IBM Open Enterprise SDK for Go 1.17

User's Guide

SC28274800 (2022-03-11)

This edition applies to version 1.17 of IBM® Open Enterprise SDK for Go (order number: SC28-2748-00) and to all subsequent releases and modifications until otherwise indicated in new editions.

It is our intention to update the product documentation for this release periodically, without updating the order number. If you need to uniquely refer to the version of your product documentation, refer to the order number with the date of update.

Last updated: 2022-03-11

Contents

Chapter 1. Blogs and videos

Understanding the business value

- Blog: IBM Open Enterprise SDK for Go 1.17 is now available!
- Overview video: IBM Open Enterprise SDK for Go

How-to tutorials and blogs

- Blog: Obtain the SMP/E edition and optional support of IBM Open Enterprise SDK for Go
- Blog: Calling a COBOL function from Go
- Tutorial video: Connecting compiled binary programs to Go programs
- Tutorial video: Direct access to VSAM databases with Go recordio module

Chapter 2. Overview of IBM Open Enterprise SDK for Go

IBM Open Enterprise SDK for Go is an industry-standard Go compiler optimized for the z/OS® platform. The Go compiler leverages the latest z/Architecture® instructions to provide an exceptional implementation on the z/OS platform.

Go is a general-purpose programming language for building large-scale complex software, and is the language of cloud infrastructure such as Kubernetes, Open Container Initiative, and OpenShift. It is designed to be a fast running programming language, easy to learn and understand due to its simple syntax. Go contains a powerful standard library of packages and built-in functions to support application development. It was rated as the 3rd most wanted language in Stack Overflow Developer Survey and 4th in GitHub pull requests recently, making it the second fastest in popularity growth.

IBM Open Enterprise SDK for Go includes the following features:

- A native Go compiler to enable new and existing applications written in Go to run on z/OS
- Enablement of Go 1.17 to z/OS UNIX Systems Services
- Support for the Go standard runtime library
- Support for Go modules and multithreading
- Support for the standard UTF-8 Unicode
- Support for cgo, which enables Go to call C code
- Support for direct access to VSAM databases
- Access to additional packages that enable more ready-to-use features
- Program licenses available at no charge
- Optional IBM Subscription and Support to provide access to IBM world-class support

Chapter 3. Release notes

This release note provides information about IBM Open Enterprise SDK for Go 1.17 in IBM Z and Cloud Modernization Stack. For release notes about other products in IBM Z and Cloud Modernization Stack, see Release notes.

If IBM Open Enterprise SDK for Go is modified between releases of IBM Z and Cloud Modernization Stack, additional versions might be available. For the latest information about product versions, always check the IBM Z and Cloud Modernization Github community.

What's new

IBM Open Enterprise SDK for Go 1.17 brings several significant features from the open-source community:

- Lazy module loading is supported to avoid processing irrelevant dependencies.
- New functions for safer pointer arithmetic are introduced.

Numerous additional improvements and bug fixes are also brought up from the community. For details, see the Go 1.17 community announcement blog.

Support for access to VSAM data sets

Direct access to the VSAM data set type without cgo is introduced. The usage and demonstrations are available at Chapter 8, “Accessing Virtual Storage Access Method (VSAM) databases directly from Go,” on page 11.

Known issues and limitations

None in this release.

Chapter 4. Installing and configuring IBM Open Enterprise SDK for Go

IBM Open Enterprise SDK for Go is available in two installation formats, SMP/E and PAX. Select the installation format that applies to you:

- "Installing and configuring the SMP/E edition" on page 4
- "Installing and configuring the PAX edition" on page 5

Installing and configuring the SMP/E edition

Planning for installation requires three main tasks: ensuring you have the correct hardware and software installed, choosing the right default location, and setting up the environment variables accordingly. For specific installation requirements and instructions of the product, see chapter 5 and 6 of the Program Directory.

Prerequisites

Hardware requirements

- z15™
- z14®/z14 model ZR1
- z13®/z13s®
- zEnterprise® EC12/BC12

Software requirements

- z/OS UNIX System Services enabled on any of following operating systems:
  - z/OS 2.4 or later with the following APARs:
    - OA59780
    - PH30936

    Note: Go programs need to bind to the LE runtime services using the c89 utility under the z/OS UNIX System Services environment. You may encounter the following error if your c89 utility was not set up correctly:

    ```
    FSUM3052 The data definition name SYSLIB cannot be resolved. The data set was
    not found. Ensure that data set name CEE.SCEEBND2 is specified correctly.
    open /tmp/go-build794287247/b001/exe/hello: EDC5129I No such file or directory.
    ```

    For details on how to fix this error, see "Cannot locate CEE.SCEEBND2 data set" on page 14.
- For using cgo (CGO_ENABLED=1):
  - XL C/C++ 2.4.1 for z/OS 2.4 with the following APAR:
    - PH27303
  - For how to compile applications written in C and C++, see Getting Started with XL C/C++ 2.4.1 for z/OS 2.4.
- Rocket bash 4.3 or later is required. Must also be available on your PATH.

