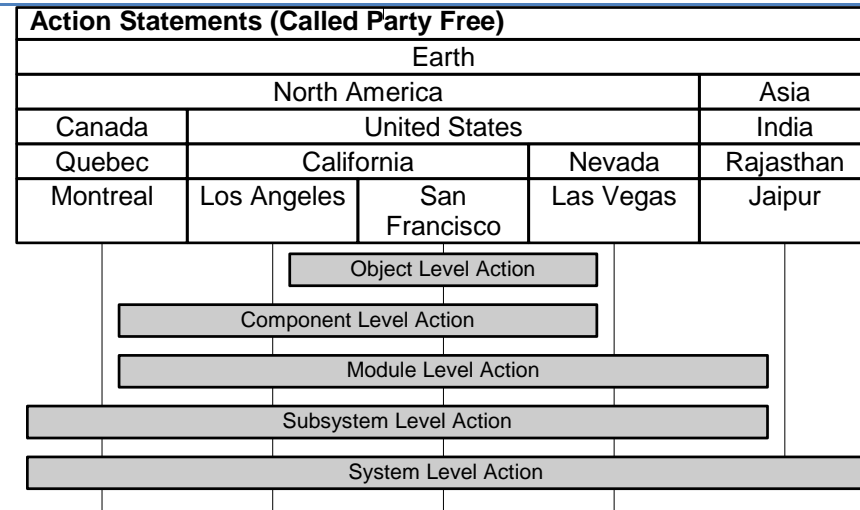


VISUAL GUIDE TO FDL

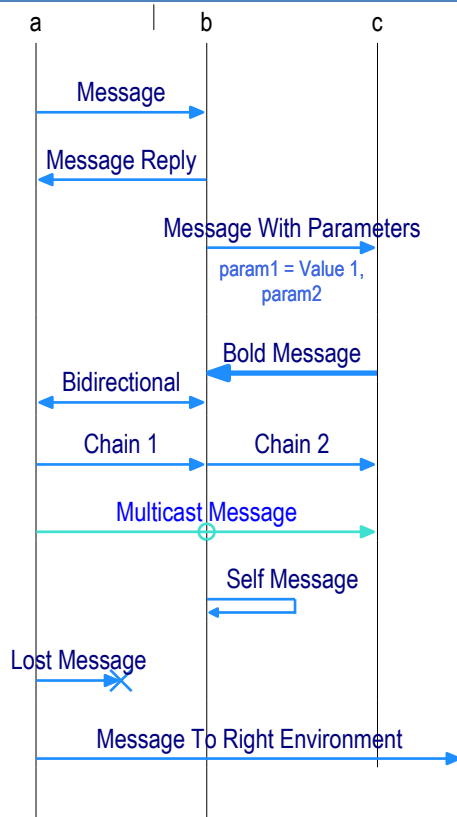
This section provides a visual introduction to FDL. Sequence diagrams and the associated FDL are shown.

Architecture Definition



```
#include "inc.fdl"  
system: Earth  
subsystem : "North America" in Earth, Asia in Earth  
module: Canada in "North America", "United States" in "North America"  
module: India in Asia  
component: California in "United States", Nevada in "United States"  
component: Quebec in Canada, Rajasthan in India  
eternal: "Los Angeles" in California, "San Francisco" in California  
eternal: "Las Vegas" in Nevada, Montreal in Quebec, Jaipur in Rajasthan  
feature "Action Statements"  
    "San Francisco" takes action "Object Level Action"  
    California takes action "Component Level Action"  
    "United States" takes action "Module Level Action"  
    "North America" takes action "Subsystem Level Action"  
    Earth takes action "System Level Action"  
endfeature
```

