SOS info utility for z/OS XL C/C++ compiler

# SOS info utility for z/OS XL C/C++ compiler

## Introduction

Starting with z/OS® V1R10, the z/OS XL C/C++ compiler saves some compiler options information inside the object code for each source file that is being compiled. These saved compiler options are known as Saved Options String (SOS). The SOS info utility decodes the SOS information from an executable file and produces a list of the compiler options that were used to control the code generation of a program. Using this utility, you can obtain the options information from an executable file without the need to produce and maintain compiler listing files.

## Detailed description

The SOS info utility decodes SOS information from a given executable file and produces a list of compiler options in a human readable form. Besides compiler options, the SOS info utility also shows the following information:

- Selected information from the Program Prolog Area 2 (PPA2), such as timestamp and service string.
- The source file name that was used to produce the compilation unit.
- The version string for each compiler component that was active during the compilation.

An executable file can be produced by linking various object files, some of which can be compiled with other compilers and some can be compiled with different releases of the z/OS XL C/C++ compiler. The SOS information is preserved during the linking of the program, so the SOS information is available for each compilation unit that is linked into the program.

To recognize compilation units that contain the SOS structure, the z/OS XL C/C++ compiler turns on the SOS bit in the PPA2 control block. Because the SOS structure can differ from release to release, the z/OS XL C/C++ compiler also emits a version number in the SOS structure. This version number changes whenever a new compiler option is introduced in a specific release.

Using the SOS bit in the PPA2 control block and the SOS version number, the SOS info utility produces options information for all compilation units that contain the SOS information.

- When the release of the compiler that is used to compile the code is the same as or lower than the release of the SOS info utility, all the SOS information is shown in the output.
- When an older release of the SOS info utility is used with an executable file that contains object code that is produced by a newer release of the compiler, the output does not show options information for options that are introduced in the newer release of the compiler.

**Notes:**

- The SOS information is produced for each compilation unit that is compiled with z/OS XL C/C++ V1R10 or later.
- The SOS info utility is initially shipped with the z/OS V2R3 XL C/C++ compiler.
- The SOS info utility is also provided in PTFs for earlier releases of the z/OS XL C/C++ compiler starting with z/OS V2R1.

## Usage

Options information that is produced by the SOS info utility can be used to diagnose problems and analyze the usage of the compiler features. Only options that affect code generation are included in the SOS information.

The SOS info utility can be used in both z/OS UNIX System Services (z/OS UNIX) and JCL.

**In z/OS UNIX**
You can invoke the SOS info utility by the **`/bin/sosinfo`** command, which is an external link to the CDASOS executable file that is shipped in the `CEE.SCEERUN2` data set. The utility produces the output in a standard output stream that can be redirected to a z/OS UNIX file.

**In JCL**
You can invoke the SOS info utility by the CDASOS JCL procedure, which is shipped in the `CEE.SCEEPROC` data set. The output is provided in the data set that is allocated to SYSPRINT DD. You can also invoke the utility by using the `EXEC PGM=CDASOS` JCL statement.

## Input

The SOS info utility requires a name of an executable file as input and produces options information for this executable file. You can specify the name of the executable file in one of the following formats:

**A fully qualified data set member name**
Case insensitive. An example would be `"//'cbc.sccncmp(ccnep)'"`.

**A full path name of a z/OS UNIX file**
An example would be **`./a.out`**.

**A module name**
Case insensitive. An example would be `ccnep`. The module must be in LPA; otherwise, the module must be contained in one of the data sets that are specified by the STEPLIB environment variable.

**A z/OS UNIX external link (z/OS UNIX only)**
An example would be `driver`, which is created by using the **`ln -e CCNDRVR driver`** z/OS UNIX command. The module that is pointed to by the external link must be in LPA; otherwise, the module must be contained in one of the data sets that are specified by the STEPLIB environment variable.

**A DD name (JCL only)**
Allocated to a data set member name of an executable file or a path name of a z/OS UNIX executable file.

