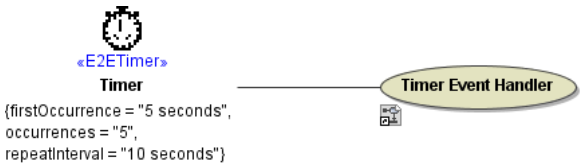
Timer Service

A class assigned the stereotype <<E2ETimer>> models a Timer. An activity assigned to the Timer class models the Timer event handler. The internal Bridge scheduler process executes the for each Timer event.

This page gives an overview of the components required to build a Timer Service.

Figure: Timer Use Case Diagram



Example File (Builder project Add-ons/Scheduling):

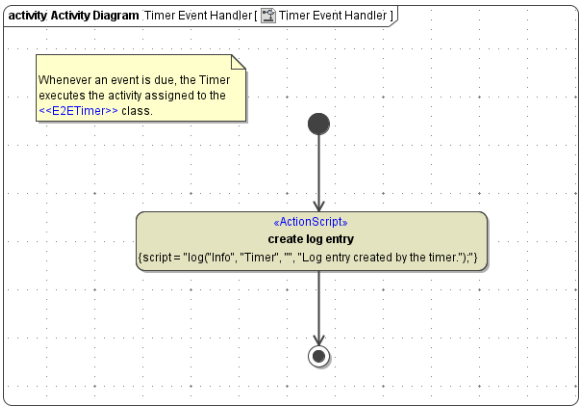
	<your example path>\Add-ons\Scheduling\uml\timer.xml

On this Page:

Timer Event Handler

Whenever an event is due, the Timer executes the activity assigned to the <<E2ETimer>> class. The following example creates a string, converts the string to a blob, and writes the blob to a file.

Figure: Activity Example of Calling a Time-Triggered Event



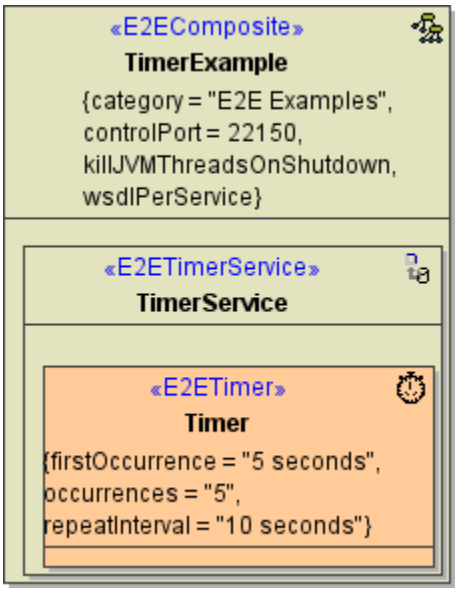
The tagged values of an instance of the <<E2ETimer>> class define when the activity above is executed. The deployment diagram specifies the tagged values.

Timer Components

A Timer service (**TimerService**) does not have any external interface, so the class resident on the <<E2E TimerService>> contains no operation. This class (**Timer**) must have the stereotype <<E2ETimer>>; that is, it must be a Timer.

There is one exception, when a Timer service has an external interface: This is a trace interface, which is a SOAP interface to the activity assigned to the Timer class for debugging purposes.The compiler generates automatically the **shadow** interface and the corresponding **shadow** service when compiling the service. The Analyzer and the Interactive Debugger display the interface as a SOAP operation (for more information see Testing of Non-SOAP Services).

Figure: Timer Component Diagram



The **TimerExample** composite contains a Timer service (<<E2ETimerService>>). The Timer service instance contains a class (**Timer**).

The Timer artifact has the following tagged values that define when the event handler is executed.

Tagged Value	Description	Default	Mandatory
repeatInterval	must be a valid time duration expression (see Time Durations) and defines the break between the repetitions of the defined action in the called activity diagram.		mandatory
occurrences	must be a positive integer and defines the count of repetitions. If the time event shall be repeated forever, set occurrences to always.	always	
firstOccurrence	must be a valid time duration expression (see below) and defines the start time of the action. The start time is not set absolutely but is counted from the beginning of the startup of the Bridge process.	0 (means directly after start-up of the xUML service)	
parallelEvents	must be an integer and defines the count of Timer event handlers running in parallel. An event handler runs in parallel when an event handler needs more time for execution than the period defined by the tagged value **repeatInterval**. In this case, the next event handler is due to start before the prior one has finished. The Timer starts the pending event handler in parallel with the one currently running. This happens as long as the number of parallel running handlers is less than, or equal to, **parallelEvents**. If the number of allowed parallel events is reached, the next event will be queued.	1 (means that Timer event handlers do not run in parallel)	
queuedEvents	must be an integer and defines the count of timer events that are stored in a queue. The Bridge queues timer events if the count of running parallel events is higher than **parallelEvents**. These events are not executed immediately but wait until the number of running parallel events is lower than, or equal to, its definition in **parallelEvents**.	1 (means that Timer event handlers that cannot run in parallel will be queued)	

Running the xUML service, the following warning may be written to the xUML service standard log that can be viewed on the Bridge:

```
[Warning][External][TIMADLM][13][Event dropped, queue is full. Request
not being executed: <internal activity diagram ID>]
```

Reason and solution for this problem are described in chapter Troubleshooting.