UVM Reporting

UVM Callbacks

UVM Message facilities

- Rich set of message-display commands & methods - alter the numbers & types of messages that are displayed without re-compilation of the design
- Severity
  - **Severity indicates importance**
  - **Examples are Fatal, Error, Warning & Info**
- Verbosity
  - **Verbosity indicates filter level**
  - **Examples are None, Low, Medium, High, Full & Debug**
- Simulation Handling Behavior
  - **Simulation handling behavior controls simulator behavior**
  - **Examples are Exit, Count, Display, Log etc**

UVM Message facilities

- Macros involved:
  - **`uvm_info(string ID, string MSG, verbosity);**
  - **`uvm_error(string ID, string MSG);**
  - **`uvm_warning(string ID, string MSG);**
  - **`uvm_fatal(string ID, string MSG);**

Severity	Default Simulator Behavior/Action
UVM_FATAL	UVM_DISPLAY | UVM_EXIT
UVM_ERROR	UVM_DISPLAY | UVM_COUNT
UVM_WARNING	UVM_DISPLAY
UVM_INFO	UVM_DISPLAY

UVM Message facilities

- Simulator Behavior/Actions Description

Simulator Action	Description
UVM_EXIT	Exit from simulation immediately
UVM_COUNT	Increment global error count
UVM_DISPLAY	Display message on console
UVM_LOG	Captures messages in a named file
UVM_CALL_BACK	Calls callback method
UVM_NO_ACTION	Do nothing

UVM Message facilities

- Controlling Messages Verbosity:
- In case of default Verbosity level i.e. UVM_MEDIUM, any messages with UVM_HIGH or above are filtered out.

Verbosity Level	Value
UVM_NONE	0
UVM_LOW	100
UVM_MEDIUM	200
UVM_HIGH	300
UVM_FULL	400
UVM_DEBUG	500

- ***A message with the Verbosity level UVM_NONE can not be disabled.***
- One can set the Verbosity level of a uvm_component individually or hierarchically using following commands:
  - **drv.set_report_verbosity_level(UVM_HIGH);**
  - **env.set_report_verbosity_level_hier(UVM_FULL);**

UVM Message facilities

```
import uvm_pkg::*;
class rpting extends uvm_component;
  `uvm_component_utils(rpting)
  function new(string name,uvm_component parent);
    super.new(name, parent);
  endfunction
  task run();
    uvm_report_info(get_full_name(),"Info Message : Verbo level - UVM_NONE  ",UVM_NONE,`__FILE__,`__LINE__);
    uvm_report_info(get_full_name(),"Info Message : Verbo level - UVM_LOW   ",UVM_LOW);
    uvm_report_info(get_full_name(),"Info Message : Verbo level - 150       ",150);
    uvm_report_info(get_full_name(),"Info Message : Verbo level - UVM_MEDIUM",UVM_MEDIUM);
    uvm_report_warning(get_full_name(),"Warning Messgae from rpting",UVM_LOW);
    uvm_report_error(get_full_name(),"Error Message from rpting \n\n",UVM_LOW);
  endtask
endclass
```