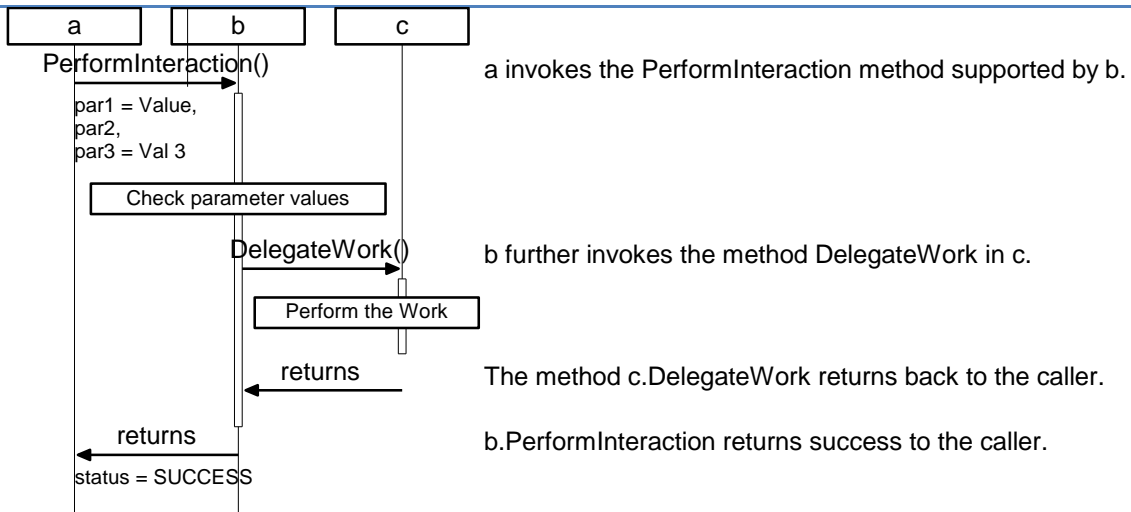
Messages



```

module: Module_01
component: Component_01 in Module_01
eternal: a in Component_01, b in Component_01, c in Component_01
feature "Message Statements"
  Message : a -> b          (* A simple message exchange *)
  "Message Reply": a <- b    (* Reply to the message. *)
  "Message With Parameters" (param1="Value 1", param2) : b -> c
  (* Parameters can be specified with messages. *)
  "Bold Message" : b <= c    (* Message shown with a thick arrow. *)
  "Bidirectional" : a <-> b  (* Model bi-directional interactions. *)
  chain
    "Chain 1" : a -> b
    "Chain 2" : b -> c
  endchain (* Messages can be chained to appear in a single line. *)
  b multicasts "Multicast Message" to a, c (* Model multicasts. *)
  "Self Message" : b -> b    (* A message sent to self. *)
  "Lost Message" : a ->X b   (* Model a lost message. *)
  "Message To Right Environment": a -> env_r
  (* External entities can be modeled as environment. *)
endfeature
  
```

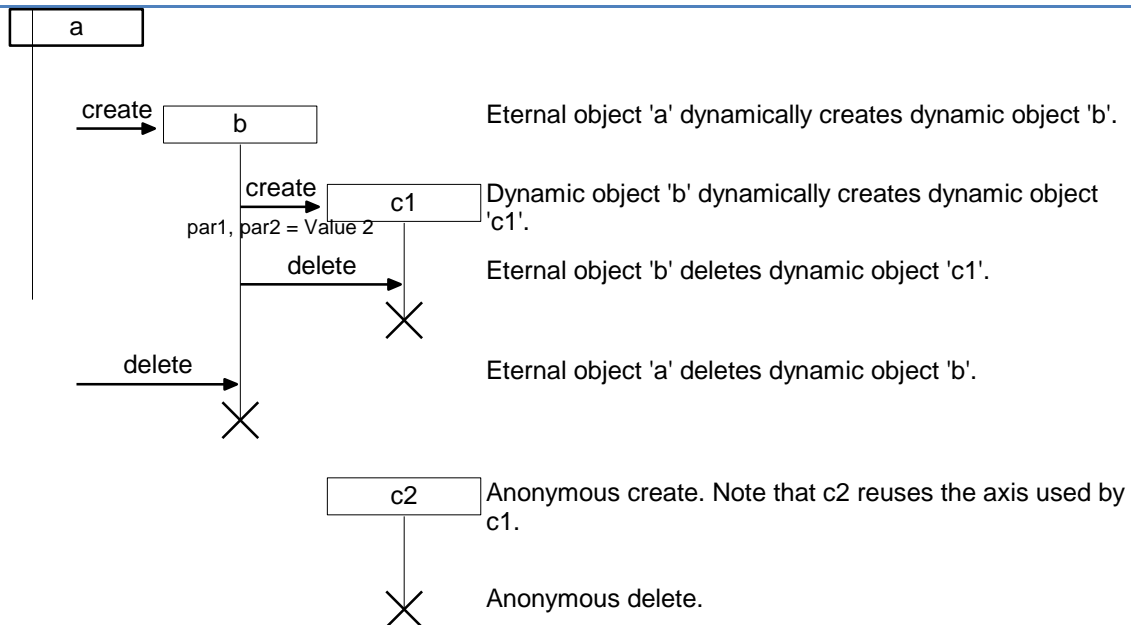
Object Interactions



```

module: Module_01
component: Component_01 in Module_01
eternal: a in Component_01, b in Component_01, c in Component_01
feature "Object Interactions"
  a invokes b.PerformInteraction(par1 = Value, par2, par3 = "Val 3")
  (* a invokes the PerformInteraction method supported by b. *)
  b takes action "Check parameter values"
  b invokes c.DelegateWork
  (* b further invokes the method DelegateWork in c. *)
  c takes action "Perform the Work"
  c.DelegateWork returns
  (* The method c.DelegateWork returns back to the caller. *)
  b.PerformInteraction returns (status = SUCCESS)
  (* b.PerformInteraction returns success to the caller. *)
endfeature
  
