

Rocket COBOL 11.0 — Technische Manual voor DLL-Generatie

Rocket COBOL 11.0 - Technische Manual voor DLL-Generatie

1. Inleiding

Deze manual beschrijft hoe met Rocket COBOL 11.0 native Windows-DLL's worden gebouwd die door andere talen (C, Fortran, ML64-assembly, Python, C#, Rust) kunnen worden aangeroepen. De nadruk ligt op correcte C-ABI-interoperabiliteit, stabiele symbol-exports en foutloze build-procedures.

2. Overzicht van de Toolchain

Rocket COBOL 11.0 ondersteunt DLL-generatie via:

- **cobol.exe** - compiler
- **cbllink.exe** - linker
- **.def-files** - expliciete exportcontrole
- **CALL-CONVENTION 0** - Windows x64 ABI

Deze combinatie maakt het mogelijk om COBOL-routines als C-compatibele functies te exporteren.

3. Structuur van een DLL-Project

Een typische projectstructuur:

```
/project
add.cbl
subtract.cbl
exports.def
build.bat
include/
cobol_exports.h
bin/
mymath.dll
mymath.lib
```

4. COBOL-Code voor DLL-Exports

4.1 Basispatroon voor een exporteerbare routine

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. ADDNUMS AS "addnums".  
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SPECIAL-NAMES.  
CALL-CONVENTION 0.  
DATA DIVISION.  
LINKAGE SECTION.  
01 A BINARY-LONG.  
01 B BINARY-LONG.  
01 RESULT BINARY-LONG.  
PROCEDURE DIVISION USING BY VALUE A B RETURNING RESULT.  
  COMPUTE RESULT = A + B  
  GOBACK.
```

4.2 Belangrijke regels

- PROGRAM-ID ... AS "symbol" bepaalt de exportnaam.
- CALL-CONVENTION 0 garandeert Windows x64-ABI.
- USING BY VALUE is verplicht voor C-compatibiliteit.
- RETURNING levert een echte C-returnwaarde.

5. Exportlijst (.def-file)

Gebruik een .def-bestand om exact te bepalen welke symbolen worden geëxporteerd:

```
LIBRARY "mymath"  
EXPORTS  
  addnums  
  subtractnums
```

Dit voorkomt dat interne COBOL-runtime-symbolen zichtbaar worden.

6. Compileren van COBOL-Modules

Compileer elke .cbl naar een objectbestand:

```
cobol add.cbl /C /O add.obj  
cobol subtract.cbl /C /O subtract.obj
```

Flags:

- /C - compile only

- /O - output object file
-

7. Linken naar een DLL

Gebruik **cbllink**:

```
cbllink add.obj subtract.obj ^  
/DLL ^  
/OUT:mymath.dll ^  
/DEF:exports.def ^  
/LIBPATH:"%COBOL_LIB%" ^  
/NOENTRY
```

Resultaat:

- mymath.dll
 - mymath.lib (import library voor C/Fortran)
-

8. C-Header voor Interoperabiliteit

```
#ifndef MYMATH_H  
#define MYMATH_H  
  
#ifdef __cplusplus  
extern "C" {  
#endif  
  
__declspec(dllimport) int addnums(int a, int b);  
__declspec(dllimport) int subtractnums(int a, int b);  
  
#ifdef __cplusplus  
}  
#endif  
  
#endif
```

9. Aanroepen vanuit C

```
#include "mymath.h"  
#include <stdio.h>  
  
int main() {  
    printf("%d\n", addnums(10, 20));  
}
```

Compileer:

```
cl main.c mymath.lib
```

10. Aanroepen vanuit Fortran (Intel / IFX)

```
interface
function addnums(a, b) bind(C, name="addnums")
use iso_c_binding
integer(c_int), value :: a, b
integer(c_int) :: addnums
end function
end interface

print *, addnums(10, 20)
```

11. Aanroepen vanuit ML64-Assembly

```
extern addnums:proc

mov ecx, 10
mov edx, 20
call addnums
```

12. Best Practices

12.1 Expliciete symbolnamen

Gebruik altijd:

```
PROGRAM-ID. FOO AS "foo".
```

12.2 BY VALUE verplicht

COBOL's default BY REFERENCE breekt C-ABI.

12.3 Gebruik BINARY-LONG / BINARY-DOUBLE

Deze mappen exact op int32_t en double.

12.4 Houd de DLL stateless

Geen globale COBOL-data over de ABI.

12.5 Gebruik een versie-API

```
PROGRAM-ID. GETVER AS "get_version".  
...  
PROCEDURE DIVISION RETURNING RESULT.
```

13. Build-Script (Windows Batch)

build.bat

```
@echo off  
set COBOL_LIB="C:\RocketCOBOL\lib"  
  
cobol add.cbl /C /O add.obj  
cobol subtract.cbl /C /O subtract.obj  
  
cbllink add.obj subtract.obj ^  
/DLL ^  
/OUT:mymath.dll ^  
/DEF:exports.def ^  
/LIBPATH:%COBOL_LIB% ^  
/NOENTRY  
  
echo Build complete.
```

14. Checklist voor Foutloze DLL's

- ☐ Alle routines hebben CALL-CONVENTION 0
 - ☐ Alle parameters zijn BY VALUE
 - ☐ Types zijn BINARY-LONG / BINARY-DOUBLE
 - ☐ .def-file bevat alleen gewenste exports
 - ☐ DLL linkt met /NOENTRY
 - ☐ C-header is consistent met COBOL-signatures
 - ☐ Testcases in C, Fortran en ML64
-

15. Conclusie

Deze manual biedt een volledig en reproduceerbaar proces voor het bouwen van stabiele, C-ABI-compatibele DLL's met Rocket COBOL 11.0. Door consistente symbol-export, correcte calling conventions en strikte type-discipline kunnen deze DLL's veilig worden aangeroepen vanuit C, Fortran, ML64 en moderne talen zoals Python en C#.