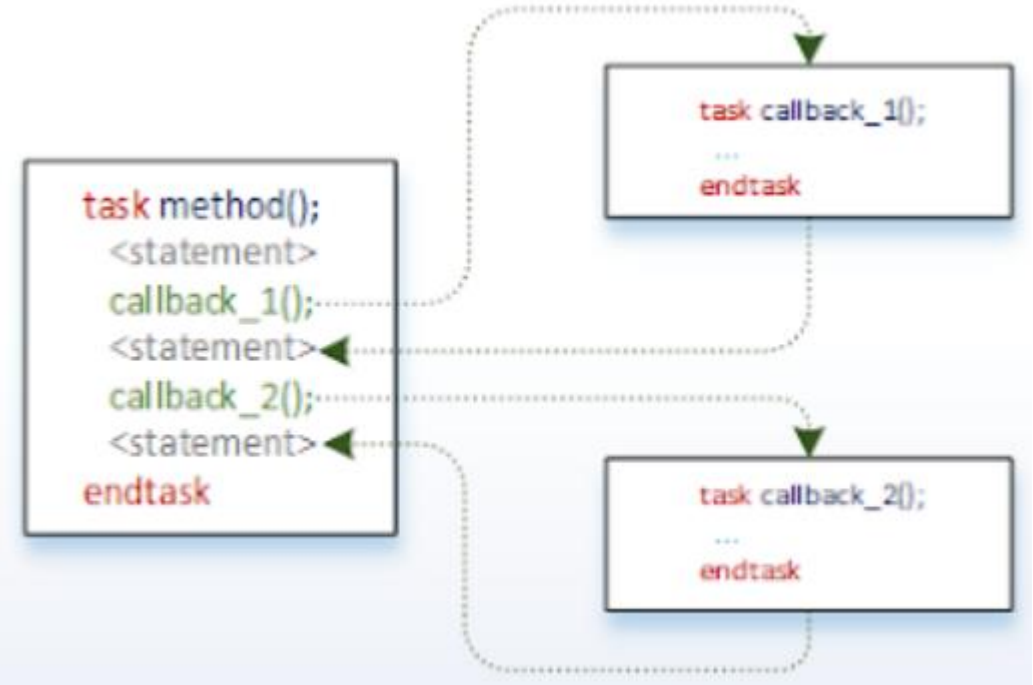
```
module top;

 rpting rpt1;
 rpting rpt2;
 rpting rpt3;

 initial begin
   rpt1 = new("rpt1",null);
   rpt2 = new("rpt2",null);
   rpt3 = new("rpt3",null);

   rpt1.set_report_verbosity_level(UVM_MEDIUM);
   rpt2.set_report_verbosity_level(UVM_LOW);
   rpt3.set_report_verbosity_level(UVM_NONE);
   run_test();
 end
endmodule
```

