

PDF-versie – Samengevoegd Document (Opnieuw Gegenereerd)

PDF-versie - Samengevoegd Document

Deze pagina bevat de volledige, opnieuw gegenereerde PDF-geschikte versie van het samengevoegde document. De structuur is geoptimaliseerd voor A4-export, consistente marges, vaste typografie en duidelijke secties.

1. Executive Summary

Dit document bundelt de volledige cross-language DLL-architectuur, inclusief het whitepaper, het uitgebreide architectuurnkader en het complete mini-project. Het vormt een toekomstvaste basis voor interoperabiliteit tussen Rocket COBOL 11.0, Intel C, IFX Fortran en ML64-assembly.

2. Architectuurprincipes

- **Single Binary Source of Truth:** één DLL bevat alle kernlogica.
 - **Strict C-ABI Contract:** stabiele interface voor alle talen.
 - **Language Isolation:** COBOL, C en Fortran bevatten alleen dunne wrappers.
 - **Replaceable Core:** DLL kan worden vervangen zonder wijzigingen in de aanroepende talen.
-

3. High-Level Architectuur

```
COBOL Program
|
| CALL USING
v
C Wrapper (extern "C")
|
| normalized parameters
v
DLL Core (C)
|
| dispatch
v
Fortran Routine ----> ML64 Fastpath (optioneel)
|
v
COBOL Array (COMP-2)
```

4. Interface-Contract

4.1 Exportregels

```
EXPORTS
FILL_VALUES
SCALE
core_get_version
```

4.2 Type-Mapping

Concept	COBOL	C	Fortran	ML64
Int32	COMP-5	int32_t	integer(c_int)	eax
Int64	COMP-5	int64_t	integer(c_long_long)	rax
Double	COMP-2	double	real(c_double)	xmm0
Pointer	COMP-5 64-bit	void*	type(c_ptr)	rax

5. Memory-Architectuur

- DLL beheert eigen geheugen.
- Callers beheren inputbuffers.

- Vrijgavefunctie:

```
void core_free(void* p);
```

6. Error-Architectuur

Code	Betekenis
0	OK
1	Invalid argument
2	Null pointer
3	Out of range
10	Memory allocation failure
100-199	Internal errors

7. Logging-Architectuur

```
void core_set_logger(void (*logger)(int level, const char* msg));
```

8. Build-Pipeline

1. ML64 → OBJ
 2. Fortran → OBJ
 3. C → OBJ
 4. Link naar DLL
 5. ABI-check
 6. Unit tests
 7. Publicatie van artefacten
-

9. Teststrategie

- Unit tests per taal
 - Integration tests
 - Memory leak detection
 - Stress-tests
-

10. Mini-Project - Volledige Implementatie

10.1 Bestandsstructuur

```
mini-dll-project\  
fill_values.f90  
core_c.c  
core.def  
prog.cbl  
build.bat
```

10.2 Fortran-routine

```
subroutine FILL_VALUES(array, length, factor, err) bind(C, name="FILL_VALUES")  
  use iso_c_binding  
  implicit none  
  real(c_double), intent(out) :: array(*)  
  integer(c_int), intent(in) :: length  
  real(c_double), intent(in) :: factor  
  integer(c_int), intent(out) :: err  
  integer :: i  
  
  if (length <= 0) then  
    err = 1  
    return  
  end if  
  
  do i = 1, length  
    array(i) = dble(i) * factor  
  end do  
  
  err = 0  
end subroutine FILL_VALUES
```

10.3 C-export

```
__declspec(dllexport)  
double SCALE(double x, double factor)  
{  
  return x * factor;  
}
```

10.4 DEF-file

```
LIBRARY core  
EXPORTS  
FILL_VALUES  
SCALE
```

10.5 COBOL-programma

```
CALL "FILL_VALUES"  
USING WS-VALUE WS-LENGTH WS-FACTOR WS-ERR-CODE
```

10.6 Build-script

```
ifx /c /MD fill_values.f90 /object:fill_values.obj  
icl /c /MD core_c.c /Fo core_c.obj  
link /DLL /OUT:core.dll fill_values.obj core_c.obj /DEF:core.def  
cobol prog.cbl  
link prog.obj core.lib /OUT:prog.exe
```

11. Conclusie

Deze PDF-geschikte versie vormt een volledig, professioneel en toekomstvast fundament voor multi-language interoperabiliteit. Het document bevat zowel de architectuur als een volledig werkend mini-project.