02-
 ↪MacOS-ARM_64Bit-SDK_Source/Bin/MacOS/x64:"${PATH}"
export DYLD_LIBRARY_PATH=/Users/rte/Desktop/USERS/MGR/Test/EC-Master-Swift-V3.
 ↪2.2.01-02-MacOS-ARM_64Bit-SDK_Source/Bin/MacOS/x64:$DYLD_LIBRARY_PATH
export ECWRAPPERSWIFT_INSTALLDIR=/Users/rte/Desktop/USERS/MGR/Test/EC-Master-
 ↪Swift-V3.2.2.01-02-MacOS-ARM_64Bit-SDK_Source/Bin/MacOS/x64/
export ECWRAPPERSWIFT_WRAPPERDIR=/Users/rte/Desktop/USERS/MGR/Test/EC-Master-
 ↪Swift-V3.2.2.01-02-MacOS-ARM_64Bit-SDK_Source/Bin/MacOS/x64/

// Master
swift run --package-path ../../../Examples/EcMasterDemoSwift/Sources/␣
 ↪EcMasterDemoSwift -mode 1 -file /Users/rte/Desktop/USERS/MGR/project.xml -
 ↪link "winpcap en7 1" -time 1
// Master+IP
swift run --package-path ../../../Examples/EcMasterDemoSwift/Sources/␣
 ↪EcMasterDemoSwift -mode 1 -file /Users/rte/Desktop/USERS/MGR/project.xml -
 ↪link "winpcap 192.168.0.1 1" -time 1
// Simulator
swift run --package-path ../../../Examples/EcMasterDemoSwift/Sources/␣
 ↪EcMasterDemoSwift -mode 1 -file /Users/rte/Desktop/USERS/MGR/project.xml -
 ↪link "simulator /Users/rte/Desktop/USERS/MGR/project.xml 1 1" -time 1
// RAS-Client
swift run --package-path ../../../Examples/EcMasterDemoSwift/Sources/␣
 ↪EcMasterDemoSwift -mode 2 -rem 172.17.5.112 -time 1
```

… and the demo is running.


rte — zsh — 120×30
(base) root@AT-MACs-Mac-mini x64 # swift run --package-path ../../../Examples/EcMasterDemoSwift/Sources/ EcMasterDemoSwi
ft -mode 2 -rem 172.17.5.112 -time 1 -lvl 2
warning: could not determine XCTest paths: terminated(1): /usr/bin/xcrun --sdk macosx --show-sdk-platform-path output:
xcrun: error: unable to lookup item 'PlatformPath' from command line tools installation
xcrun: error: unable to lookup item 'PlatformPath' in SDK '/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk'
Building for debugging...
Build complete! (0.03s)
EtherCAT network adapter MAC: 64-70-02-04-D9-A3
EcMasterDemoSwift runtime: 0s ...
(base) root@AT-MACs-Mac-mini x64 #

3 FAQ

I installed Swift 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 Swift runtime for x64 bit, please install also EC-Master for x64 bit.