UVM Message facilities

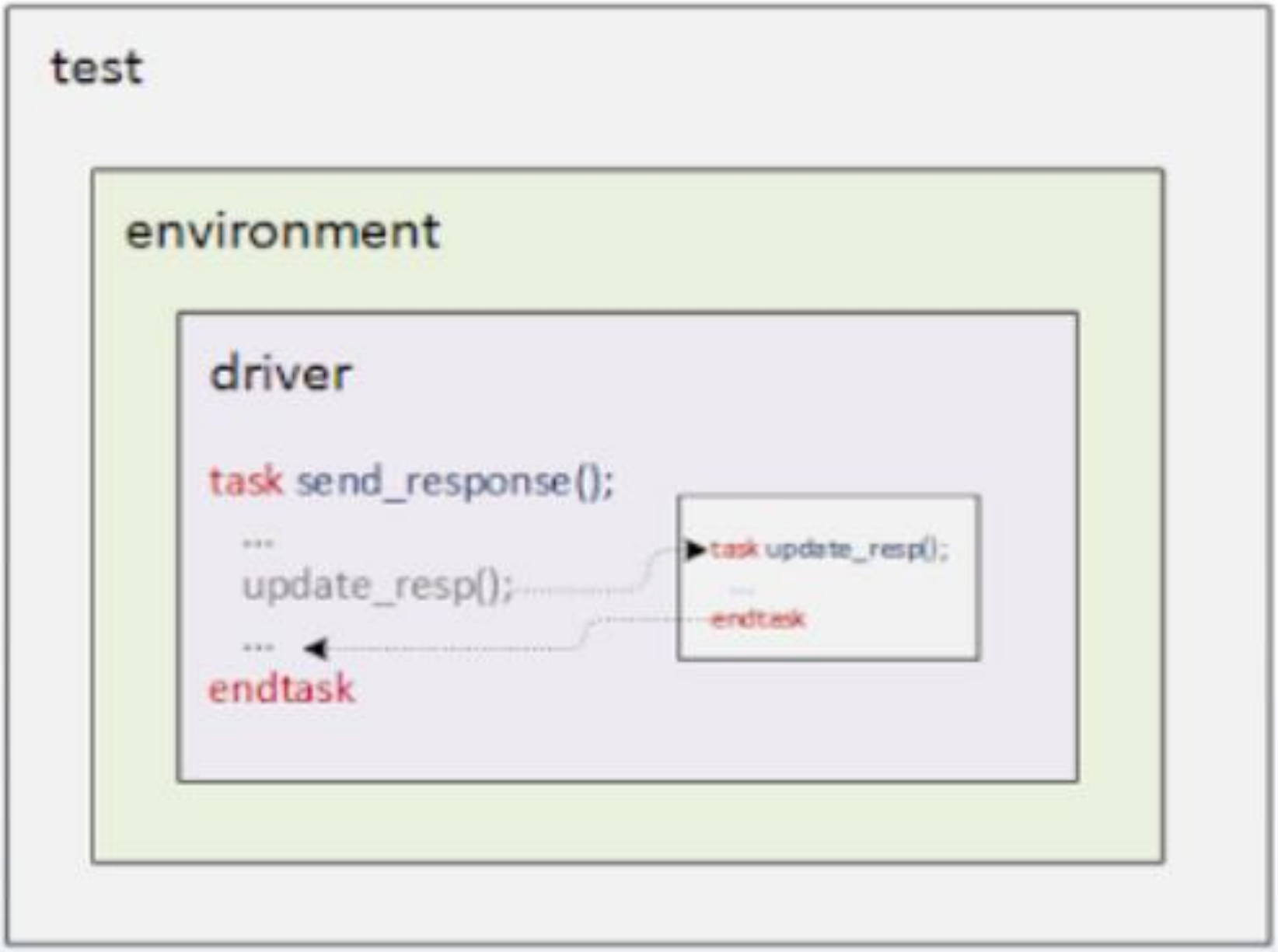
```
UVM_INFO testbench.sv(12) @ 0: rpt1 [rpt1] Info Message : Verbo level - UVM_NONE
UVM_INFO @ 0: rpt1 [rpt1] Info Message : Verbo level - UVM_LOW
UVM_INFO @ 0: rpt1 [rpt1] Info Message : Verbo level - 150
UVM_INFO @ 0: rpt1 [rpt1] Info Message : Verbo level - UVM_MEDIUM
UVM_WARNING @ 0: rpt1 [rpt1] Warning Messgae from rpting
UVM_ERROR @ 0: rpt1 [rpt1] Error Message from rpting

UVM_INFO testbench.sv(12) @ 0: rpt2 [rpt2] Info Message : Verbo level - UVM_NONE
UVM_INFO @ 0: rpt2 [rpt2] Info Message : Verbo level - UVM_LOW
UVM_WARNING @ 0: rpt2 [rpt2] Warning Messgae from rpting
UVM_ERROR @ 0: rpt2 [rpt2] Error Message from rpting

UVM_INFO testbench.sv(12) @ 0: rpt3 [rpt3] Info Message : Verbo level - UVM_NONE
UVM_INFO /apps/vcsmx/vcs/Q-2020.03-SP1-1//etc/uvm-1.2/src/base/uvm_report_server.svh(894) @ 0: reporter [UVM/REPORT/SERVER]
```

UVM Callback

- Callbacks are empty methods with a call to them.
- They are hooks to execute code that gets defined later.
  - Can be implemented in an object or component
  - UVM provides set of classes, methods and macros to implement callbacks.



