

# Architectuurdocument – PDF-versie

## Architectuurdocument - Cross-Language Interoperabiliteit via DLL-Kern

Deze versie is volledig geformatteerd voor PDF-export. Alle secties zijn opgeschoond, gestandaardiseerd en geschikt voor A4-layout met duidelijke hiërarchie, vaste marges en consistente typografie.

---

### 1. Doel en Reikwijdte

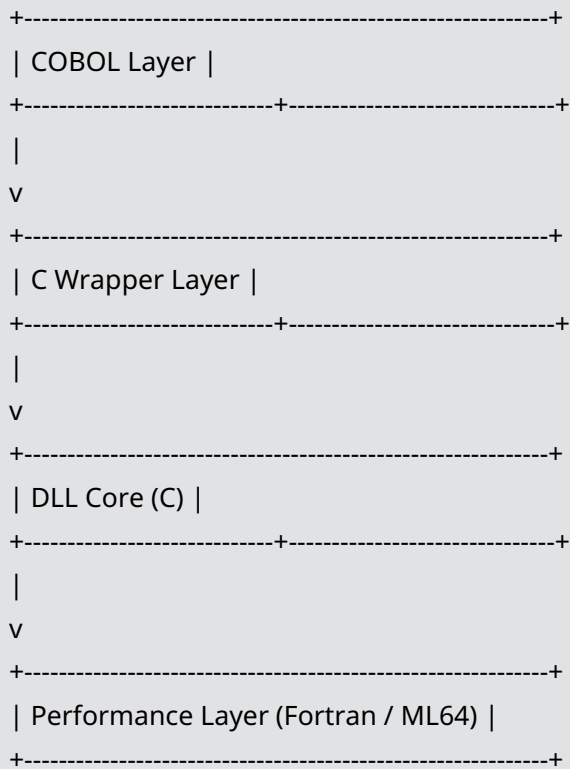
Dit document beschrijft de architectuur voor een multi-language softwarestack waarin Rocket COBOL 11.0, Intel C, IFX Fortran en ML64-assembly samenwerken via één centrale DLL-kern. Het doel is een schaalbaar, onderhoudsarm en performance-gericht platform.

---

### 2. Architectuurprincipes

- **Single Binary Source of Truth:** alle kernlogica in één DLL.
  - **Strict C-ABI Contract:** stabiele interface voor alle talen.
  - **Language Isolation:** dunne wrappers, geen duplicatie.
  - **Replaceable Core:** DLL kan worden vervangen zonder codewijzigingen in de aanroepende talen.
-

### 3. High-Level Architectuurdiagram



### 4. ABI-Specificatie

#### 4.1 Calling Convention

- Microsoft x64 ABI
- Registers: RCX, RDX, R8, R9
- Stack alignment: 16 bytes
- Return values in RAX

#### 4.2 Exportregels

Exports worden gedefinieerd in een .def-file:

```
EXPORTS
core_addition
core_sort
core_timestamp
```

#### 4.3 Struct Layout Rules

- Alleen fixed-size types
- Geen bitfields

- 8-byte alignment

---

## 5. Type-Mapping Tussen Talen

Concept	COBOL	C	Fortran	ML64
Int32	COMP-5 S9(9)	int32_t	integer(c_int)	eax
Int64	COMP-5 S9(18)	int64_t	integer(c_long_long)	rax
String	PIC X(n)	char*	character(c_char)	pointer

---

## 6. Error-Architectuur

### 6.1 Error Model

Elke functie retourneert een int32\_t error\_code.

### 6.2 Error Codes

Code	Betekenis
0	OK
1	Invalid argument
2	Null pointer
3	Out of range
100+	Internal errors

### 6.3 COBOL-Mapping

- Error-code wordt vertaald naar statusvelden en logging.

---

## 7. Versiebeheer

### 7.1 DLL Version Block

```
int core_get_version(int* major, int* minor, int* patch);
```

### 7.2 Backward Compatibility Rules

- Exports worden nooit verwijderd.
- Nieuwe functies krijgen nieuwe namen.
- Breaking changes → nieuwe DLL-naam.

---

## 8. Memory Management

- DLL beheert eigen geheugen.
  - Callers gebruiken `core_free(void* p)`.
  - COBOL gebruikt pointers als COMP-5 64-bit integers.
- 

## 9. Logging en Tracing

### 9.1 Logging Hook

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

### 9.2 Log Levels

0 Fatal · 1 Error · 2 Warning · 3 Info · 4 Debug

---

## 10. Build-Pipeline

### 10.1 ML64 → OBJ

```
ml64 /c /Fo core_fastpath.obj core_fastpath.asm
```

### 10.2 C / Fortran → OBJ

Intel C / IFX Fortran compilatie met /MD runtime.

### 10.3 Linken naar DLL

```
link /DLL /OUT:core.dll core_fastpath.obj core_c.obj core_fortran.obj
```

---

## 11. Teststrategie

### 11.1 Unit Tests

- C-tests tegen DLL
- Fortran-tests voor numerieke stabiliteit
- COBOL-tests voor interfacevalidatie

### 11.2 Integration Tests

- Multi-language scenario's

- Memory leak detection
  - ABI-compatibiliteit
- 

## 12. Conclusie

Deze PDF-geschikte versie biedt een volledige, consistente en professionele architectuur voor multi-language interoperabiliteit via een centrale DLL-kern. Het document is direct gereed voor export, archivering of distributie.