```

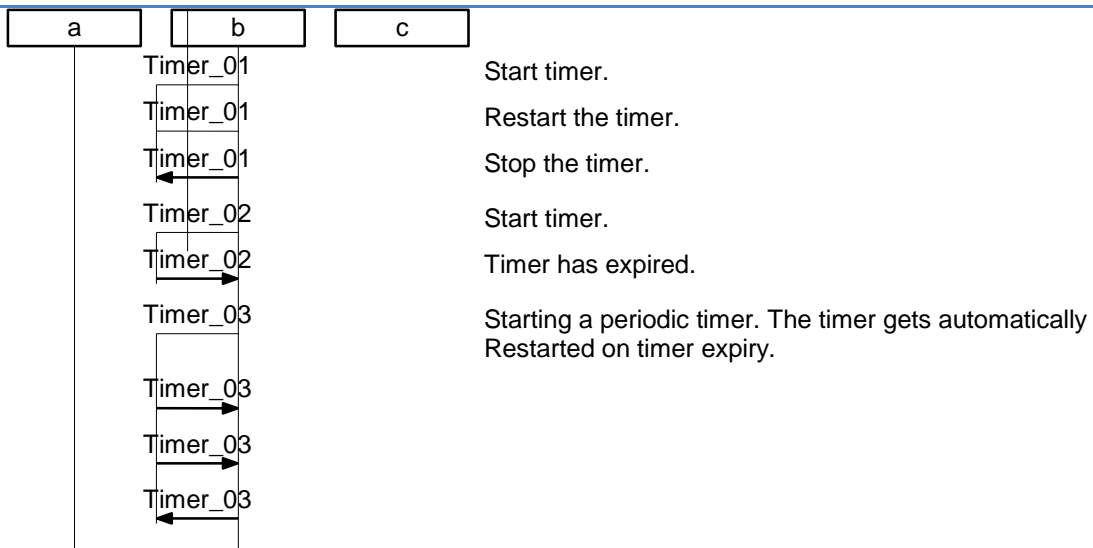
Object Creation and Deletion



```

module: Module_01
component: Component_01 in Module_01
eternal: a in Component_01
dynamic: b in Component_01, c1 | c2 in Component_01
/* Dynamic object c1 and c2 share an axis. Only one of them
can be active at a given time */
feature "Object Creation and Deletion"
  a creates b
  (* Eternal object 'a' dynamically creates dynamic object 'b'. *)
  b creates c1(par1, par2="Value 2")
  (* Dynamic object 'b' dynamically creates dynamic object 'c1'. *)
  b deletes c1
  (* Dynamic object 'b' deletes dynamic object 'c1'. *)
  a deletes b
  (* Eternal object 'a' deletes dynamic object 'b'. *)
  create c2
  (* Anonymous create. Note that c2 reuses the axis used by c1. *)
  delete c2  (* Anonymous delete. *)
endfeature
  
