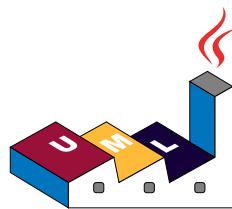


Diagramando UML con PlantUML



Guía de Referencia del Lenguaje PlantUML

(Version 1.2023.11)

PlantUML es un proyecto Open Source (código abierto) que permite escribir rápidamente:

- Diagramas de Secuencia
- Diagramas de Casos de uso
- Diagramas de Clases
- Diagramas de Objetos
- Diagramas de Actividades
- Diagramas de Componentes
- Diagramas de Despliegue
- Diagramas de Estados
- Timing diagram

Los siguientes diagramas no-UML también están soportados:

- JSON Data
- YAML Data
- Network diagram (nwdiag)
- Wireframe graphical interface
- Archimate diagram
- Specification and Description Language (SDL)
- Ditaa diagram
- Diagrama de Gantt
- MindMap diagram
- Work Breakdown Structure diagram
- Mathematic with AsciiMath or JLaTeXMath notation
- Entity Relationship diagram

Los diagramas son definidos usando un lenguaje simple e intuitivo.

1 Diagrama de secuencia

Crear diagramas de secuencia con PlantUML es notablemente sencillo. Esta facilidad de uso se atribuye en gran medida a la naturaleza amigable de su sintaxis, diseñada para ser intuitiva y fácil de recordar.

- **Sintaxis intuitiva:**

En primer lugar, los usuarios aprecian la sintaxis sencilla e intuitiva de PlantUML. Este diseño bien pensado significa que incluso aquellos que son nuevos en la creación de diagramas encuentran fácil comprender los conceptos básicos de forma rápida y sin complicaciones.

- **Correlación texto-gráfico:**

Otra característica distintiva es la estrecha semejanza entre la representación textual y el resultado gráfico. Esta correlación armoniosa garantiza que los borradores textuales se traduzcan con bastante precisión en diagramas gráficos, proporcionando una experiencia de diseño cohesiva y predecible sin sorpresas desagradables en el resultado final.

- **Proceso de elaboración eficiente:**

La fuerte correlación entre el texto y el resultado gráfico no sólo simplifica el proceso de creación, sino que también lo acelera significativamente. Los usuarios se benefician de un proceso más ágil con menos necesidades de revisiones y ajustes que requieren mucho tiempo.

- **Visualización durante la redacción:**

La posibilidad de visualizar el resultado gráfico final mientras se redacta el texto es una función que muchos consideran inestimable. Fomenta de forma natural una transición fluida del borrador inicial a la presentación final, mejorando la productividad y reduciendo la probabilidad de errores.

- **Facilidad de edición y revisión:**

Es importante destacar que la edición de los diagramas existentes es un proceso sin complicaciones. Dado que los diagramas se generan a partir de texto, los usuarios descubren que realizar ajustes es considerablemente más fácil y preciso que alterar una imagen utilizando herramientas gráficas. Se reduce simplemente a modificar el texto, un proceso mucho más sencillo y menos propenso a errores que realizar cambios a través de una interfaz gráfica con un ratón.

PlantUML facilita un enfoque sencillo y fácil de usar para crear y editar diagramas de secuencia, satisfaciendo las necesidades tanto de principiantes como de diseñadores experimentados. Aprovecha hábilmente la simplicidad de las entradas textuales para elaborar diagramas visualmente descriptivos y precisos, estableciéndose así como una herramienta imprescindible en el conjunto de herramientas de creación de diagramas.

Puede obtener más información sobre algunos de los comandos comunes en Plant UML para mejorar su experiencia de creación de diagramas.

1.1 Ejemplo básico

La secuencia `->` es usada para dibujar un mensaje entre dos participantes. Los participantes tienen que ser declarados explícitamente.

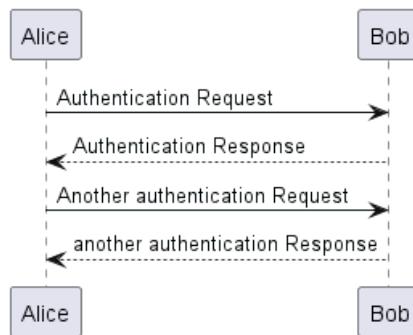
Para definir una flecha punteada, se debe usar `-->`

También se puede usar `<-` y `<--`. No provoca cambios en el dibujo, pero puede mejorar la legibilidad. Tenga en cuenta que esto sólo es posible en diagramas de secuencia, las reglas son diferentes para otros diagramas.

```
@startuml
Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response
```

```
Alice -> Bob: Another authentication Request
Alice <-- Bob: another authentication Response
@enduml
```