Note: To download Rocket bash, go to Bash for z/OS on the Rocket Software website.

Procedure

Configuration

IBM Open Enterprise SDK for Go is an OMVS-based application, which requires certain configuration on the z/OS UNIX System Services file system to ensure proper operation.

- Validate that `/usr/bin/env` exists.
- Ensure that `/tmp` has at least 860 MB or more of disk space configured. To use an alternative file system, you can set the *TMPDIR* environment variable to a directory that has sufficient space.

Default installation location for IBM Open Enterprise SDK for Go

The default Go SMP/E installation location on z/OS is `/usr/lpp/IBM/cvg/v1r17/IBM/`.

Environment variables for SMP/E installation

Set the following environment variables before using IBM Open Enterprise SDK for Go.

- If you use cgo, configure the `COMPILER_PATH` environment variable and source the envsetup script in your .bash profile:

```
export COMPILER_PATH=<path to xlclang command>
source <go installation path>/go/etc/envsetup
```

- If you do not use cgo, source the envsetup script in your .bash profile:

```
source <go installation path>/go/etc/envsetup
```

- Reload your bash profile:

```
source ~/.bashrc
```

Installing and configuring the PAX edition

Planning for installation requires three main tasks: ensuring you have the correct hardware and software installed, choosing the right default location, and setting up the environment variables accordingly. To download the latest PAX edition, visit IBM Open Enterprise SDK for Go product page.

Prerequisites

Hardware requirements

- z15™
- z14®/z14 model ZR1
- z13/z13s
- zEnterprise EC12/BC12

Software requirements

- z/OS UNIX System Services enabled on any of following operating system:
  - z/OS 2.4 or later with the following APARs:
    - OA59780
    - PH30936

    Note: Go programs need to bind to the LE runtime services using the c89 utility under the z/OS UNIX System Services environment. You may encounter the following error if your c89 utility was not set up correctly:

    ```
    FSUM3052 The data definition name SYSLIB cannot be resolved. The data set was
    not found. Ensure that data set name CEE.SCEEBND2 is specified correctly.
    open /tmp/go-build794287247/b001/exe/hello: EDC5129I No such file or directory.
    ```

    For details on how to fix this error, see “Cannot locate CEE.SCEEBND2 data set” on page 14.

- For using cgo (CGO_ENABLED=1):
  - XL C/C++ 2.4.1 for z/OS 2.4 with the following APAR:
    - PH27303
  - For how to compile applications written in C and C++, see Getting Started with XL C/C++ V2.4.1 for z/OS 2.4.
- Rocket bash 4.3 or later is required. Must also be available on your PATH.

Note: To download Rocket bash, go to Bash for z/OS on the Rocket Software website.

Procedure

Install the PAX archive file

- 250 MB is required to download the PAX archive file.
- Minimum 850 MB is required to extract and install Go.
- Create a directory <mydir> to hold the extracted PAX files.
- Unpax the downloaded file with the following command:

```
$ cd <mydir>
$ pax -p p -r -f <path to downloaded paxfile>
```

Environment variables for PAX archive installation

Set the following environment variables before using IBM Open Enterprise SDK for Go.

- If you use cgo, configure the COMPILER_PATH environment variable and source the envsetup script in your .bash profile:

```
export COMPILER_PATH=<path to xlclang command>
source <go installation path>/go/etc/envsetup
```

- If you do not use cgo, source the envsetup script in your .bash profile:

```
source <go installation path>/go/etc/envsetup
```

- Reload your bash profile:

```
source ~/.bashrc
```

Chapter 5. Getting started with IBM Open Enterprise SDK for Go

To get started with IBM Open Enterprise SDK for Go, you can work with the Go edition of a "Hello World" program as an example.

