

# ADR-07 — Fortran 95 DLL Interoperabiliteitsstrategie

## ADR-07 - Fortran 95 DLL Interoperabiliteitsstrategie

### Status

Goedgekeurd - Dit besluit is vastgesteld als standaard voor alle numerieke kernels en interoperabiliteitscomponenten binnen de multi-language architectuur (Fortran 95, Rocket COBOL, Intel C, ML64).

---

### 1. Context

Het project vereist een stabiele, taal-agnostische ABI-laag die door meerdere talen gebruikt kan worden, waaronder Rocket COBOL 11.0, Intel C (CL), ML64-assembly en toekomstige bindings voor Python, C#, Rust. Fortran wordt ingezet voor numerieke kernels vanwege performance en legacy-compatibiliteit.

Moderne Fortran-modules genereren compiler-specifieke .mod-bestanden, veroorzaken symbol-mangling en zijn niet bruikbaar buiten Fortran-compilers. Daarom is onderzocht of Fortran 95 zonder modules, met uitsluitend C-ABI-compatibele DLL-exports, een geschikte strategie is.

---

### 2. Beslissing

Het project kiest voor een Fortran 95-architectuur zonder modules, waarbij:

1. Alle procedures worden gedefinieerd als bind(C) voor een stabiele C-ABI.
2. Alle functionaliteit wordt aangeboden via een DLL-exportlaag.
3. Geen gebruik wordt gemaakt van Fortran-modules.
4. Interfaces worden beheerd via C-headers, COBOL-copybooks en ML64-prototypes.

Deze strategie wordt toegepast op alle numerieke kernels en interoperabiliteitscomponenten waar ABI-stabiliteit en multi-language integratie belangrijker zijn dan moderne Fortran-features.

---

### 3. Motivatie

#### 3.1 Maximale interoperabiliteit

Een pure C-ABI-laag is de enige interface die door alle betrokken talen wordt ondersteund.

#### 3.2 Compiler-onafhankelijkheid

.mod-bestanden zijn niet gestandaardiseerd en verschillen per compiler en versie. DLL-exports met C-ABI zijn langdurig stabiel.

### 3.3 Volledige controle over symbolen

Elke export is expliciet en voorspelbaar, zonder verborgen module-symbolen of naam-mangling.

### 3.4 Geschikt voor numerieke kernels

De Fortran-code is klein, functioneel en numeriek; modules bieden hier weinig extra waarde.

---

## 4. Gevolgen

### 4.1 Positieve gevolgen

- Stabiele ABI voor alle talen
- Geen compiler-lock-in
- Eenvoudige integratie met COBOL, C, ML64, Python, C#, Rust
- Volledige controle over exports en symbolen
- Eenvoudige deployment (DLL + header)

### 4.2 Negatieve gevolgen

- Geen interface-controle door Fortran zelf
  - Geen moderne Fortran-features zoals modules en submodules
  - Meer boilerplate in interface-definities
  - Hogere foutkans bij verkeerde argumenten (runtime-detectie)
- 

## 5. Alternatieven

### A. Fortran 95 met modules

Voordelen: type-veiligheid, encapsulatie, moderne features. Nadelen: .mod-bestanden zijn niet bruikbaar buiten Fortran.

### B. Mixed approach (modules intern, C-exports extern)

Voordelen: interne structuur + externe interoperabiliteit. Nadelen: complexere build-chain, dubbele interface-definitie.

### C. Alles in C implementeren

Voordelen: maximale interoperabiliteit. Nadelen: verlies van Fortran-performance en numerieke stabiliteit.

---

## 6. Besluitcriteria

Deze strategie is gekozen omdat:

- Interoperabiliteit belangrijker is dan taalfeatures.
  - ABI-stabiliteit belangrijker is dan module-functionaliteit.
  - Multi-language integratie een harde eis is.
  - De Fortran-code klein en numeriek is.
  - De C-laag de centrale interface vormt.
- 

## 7. Implementatie-richtlijnen

- Alle procedures krijgen een C-compatibele interface via `bind(C)`.
  - Parameters zijn flat, POD-types of C-structs.
  - Memory-ownership wordt expliciet vastgelegd in de C-header.
  - COBOL-copybooks worden gegenereerd vanuit de C-header.
  - ML64-prototypes volgen exact dezelfde C-signatures.
  - De DLL-exportlijst wordt versie-gecontroleerd en nooit gebroken.
- 

## 8. Conclusie

Deze strategie biedt maximale interoperabiliteit en ABI-stabiliteit, met als nadeel het verlies van moderne Fortran-features en automatische interface-controle. Voor een multi-language architectuur met numerieke kernels is dit een rationele en duurzame keuze.