```

Timers



```

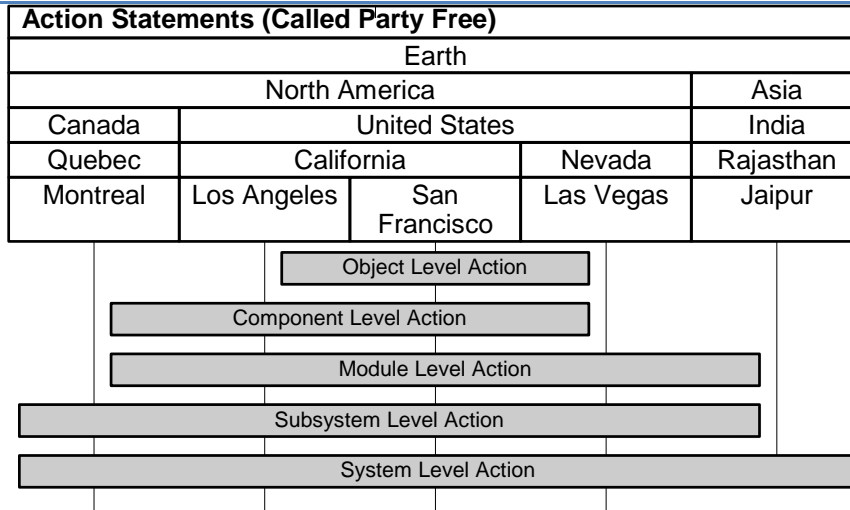
module: Module_01
component: Component_01 in Module_01
eternal: a in Component_01, b in Component_01, c in Component_01
feature "Timer Statements"
  b starts Timer_01 (* Start timer. *)
  b restarts Timer_01 (* Restart the timer. *)
  b stops Timer_01 (* Stop the timer. *)

  b starts Timer_02 (* Start timer. *)
  timeout Timer_02 (* Timer has expired. *)

  b starts periodic Timer_03
  (* Starting a periodic timer. The timer gets automatically
  Restarted on timer expiry. *)
  timeout Timer_03
  timeout Timer_03
  b stops Timer_03
endfeature
  
```

Actions

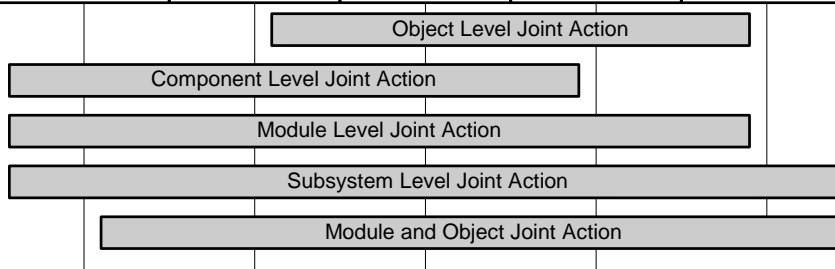
Single Action



```
#include "inc.fdl"
system: Earth
subsystem : "North America" in Earth, Asia in Earth
module: Canada in "North America", "United States" in "North America"
module: India in Asia
component: California in "United States", Nevada in "United States"
component: Quebec in Canada, Rajasthan in India
eternal: "Los Angeles" in California, "San Francisco" in California
eternal: "Las Vegas" in Nevada, Montreal in Quebec, Jaipur in Rajasthan
feature "Action Statements"
    "San Francisco" takes action "Object Level Action"
    California takes action "Component Level Action"
    "United States" takes action "Module Level Action"
    "North America" takes action "Subsystem Level Action"
    Earth takes action "System Level Action"
endfeature
```