Prerequisites

Before getting started, you must meet the following prerequisites:

- Ensure that the required environment variables are set up properly. See Environment variables for SMP/E installation or Environment variables for PAX archive installation.
- Ensure that the Go version is up to date. Check your Go version with the following line:

  ```
  go version
  ```

  When you're requested by service, use this command to display the current Go version built date and commit from official builds:

  ```
  /bin/strings $(type -p go) | /bin/awk '/License/{ for(i=0;i<7;i++) {print;getline}}'
  ```

  Note: `go version <executable-name>` does not work where <executable-name> is any executable built with the Go compiler on z/OS.

Procedure

You can set up a simple file with the `Hello World` message through the following steps:

1. Create a file named "hello.go" with the following command:

   ```
   package main

   import "fmt"

   func main() {
       fmt.Println("Hello World")
   }
   ```

2. Execute the following line:

   ```
   go build hello.go

   ./hello
   ```

3. The message is printed as follows:

   ```
   go build hello.go

   ./hello
   ```

Through the previous steps, you have completed the getting started task of compiling a sample Go program.

Chapter 6. Using z/OS specific extended modules

For Open Enterprise SDK for Go, porting applications to z/OS requires you to use the extended modules. This topic introduces how to use the extended modules to port applications.

The following extended modules are ported to z/OS and upstreamed:

- `golang.org/x/crypto`
- `golang.org/x/net`
- `golang.org/x/sys`
- `golang.org/x/term`

To use the extended modules, you need to check for the `go.mod` file in the root directory of the package you are trying to build:

- Submodules of `/home/robin/earlybird`
- Vendor subdirectory

Note: If there is no `go.mod` file, you must create one with the following 2 cases to consider:

- If using GOPATH, run the following command:

```
go mod init
```

- If not using GOPATH, run the following command:

```
go mod init modname
```

If the `go.mod` file references any of the extended modules, update the version number by replacing the version number to the word "latest", and the module will be automatically updated to the latest version when you run the following command:

```
go mod init
```

If there is a vendor subdirectory, you must execute the following command each time you update the `go mod` file:

```
go mod vendor
```

This will cause Go to replicate dependencies in the vendor directory and include updated versions of the extended modules.

Chapter 7. Porting applications to z/OS

This topic contains information about porting Go applications to z/OS. To achieve execution of Go modules on z/OS, porting has to occur especially for modules of significant size. Porting Go is free of the endianness issues found in C/C++ and other statically compiled languages, and you need to change or copy certain files for certain modules.

For example, build tags and file naming are two techniques used in Go that determine what platforms to build on. A file with a comment "// +build zos" at the top will build on z/OS, a file with a tag of "!zos" will not build on z/OS, and a file with a name "myfile_zos_s390x.go" will build only on z/OS. You need to determine whether to add tags or change names to certain files in the module. If not, you might run into "undefined" symbol errors during the build.

If the module has many dependent modules, you also need to port some of them by modifying their `go.mod` files.

Basic steps to port a dependent package

To port a dependent package to z/OS, take the following steps:

1. Run `export GO111MODULE=auto` to use `GOPATH`.
2. In order to use `GOPATH`, first `export GO111MODULE=auto`, then set the environment variable `GOPATH` to a directory of your choice and ensure that directory has subdirectories: `src`, `pkg`, and `bin`.
3. Start with a clean environment with the following command:

   ```
   go clean; go clean -cache -modcache
   ```

4. Download package source using `go get` or `git clone`.
5. Check for `go.mod` file, and update the version number to the word "latest" if the file references any of the following extended modules:

   ```
   golang.org/x/crypto,golang.org/x/net,golang.org/x/sys,golang.org/x/term
   ```

   Note: For vendor directory, each time you update the `go.mod` file, you must execute the following command:

   ```
   go mod vendor
   ```

6. Install the Go package with the following line:

   ```
   go install
   ```

   The version number for the extended modules will be updated automatically to the latest.
7. Check for build errors.
8. Port this package and any additional dependent packages.
9. Keep iterating and porting until the build is clean.

Tips:

- Look for files with names like ..._linux.go. They are likely to be copied or modified for z/OS.
- Look for files with build tags like '+build linux' or '+build !linux', which might need to add zos to the build tag: "+build linux zos" or "+build !linux,!zos".
- Look for `go.mod` file and update the version number to the word "latest" for crypto, net, sys, and term packages.

  Note: Perform `go install` after the updates are done so that the version number will be updated automatically to the latest.

