

```
package Alerts.Low is
  type Low_Alert is new Alert with private;
  procedure Log (A : access Low_Alert);
  private
    ...
  end Alerts.Low;
```

```
with Alerts.Pool; use Alerts.Pool;
package body Alerts.Low is
  ...
  begin
    Register (new Low_Alert);
  end Alerts.Low;
```

<http://libre.act-europe.fr>

© ACT Europe under the GNU Free Documentation License

64

Remote_Types

An Example

```
with Alerts.Pool; use Alerts.Pool;
package Alerts is
  type Low_Alert is new Alert with private;
  procedure Log (A : access Low_Alert);
  private
    ...
  end Alerts.Low;
```

<http://libre.act-europe.fr>

© ACT Europe under the GNU Free Documentation License

62

```
package Alerts is
  type Alert is abstract tagged private;
  type Alert_Ref is access all Alert'Class;
  procedure Handle (A : access Alert);
  procedure Log (A : access Alert) is abstract;
  private
    ...
  end Alerts;
```

Write App

```
package Alerts.Pool is
  procedure Register (A : Alert_Ref);
  function Get_Alert return Alert_Ref;
  end Medium;
  ...
  with Alerts_Alerts.Pool; use Alerts;
  procedure Process_Alerts is
    begin
      loop
        Handle (Pool.Get_Alert);
      end loop;
    end Process_Alerts;
```

```
with Alerts.Pool; use Alerts.Pool;
package body Alerts is
  type Medium_Alert is new Alert with private;
  procedure Handle (A : access Medium_Alert);
  procedure Log (A : access Medium_Alert);
  private
    ...
  end Alerts.Medium;
```

<http://libre.act-europe.fr>

© ACT Europe under the GNU Free Documentation License

65

```
package Alerts.Medium is
  type Medium_Alert is new Alert with private;
  procedure Handle (A : access Medium_Alert);
  procedure Log (A : access Medium_Alert);
  private
    ...
  end Alerts.Medium;
```

```
with Alerts.Pool; use Alerts.Pool;
package body Alerts.Medium is
  ...
  begin
    Register (new Medium_Alert);
  end Alerts.Medium;
```

© ACT Europe under the GNU Free Documentation License

<http://libre.act-europe.fr>

Build & Test

```
package Alerts is
  pragma Remote_Types;
  type Alert is abstract tagged private;
  type Alert_Ref is access all Alert'Class;
  procedure Handle (A : access Alert);
  procedure Log (A : access Alert);
private
  ...
end Alerts;
```

```
package Alerts.Low is
  pragma Remote_Types;
  type Low_Alert is new Alert with private;
  procedure Handle (A : access Medium_Alert);
  procedure Log (A : access Medium_Alert);
private
  ...
end Alerts.Low;
```

```
package Alerts.Medium is
  pragma Remote_Types;
  type Medium_Alert is new Alert with private;
  procedure Handle (A : access Medium_Alert);
  procedure Log (A : access Medium_Alert);
private
  ...
end Alerts.Medium;
```

```
with Alerts, Alerts.Pool; use Alerts;
procedure Register (A : Alert_Ref);
begin
  loop
    Handle (Pool.Get_Alert);
    end loop;
  end Process_Alerts;
```

```
package Alerts.Pool is
  pragma Remote_Call_Interface;
  procedure Register (A : Alert_Ref);
  function Get_Alert return Alert_Ref;
end Medium;
```

Categorize

```
package Alerts is
  pragma Remote_Types;
```

```
type Alert is abstract tagged private;
type Alert_Ref is access all Alert'Class;
procedure Handle (A : access Alert);
procedure Log (A : access Alert) is abstract;
private
```

```
package Alerts.Pool is
```

```
  pragma Remote_Call_Interface;
  procedure Register (A : Alert_Ref);
  function Get_Alert return Alert_Ref;
end Medium;
```

```
with Alerts, Alerts.Pool; use Alerts;
procedure Process_Alerts is
begin
  loop
    Handle (Pool.Get_Alert);
    end loop;
  end Process_Alerts;
```

<http://libre.act-europe.fr>

Partition

```
configuration Config_2 is
  Node_AL : Partition := (Alerts.Low);
  Node_AM : Partition := (Alerts.Medium);
  Node_B : Partition := (Alerts.Pool);
  Node_C : Partition := (Process_Alerts);
end Config_2;
```

```
package Alerts.Low is
```

```
  pragma Remote_Types;
  type Low_Alert is new Alert with private;
  procedure Log (A : access Low_Alert);
private
  ...
end Alerts.Low;
```

```
package Alerts.Medium is
  pragma Remote_Types;
  type Medium_Alert is new Alert with private;
  procedure Handle (A : access Medium_Alert);
  procedure Log (A : access Medium_Alert);
private
  ...
end Alerts.Medium;
```

What Happens When Executing the Distributed Program ?