Callback example



Example: Driver without callback

```
class driver extends uvm_component;

  `uvm_component_utils(driver)

  function new(string name, uvm_component parent);
    super.new(name,parent);
  endfunction

  task run_phase(uvm_phase phase);
      drive_pkt();
  endtask

  task drive_pkt();
    `uvm_info("DRIVER","Inside drive_pkt method",UVM_LOW);
  endtask

endclass
```

```
class environment extends uvm_env;
  driver driv;

  `uvm_component_utils(environment)

  function new(string name, uvm_component parent);
    super.new(name,parent);
  endfunction

  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    driv = driver::type_id::create("driv", this);
  endfunction

endclass
```

Example: Driver without callback

```
class basic_test extends uvm_test;
  environment env;

  `uvm_component_utils(basic_test)

  function new(string name = "basic_test", uvm_component parent=null);
    super.new(name,parent);
  endfunction

  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    env = environment::type_id::create("env", this);
  endfunction
endclass
```

```
`include "driver.sv"
`include "environment.sv"
`include "basic_test.sv"

program testbench_top;

  initial begin
    run_test();
  end

endprogram
```

```
UVM_INFO driver.sv(18) @ 0: uvm_test_top.env.driv [DRIVER] Inside drive_pkt method
```

Example: Adding callback support

```
class driver extends uvm_component;

  `uvm_component_utils(driver)
  `uvm_register_cb(driver,driver_callback)

  function new(string name, uvm_component parent);
    super.new(name,parent);
  endfunction

  task run_phase(uvm_phase phase);
      `uvm_do_callbacks(driver,driver_callback,pre_drive());

      drive_pkt();

      `uvm_do_callbacks(driver,driver_callback,post_drive());
  endtask

  task drive_pkt();
    `uvm_info("DRIVER","Inside drive_pkt method",UVM_LOW);
  endtask

endclass
```

```
class driver_callback extends uvm_callback;

  `uvm_object_utils(driver_callback)

  function new(string name = "driver_callback");
    super.new(name);
  endfunction

  virtual task pre_drive; endtask
  virtual task post_drive; endtask
endclass
```

Example: Adding callback support

```
class user_callback extends driver_callback;

  `uvm_object_utils(user_callback)

  function new(string name = "user_callback");
    super.new(name);
  endfunction

  task pre_drive;
    `uvm_info("USER_CALLBACK","Inside pre_drive method",UVM_LOW);
  endtask

  task post_drive;
    `uvm_info("USER_CALLBACK","Inside post_drive method",UVM_LOW);
  endtask
endclass
```

```
class basic_test extends uvm_test;
  environment env;

  `uvm_component_utils(basic_test)

  function new(string name = "basic_test", uvm_component parent=null);
    super.new(name,parent);
  endfunction

  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    env = environment::type_id::create("env", this);
  endfunction
endclass
```

Example: Adding callback support

```
class user_callback_test extends basic_test;
  user_callback callback_1;

  `uvm_component_utils(user_callback_test)

  function new(string name = "user_callback_test", uvm_component parent=null);
    super.new(name,parent);
  endfunction

  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    callback_1 = user_callback::type_id::create("callback_1", this);

    uvm_callbacks#(driver,driver_callback)::add(env.driv,callback_1);
  endfunction
endclass
```

```
class environment extends uvm_env;
  driver driv;

  `uvm_component_utils(environment)

  function new(string name, uvm_component parent);
    super.new(name,parent);
  endfunction

  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    driv = driver::type_id::create("driv", this);
  endfunction

endclass
```

Example: Adding callback support

```
import uvm_pkg::*;

`include "driver_callback.sv"
`include "driver.sv"
`include "environment.sv"
`include "basic_test.sv"

`include "user_callback.sv"
`include "user_callback_test.sv"

program testbench_top;

  initial begin
    run_test();
  end

endprogram
```

```
UVM_INFO user_callback.sv(14) @ 0: reporter [USER_CALLBACK] Inside pre_drive method
UVM_INFO driver.sv(23) @ 0: uvm_test_top.env.driv [DRIVER] Inside drive_pkt method
UVM_INFO user_callback.sv(18) @ 0: reporter [USER_CALLBACK] Inside post_drive method
```