## Comparison to listing files

An alternative to using the SOS info utility is to obtain the options information from the listing files that are produced during the compilation.

The SOS info utility shows only resolved options that affect code generation. While listing files show user specified options, regardless of whether they affect code generation.

Comparing to the SOS info utility, listing files have the following disadvantages:

- Might not be available for object codes that are owned by third-party vendors.
- Separate from the executable file and can get out of sync.
- Might not always be produced.
- Might require sizable storage.

## Examples

### Example 1

In z/OS UNIX, you can run the following command to invoke the SOS info utility:

```
/bin/sosinfo executable_name
```

Where `executable_name` can be one of the following types in Table 1.

*Table 1. Valid executable_name types and examples*

| **Type of** *executable_name* | **Example** |
|---|---|
| A fully qualified data set member name | **/bin/sosinfo "//'cee.sceerun2(cdadbgld)'"** |
| A full path name of a z/OS UNIX executable file | **/bin/sosinfo ./a.out** |
| A module name | **/bin/sosinfo CCNEP** |
| An external link to an executable file | **ln -e CCNDRVR driver**<br>**/bin/sosinfo driver** |

### Example 2

In JCL, you can invoke the SOS info utility by using the CDASOS JCL procedure. For example:

```
//ORDER JCLLIB ORDER=(CEE.SCEEPROC)
//*-----------------------------------------------------------
//STEP1 EXEC CDASOS,INFILE='DD:MYEXEC'
//MYEXEC DD PATHOPTS=(ORDONLY,ONONBLOCK),
// PATH='/tmp/sosinfo/a.out'
//SYSPRINT DD PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRWXU),FILEDATA=TEXT,
// PATH='/tmp/sosinfo/sos.out'
```

### Example 3

In JCL, you can also invoke the SOS info utility as a regular program. For example:

```
//STEP1 EXEC PGM=CDASOS,PARM='DD:EXECDD'
//STEPLIB DD DISP=SHR,DSN=CEE.SCEERUN2
// DD DISP=SHR,DSN=CEE.SCEERUN
//EXECDD DD DISP=SHR,DSN=HLQ.LOAD(MYPGM)
//SYSPRINT DD SYSOUT=*
```

The following example shows a partial output from the SOS info utility:

```
Compile Unit #1

ppa2_time_stamp = 2017 03:09 15:06:27 1.13

ppa2_flt_ieee = NOIEEE

ppa2_service = SERVICE(B20121019.B2.zosv1r13)

ppa2_xpl_stargs = NOSTOREARGS

ppa2_charset = NOASCII

ppa2_sos = SOS

ppa2_xpl_compile = XPLINK

ppa2_md5_signature = NOMD5

ppa2_flt_afp_vol = NOVOLATILE

sos_words = 8

sos_version = 3

sos_arch = ARCH(7)

sos_tune = TUNE(8)

* * *

sos_rtc_bounds = NORTCHECK(NOBOUNDS)

sos_rtc_divzero = NORTCHECK(NODIVZERO)

sos_rtc_nullptr = NORTCHECK(NONULLPTR)

sos_restrict_param = NORESTRICT

sos_tempsaslocals = LANGLVL(TEMPSASLOCALS)

source file name = cbcphsid.c

CCNDRVR = B20121019.B2.zosv1r13

CCNEOPTP = B20121019.B2.zosv1r13

CCNEP = B20121019.B2.zosv1r13

CCNETBY = B20121019.B2.zosv1r13
```

As you can see from this example, the SOS info utility can work with programs that are compiled with older versions of the z/OS XL C/C++ compiler.

## Purchasing

Information about purchasing z/OS XL C/C++ is available at the Marketplace website:

https://www.ibm.com/us-en/marketplace/xl-cpp-compiler-zos

## Contacting IBM

IBM welcomes your comments. You can send them to compinfo@cn.ibm.com.

**August 2017**

References in this document to IBM® products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.