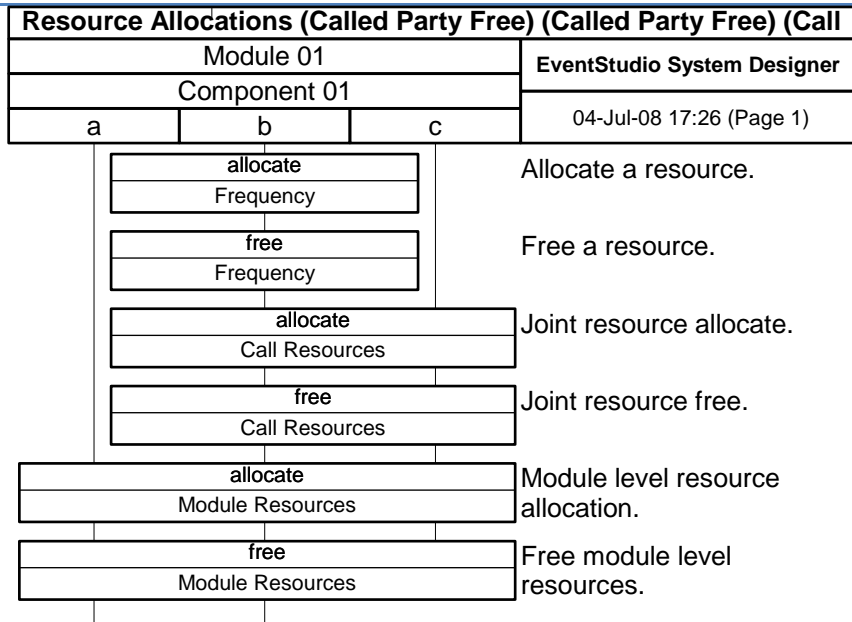
Multiple Action

Action Statements (Called Party Free)				
Earth				
North America				Asia
Canada	United States			India
Quebec	California	Nevada	Rajasthan	
Montreal	Los Angeles	San Francisco	Las Vegas	Jaipur



```
#include "inc.fdl"
system: Earth
subsystem : "North America" in Earth, Asia in Earth
module: Canada in "North America", "United States" in "North America"
module: India in Asia
component: California in "United States", Nevada in "United States"
component: Quebec in Canada, Rajasthan in India
eternal: "Los Angeles" in California, "San Francisco" in California
eternal: "Las Vegas" in Nevada, Montreal in Quebec, Jaipur in Rajasthan
feature "Action Statements"
    "San Francisco", "Las Vegas" take action "Object Level Joint Action"
    California, Quebec take action "Component Level Joint Action"
    "United States", Canada take action "Module Level Joint Action"
    "North America", Asia take action "Subsystem Level Joint Action"
    "United States", Jaipur take action "Module and Object Joint Action"
endfeature
```

Resource Allocations



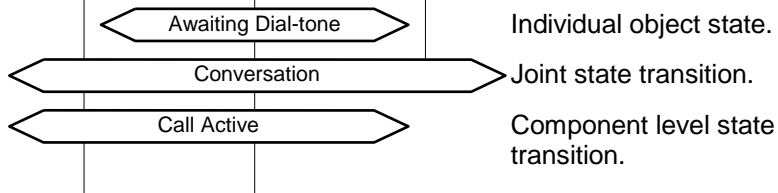
```

module: Module_01
component: Component_01 in Module_01
eternal: a in Component_01, b in Component_01, c in Component_01
feature "Resource Allocations"
  b allocates "Frequency"      (* Allocate a resource. *)
  b frees "Frequency"         (* Free a resource. *)
  b, c allocate "Call Resources" (* Joint resource allocate. *)
  b, c free "Call Resources"   (* Joint resource free. *)
  Module_01 allocates "Module Resources"
  (* Module level resource allocation. *)
  Module_01 frees "Module Resources"
  (* Free module level resources. *)
endfeature

```

State Transitions

Resource Allocations (Called Party Free) (Called Party Free) (Call			
Module 01		Module 02	EventStudio System Designer
Component 01		Component 02	
a	b	c	04-Jul-08 12:48 (Page 1)

