

Handling C++ Exception Hierarchies in Ada

Alexandre Oliva

<https://www.lx.oliva.nom.br/>

oliva@gnu.org, oliva@adacore.com

AdaCore

GNU Tools Cauldron, September, 2025

Copyright 2025 AdaCore (last changed in September 2025)

This work is licensed under the [Creative Commons BY-SA 4.0 International](https://creativecommons.org/licenses/by-sa/4.0/) License.

[git://git.lx.oliva.nom.br/blog](https://git.lx.oliva.nom.br/blog) pres/adacxxcept/



Summary

- Exception Models
- Ada/C++ interoperation
 - Before
 - After
- Base-class type-matching extensions
- Using Run-Time Type Information

Ada Exceptions

E : exception; -- *declared explicitly, flat*

E2 : exception; -- *no associated app data*

begin [...] raise E [*with "Message"*];

exception

when X : **E** | E2 => null; -- *handler*

when **E** => [...] -- *error, duplicate E*

when others => raise; -- *catch-all; reraise*

C++ Exceptions

```
struct E3 {[...]};    // unmarked data carriers
```

```
struct E4 : E3 {[...]};    // hierarchical
```

```
try {
```

```
    throw E4 ();
```

```
    } catch (E3 &x) {    // handle
```

```
    } catch (E4 &) {    // unreachable
```

```
    } catch (...) { throw; } // catch-all; reraise
```

Ada/C++ Interoperation

```
type E3 is tagged limited record [...]
```

```
end record; pragma Import (Cxx, E3);
```

```
type E4 is limited new E3 with record [...]
```

```
end record; pragma Import (Cxx, E4);
```

```
Ex3 : exception;
```

```
pragma Import (Cxx, Ex3, "_ZTI2E3");
```

- Separate data types and exceptions

C++ Exceptions with GNAT (then)

function Get_E3 is new -- *generic instantiation*

GNAT.CPP_Exceptions.Get_Object (E3);

exception

when X : **Ex3** => -- *exact match*

Var : E3 := Get_E3 (X); -- *wouldn't slice*

- Won't handle E4 or other E3-derived types
- This generic requires non-**limited** types

Exact matches (now)

exception

when $X : \text{Ex3} \Rightarrow$

declare

```
Obj : E3;           -- known full type
```

```
pragma Import (Ada, Obj);      -- no init
```

for Obj'Address use GNAT.CPP_Exceptions

```
.Get_Object_Address (X);    -- in place
```

Hierarchical matches (now)

Eb3 : exception;

pragma Import (Cpp, Eb3, “_ZTI2E3'**Class**”);

function Get_E3a is new GNAT.CPP_Exceptions
 .Get_Access_To_Tagged_Object (E3);

exception

when X : **Eb3** => *-- is-a, base-type match*

Var : **access E3'Class** := Get_E3a (X);

Foreign others matches (then)

when X : **others** =>

if Is_Foreign_Exception (X) and then

Exception_Language (X) = EL_Cpp and then

(??? is it an E3?) *-- ???*

then

declare

Obj : E3 := Get_E3 (X); *-- no check*

Foreign others matches (now)

when X : **others** =>

if Is_Foreign_Exception (X) and then

Exception_Language (X) = EL_Cpp and then

Name (Get_Type_Info (X)) = "2**E3**" -- *check*

then

declare

Ref : **access** E3 := Get_E3a (X);

Current Limitations

- Unrelated exceptions and data types
 - Room for unsafe type conversions
- One-way compatibility
 - Cannot raise actual C++ objects from Ada
 - Raise_Cpp_Exception vs **limited** types

Extensions

- Access to raised object in place
 - Safe reraise, postponing object release
- Exception import “_ZTI*’Class” syntax
 - New “Foreign_Exception” kind
- Handlers for base and derived allowed
- C++ “std::type_info*” exposed in Ada
- Full object for “others” handlers

Thank you!

AdaCore

<https://www.lx.oliva.nom.br/>

oliva@gnu.org, oliva@adacore.com

Questions?

