Event Handling

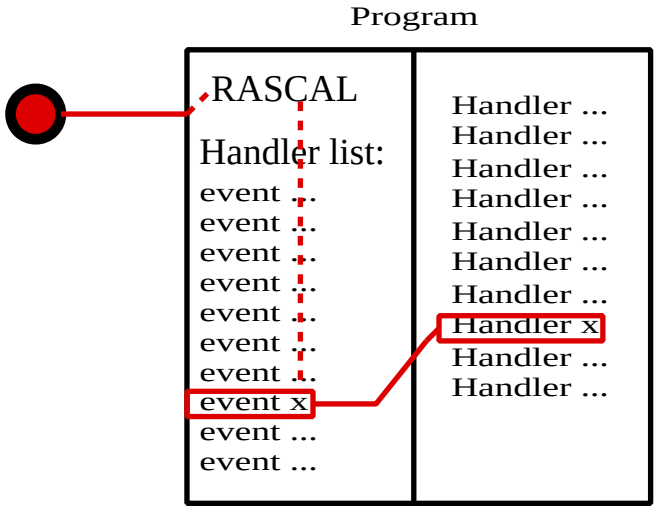
What is a event ?

An event is a signal that the user or another program has done something you may want to react upon. There are currently approximately 200 predefined events known by RASCAL all of which can be divided into 3 groups:

- **Wimp events**
  When the user has done something affecting a program's windows/icons the program gets these events.
- **Toolbox events**
  When the user has done something affecting a program's windows/icons and these are controlled by the Toolbox then the program gets these events.
- **Messages events**
  Other programs and the operating system can send messages to your program.

What is a handler ?

To receive these events you must tell RASCAL what to do when such an event occurs. This is what a Handler or Listener is for. It is a procedure associated with an event and after a handler has been registered with RASCAL its procedure will be called every time its event occurs.



The program receives an event and RASCAL looks in its handler list for a handler for that particular event. There is one and RASCAL calls the handler's procedure.

If RASCAL receives an event but can not find a handler for that event, the event will be ignored.

Creating a handler

A handler consists of an event and a procedure, how exactly the procedure has to look is defined in an *abstract* handler.

```
procedure Handle (The : in Event_Listener) is abstract;
```

The handler procedure must be an implementation of the above this procedure. The

'Event_Listener' is the event type and should be replaced by the type of the event you want to handle.
All events RASCAL knows of are defined as abstract types and to use these definitions you must therefore define a non-abstract version of them.

An example:
Let us assume you want to receive the quit message which is send by the operating system when a user tries to quit your application from the taskmanager. The type defined by RASCAL is 'AMEL_Message_Quit' and defined in the 'TaskManager' package.

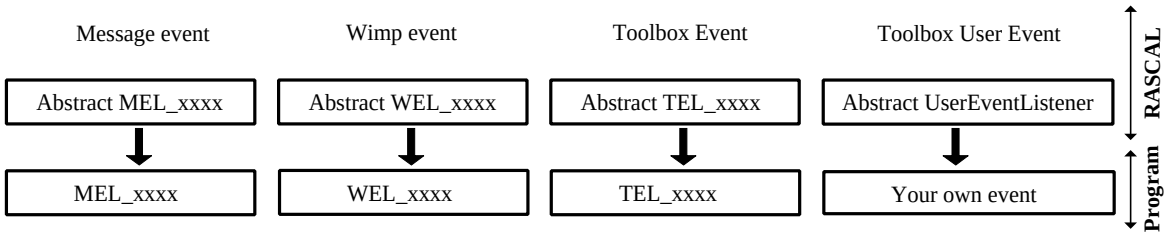
```
with TaskManager; use TaskManager;

package Example is
   type MEL_Message_Quit is
        new AMEL_Message_Quit with null record;

   procedure Handle (The : in MEL_Message_Quit);

end Example;
```

Defining a event handler

All that is left now is to register this event handler with RASCAL which you should put into the main program before the multi-tasking is started (Poll procedure call):

```
    Add_Listener (Main_Task, new MEL_Message_Quit);
```

As you may have guessed this event is a Message event. Predefined Message event types are prefixed with 'AMEL' meaning 'Abstract Message Event Listener'. The prefixes for Wimp event types and Toolbox event types are 'AWEL' and 'ATEL'.



Event listener hierarchy

The predefined event types form the top row in the above diagram. The lower row consists of your implementation of the abstract event types.

If you want to create an toolbox event handler for a user event you have created in ResEd then you should extend the abstract UserEventListener.

Example:
Let us assume you want to handle the event ’MyOwnEvent’ which has the event nr 12345 and is a Toolbox event you have ’created’ in ResEd.

```
type TEL_MyOwnEvent_Type is
   new Toolbox_UserEventListener(12345,-1,-1) with null
record;

procedure Handle (The : in TEL_MyOwnEvent_Type);
```

And to add it to the handler list:

```
    Add_Listener (Main_Task, new TEL_MyOwnEvent_Type);
```