70

© ACT Europe under the GNU Free Documentation License

<http://libre.act-europe.fr>

```
package Alerts.Low is
pragma Remote_Types;
type Medium_Alert is new Alert with private;
procedure Handle (A : access Medium_Alert);
procedure Log (A : access Medium_Alert);
private
...
end Alerts.Low;
```

Node_AM

```
package Alerts.Pool is
pragma Remote_Types;
type Low_Alert is new Alert with private;
procedure Register (A : Alert_Ref);
function Get_Alert return Alert_Ref;
end Medium;
```

Node_AL

Step 2: A Medium_Alert object in Node_AM registers itself with Node_B

72

© ACT Europe under the GNU Free Documentation License

<http://libre.act-europe.fr>

```
package Alerts.Pool is
pragma Remote_Call_Interface;
procedure Register (A : Alert_Ref);
function Get_Alert return Alert_Ref;
end Medium;
```

Node_C

```
package Alerts.Pool is
pragma Remote_Types;
type Low_Alert is new Alert with private;
procedure Register (A : Alert_Ref);
function Get_Alert return Alert_Ref;
end Medium;
```

Node_B

```
package Alerts.Medium is
pragma Remote_Types;
type Medium_Alert is new Alert with private;
procedure Handle (A : access Medium_Alert);
procedure Log (A : access Medium_Alert);
private
...
end Alerts.Medium;
```

Node_AM

```
package Alerts.Pool is
pragma Remote_Types;
type Low_Alert is new Alert with private;
procedure Register (A : Alert_Ref);
function Get_Alert return Alert_Ref;
end Medium;
```

Node_AL

Step 3: Process_Alerts in Node_C does an RPC to Get_Alert in Node_B

73

© ACT Europe under the GNU Free Documentation License

<http://libre.act-europe.fr>

```
with Alerts.Alerts.Pool; use Alerts;
procedure Process_Alerts is
begin
loop
Handle (Pool.Get_Alert);
end loop;
end Process_Alerts;
```

Node_C

```
package Alerts.Medium is
pragma Remote_Types;
type Medium_Alert is new Alert with private;
procedure Handle (A : access Medium_Alert);
procedure Log (A : access Medium_Alert);
private
...
end Alerts.Medium;
```

Node_AM

```
package Alerts.Low is
pragma Remote_Types;
type Low_Alert is new Alert with private;
procedure Log (A : access Low_Alert);
private
...
end Alerts.Low;
```

Node_AL

Step 1: A Low_Alert object in Node_AL registers itself with Node_B

71

© ACT Europe under the GNU Free Documentation License

<http://libre.act-europe.fr>

```
with Alerts.Alerts.Pool; use Alerts;
procedure Process_Alerts is
begin
loop
Handle (Pool.Get_Alert);
end loop;
end Process_Alerts;
```

Node_B

What Does Get_Alert Return ?

Get_Alert → Pointer

Address of
Alert object
on the
Machine

Machine

© ACT Europe under the GNU Free Documentation License

<http://libre.act-europe.fr>

76

Remote Access to Class Wide Type

- At compile time:

– You do not know what operation you'll dispatch to

– On what node that operations will be executed on

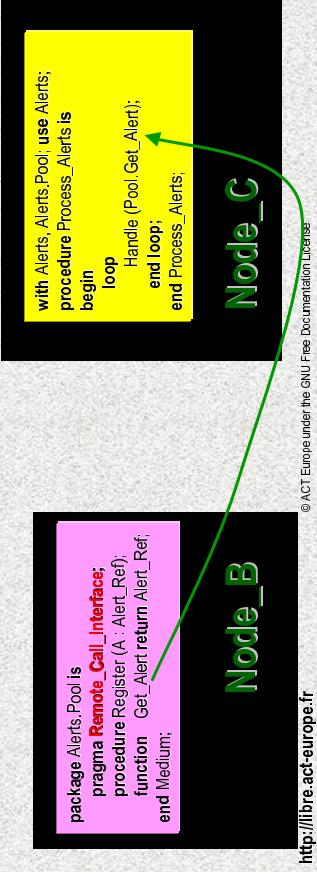
```
package Alerts.Medium is
  pragma Remote_Types;
  type Medium_Alert is new Alert with private;
  procedure Handle (A : access Medium_Alert);
  procedure Log (A : access Medium_Alert);
private
  ...
end Alerts.Medium;
```

Node_AM

```
package Alerts.Low is
  pragma Remote_Types;
  type Low_Alert is new Alert with private;
  procedure Register (A : access Low_Alert);
  ...
end Alerts.Low;
```

Node_AL

Step 4: Get_Alert returns a pointer to an Alert object (Low_Alert or Medium_Alert)



```
package Alerts.Medium is
  pragma Remote_Types;
  type Medium_Alert is new Alert with private;
  procedure Handle (A : access Medium_Alert);
  procedure Log (A : access Medium_Alert);
private
  ...
end Alerts.Medium;
```

Node_AM

```
package Alerts.Low is
  pragma Remote_Types;
  type Low_Alert is new Alert with private;
  procedure Register (A : access Low_Alert);
  ...
end Alerts.Low;
```

Node_AL ?

Step 5: Node_C performs a dispatching RPC. It calls Handle in Node_AL or Node_AM

```
with Alerts.Alerts.Pool; use Alerts;
procedure Process_Alerts is
begin
  loop
    Handle (Pool.Get_Alert);
    end loop;
  end Process_Alerts;
```

Node_C

```
package Alerts.Pool is
  pragma Remote_Call_Interface;
  procedure Register (A : Alert_Ref);
  function Get_Alert return Alert_Ref;
end Medium;
```

Node_B

© ACT Europe under the GNU Free Documentation License

http://libre.act-europe.fr

75