- For sycalls, modules often contain calls to the raw `Syscall` functions (`Syscall`, `RawSyscall`, `Syscall6`, `RawSyscall6`) on Linux®, which should be replaced with calls to the associated published APIs. For example, a call like:

  ```
  syscall.Syscall(SYS_FSTAT, uintptr(fd), uintptr(unsafe.Pointer(stat)), 0)
  ```

  will give an "undefined syscall.Syscall" build error on z/OS, but the equivalent call to the API `syscall.Fstat(...)` will work fine.

  gRPC porting example

  Port gRPC to z/OS by taking the following steps:

  1. Run `export GO111MODULE=auto` to use `GOPATH`.
  2. Set the environment variable `GOPATH` to a directory of your choice and ensure that directory has subdirectories: `src`, `pkg`, and `bin`.
  3. Start with a clean environment with the following command:

     ```
     go clean go clean -cache -modcache
     ```

  4. Download the gRPC package with the following command:

     ```
     go get google.golang.org/grpc
     cd $GOPATH/src/google.golang.org/grpc
     ```

  5. In `go.mod` file, change the version number to the word "latest" for `golang.org/x/crypto`, `golang.org/x/net`, `golang.org/x/sys`, and `golang.org/x/term` packages.
  6. Execute the following command:

     ```
     go install
     ```

Chapter 8. Accessing Virtual Storage Access Method (VSAM) databases directly from Go

zosrecordio is a Go module that provides a way to access and manipulate VSAM databases directly from Go on z/OS. Operations are implemented through the z/OS Language Environment interfaces without calling C from Go. The following operations are included:

- Opening, closing, and reopening VSAM databases
- Record, reading, and writing records
- Locating records
- Updating records
- Deleting records

Inputs and outputs from these operations are uniform byte slices, which eliminates the need for error-prone size specifications. Various utility functions are also included for converting back and forth between byte slices and structs, so that record types might be programmed as Go structs.

For details on how to access VSAM database, see recordio in Go: A Go module for record i/o in VSAM databases directly from Go.

Chapter 9. Integrating Go with middleware and other programs on z/OS

IBM Open Enterprise SDK for Go provides connectivity to IMS data sets through z/OS Connect EE.

Use REST APIs created by z/OS Connect EE to access IMS data sets. For demonstrations on how sample Go applications on z/OS use the REST APIs created by z/OS Connect EE to access z/OS applications and data in CICS®, IMS, and Db2®, see Sample Go applications on z/OS.

Chapter 10. Troubleshooting

This topic describes some common issues that you might encounter with the IBM Open Enterprise SDK for Go:

Validation failed for install_test.sh without bash

The validation task is unable to execute `install_test.sh` without bash. This is because Rocket bash is not available on your PATH. Rocket bash 4.3 or later is required for installing and configuring Open Enterprise SDK for Go. To download bash, go to Bash for z/OS on the Rocket Software website.

run clang failed

- Ensure that you sourced the `.envsetup` script in `go/etc` directory.
- Use the required version of bash.

Certificate signed by unknown authority

When you run a Go application on z/OS that uses packages outside of the standard Go library, you might encounter this error: Certificate signed by unknown authority. This could be a result of git failure under openSSL framework. You can try to fix the problem using one the following methods:

- Set SSL_CERT_FILE environment variable correctly, and then export SSL_CERT_FILE variable.
- Source `.envsetup` in `go/etc` which sets the SSL_CERT_FILE to go/etc/cacert.pem.

After you download the packages, you need to add build tags to port the applications to z/OS. See Chapter 7, "Porting applications to z/OS," on page 9 for more details.

Messages about finding or downloading x/sys/unix, x/crypto, or x/net

When you see this message, try the following solutions:

- Edit the `go.mod` file by changing the version number of `golang.org/x/sys/unix, x/crypto, or x/net` packages to the word "latest".
- Run the following command:

  ```
  go install
  ```
- Change minimum Go version to Go 1.17.

Undefined errors

You might encounter undefined errors in the following cases:

1. The package has a Unix version of the file but no z/OS build tag.

   To tackle this problem, add z/OS to + build line at top of the file.
2. The package only has a Linux version of the file but all functionality used in the file exists on z/OS.

To tackle this problem, add z/OS to + build line at top of the file.

3. The function uses `Syscall(…) or RawSyscall(...)`.

   This is not supported on z/OS. You need to make z/OS version of this function.