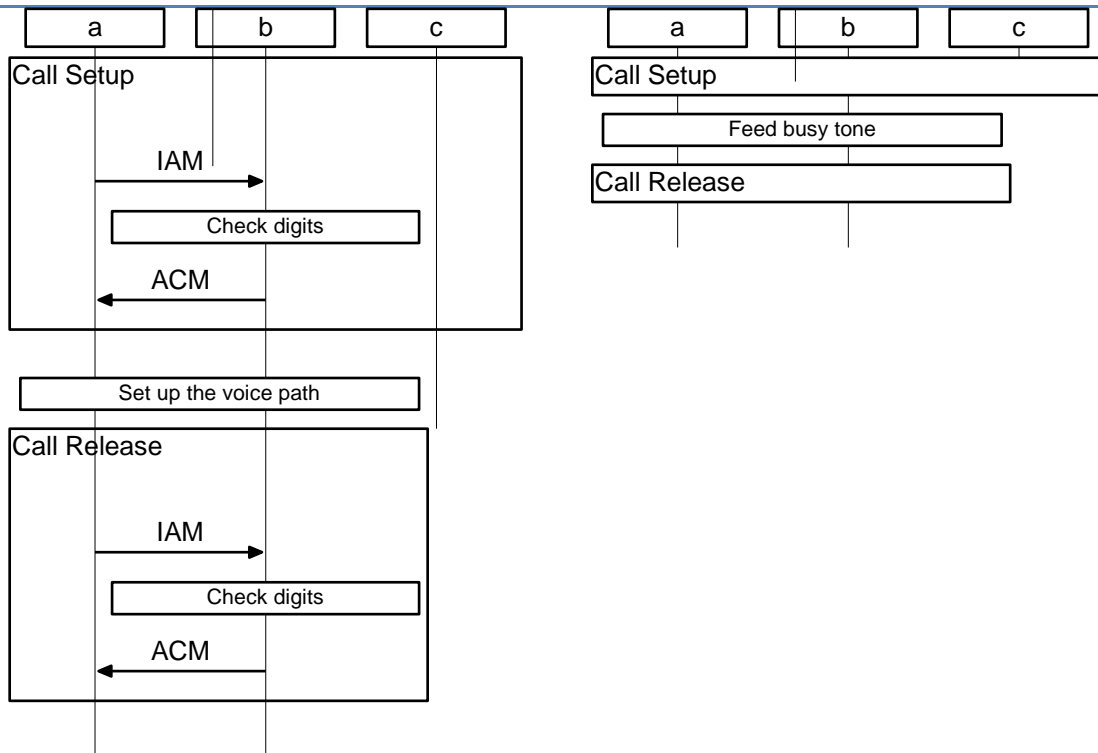


```

module: Module_01, Module_02
component: Component_01 in Module_01, Component_02 in Module_02
eternal: a in Component_01, b in Component_01, c in Component_02
feature "Resource Allocations"
  b state = "Awaiting Dial-tone"           (* Individual object state. *)
  a,b,c state = "Conversation"           (* Joint state transition. *)
  Component_01 state = "Call Active"
  (* Component level state transition. *)
endfeature

```

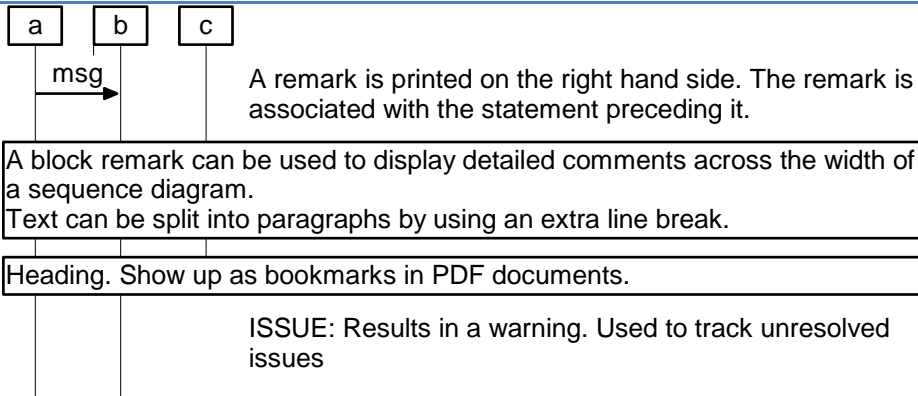
Sequence Groupings



```

module: Module_01
component: Component_01 in Module_01
eternal: a in Component_01, b in Component_01, c in Component_01
feature "Sequence Grouping"
  sequence "Call Setup"
    IAM : a -> b
    b takes action "Check digits"
    ACM : a <- b
  endsequence
  case
    leg "Called Party Free":
      a, b take action "Set up the voice path"
    leg "Called Party Busy":
      a, b take action "Feed busy tone"
  endcase
  a, b participate in sequence "Call Release"
  REL : a -> b
  RLC : a <- b
endsequence
endfeature
  
```

Remarks



```

module: Module_01
component: Component_01 in Module_01
eternal: a in Component_01, b in Component_01, c in Component_01
feature "Sequence Grouping"
  msg : a -> b (* A remark is printed on the right hand side.
  The remark is associated with the statement preceding it. *)

  [* A block remark can be used to display detailed comments across
  the width of a sequence diagram.
  Text can be split into paragraphs by using an extra line break.*]

  /* C-style comment, ignored in document generation. */

  heading "Heading. Show up as bookmarks in PDF documents."

  issue "Results in a warning. Used to track unresolved issues"
endfeature

```