Cannot locate CEE.SCEEBND2 data set

If you build Go programs under the default z/OS USS system, you're supposed to locate CEE.SCEEBND2 data set. If you encounter the following error message, there might be something missing in you configuration.

```
FSUM3052 The data definition name SYSLIB cannot be resolved. The data set was
not found. Ensure that data set name CEE.SCEEBND2 is specified correctly.
open /tmp/go-build794287247/b001/exe/hello: EDC5129I No such file or directory.
```

Try to fix this issue by simulating binding a 64-bit program using the c89 utility with the following command:

```
export V=1
touch hello.o
c89 -v -Wl,LP64 hello.o
```

You should see pseudo JCL output as follows:

```
export _C89_PLIB_PREFIX=SYS1.CEE
export _C89_CLIB_PREFIX=SYS1.CBC
```

For more details, see Unix System Services Command Reference.

Cannot produce dumps

Try the following procedure to cause a dump to occur:

1. Set LE runtime environment with the command:

   ```
   export _CEE_RUNOPTS=FILETAG(AUTOCVT,AUTOTAG) TERMTHDACT(UADUMP) ABTERMENC(ABEND)
   ```

2. Disable the Go signal handler for the signal in question by setting the signal handler to the default. This can be done in either your C code or Go code.
3. Export your dump with the following line:

   ```
   export _BPXK_MDUMP=your_dump_data_set_name
   ```

Chapter 11. Known issues and limitations

IBM Open Enterprise SDK for Go has the following known issues or limitations:

- Plugins are not supported on z/OS.
- The following cgo functionalities are not supported on z/OS:
  - Accessing C file scope variables
  - Callback from a C program to Go
  - Signal handling from C
- The following commands do not work for z/OS:
  - `addr2line`: A minimal simulation of the GNU addr2line tool, just enough to support pprof.
  - `cover`: A program for analyzing the coverage profiles generated by 'go test -coverprofile=cover.out'.
  - `objdump`: Disassembles executable files.
  - `oldlink`: Link, typically invoked as “go tool link,” reads the Go archive or object for a package main, along with its dependencies, and combines them into an executable binary.
  - `trace`: A tool for viewing trace files.

  For full list of Go commands, see Go commands list.
- Others:

  race detector is not available on z/OS. You can run into error when using -race flag during the build.

Chapter 12. FAQs

Find answers for commonly asked questions for Go on z/OS.

What are the supported target z/OS levels and architectures for the compiled binaries to run?

Go on z/OS 1.16 is supported on z/OS 2.4 and later both for compilation and running, so a program compiled on z/OS 2.4 should run fine on 2.5. Go on z/OS supports z196 and later machines.

Are compilation APARs and C compiler necessary for Go on z/OS at run time?

Go programs are statically linked, so they carry all their dependencies with them, which means that even a Go program with cgo can run on a system with no C compiler installed. Once compiled, programs shouldn't need the C compiler or binder APARs.

Can I run Go binaries on z/OS 2.3?

Go programs can be run on z/OS 2.3 at risk because it is not a supported configuration. For software and hardware prerequisites, see Chapter 4, “Installing and configuring IBM Open Enterprise SDK for Go,” on page 4.

Is debugger support available for Go on z/OS?

IBM® z/OS® Debugger supports debugging programs compiled with IBM Open Enterprise SDK for Go. For details, see **Debugging programs compiled with IBM Open Enterprise SDK for Go.**

Is Go binary on z/OS UNIX static or dynamic?

Go binary on z/OS UNIX is always static.

Chapter 13. Getting help

To find help about IBM Open Enterprise SDK for Go, it is important to collect as much information as possible about your installation configuration.

To establish what version of IBM Open Enterprise SDK for Go is in use, run the following command:

```
go version
```

The version of Go is displayed.

Online self-help

Online documentation is available on IBM Documentation. You can also download the PDF format documentation for offline use.

Getting IBM experts to solve your problem by opening a case

Paid IBM Subscription & Support (S&S) entitles you to world-class IBM support for IBM Open Enterprise SDK for Go. Get IBM support by opening a case. First, obtain the SMP/E edition and purchase S&S. Once you have purchased S&S, visit the Go Support Portal to request support from IBM after logging in with your IBM customer ID.

General Help

For help with writing Go programs, working with the standard library or other general inquiries, see Go get help page.

Learning resources

This topic lists both Go community resources and IBM resources that you can refer to.

Go community resources

- IBM Open Enterprise SDK for Go product page
- Go on z/OS community
- Request for Enhancement (RFE) community
- Go Support Portal

IBM resources

- IBM Community
- Redbooks® Introduction to z/OS
- IBM Online Software Catalog
- Explore IBM Systems

Product Number: 5655-GOZ

SC28274800 (2022-03-11)