1.2 Declarando participantes

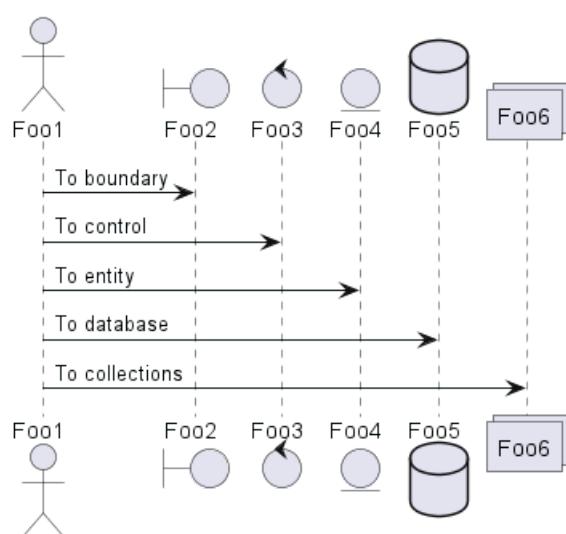
Es posible cambiar el orden de los participantes usando la palabra reservada `participant`.

También es posible el uso de otras palabras reservadas para declarar un participante:

- `actor`
- `boundary`
- `control`
- `entity`
- `database`
- `collections`

```
@startuml
actor Foo1
boundary Foo2
control Foo3
entity Foo4
database Foo5
collections Foo6
Foo1 -> Foo2 : To boundary
Foo1 -> Foo3 : To control
Foo1 -> Foo4 : To entity
Foo1 -> Foo5 : To database
Foo1 -> Foo6 : To collections
```

```
@enduml
```



Se puede renombrar un participante usando la palabra reservada `as`.



También es posible cambiar el color de fondo de los actores o participantes.

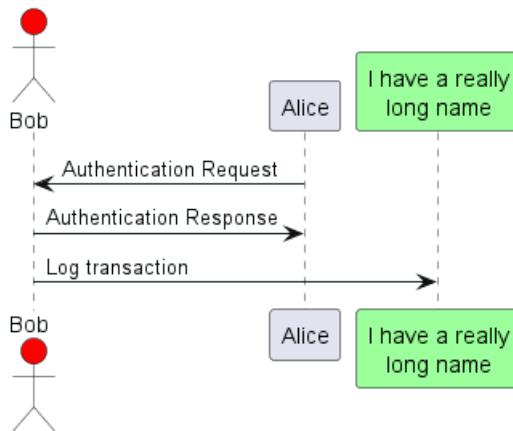
```
@startuml
actor Bob #red
' The only difference between actor
and participant is the drawing
participant Alice
participant "I have a really\nlong name" as L #99FF99
/' You can also declare:
    participant L as "I have a really\nlong name" #99FF99
'/
```

Alice->Bob: Authentication Request

Bob->Alice: Authentication Response

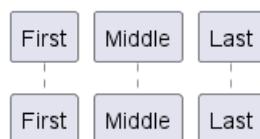
Bob->L: Log transaction

@enduml



You can use the `order` keyword to custom the print order of participant.

```
@startuml
participant Last order 30
participant Middle order 20
participant First order 10
@enduml
```



1.3 Declaring participant on multiline

Puedes declarar el participante en múltiples líneas.

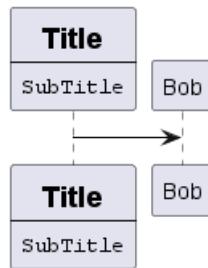
```
@startuml
participant Participant [
    =Title
    -----
    """SubTitle"""
]

participant Bob
```

Participant -> Bob

@enduml





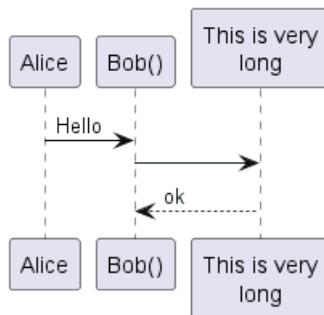
[Ref. QA-15232]

1.4 Sin usar letras en participantes

Puedes usar comillas para definir participantes. Y puedes usar la palabra reservada `as` para asignar un alias a esos participantes.

```

@startuml
Alice -> "Bob()" : Hello
"Bob()" -> "This is very\nlong" as Long
' You can also declare:
' "Bob()" -> Long as "This is very\nlong"
Long --> "Bob()" : ok
@enduml
  
```



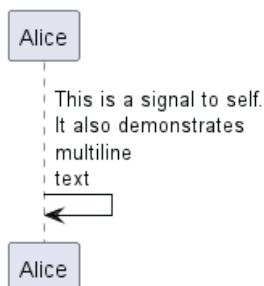
1.5 Mensaje a sí mismo

Un participante puede enviarse un mensaje a sí mismo.

También es posible tener varias líneas utilizando .

```

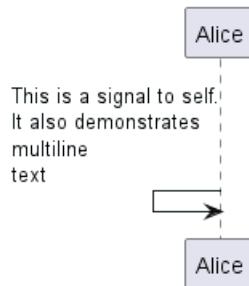
@startuml
Alice -> Alice: This is a signal to self.\nIt also demonstrates\nmultiline \ntext
@enduml
  
```



```

@startuml
Alice <- Alice: This is a signal to self.\nIt also demonstrates\nmultiline \ntext
@enduml
  
```





[Ref. QA-1361]

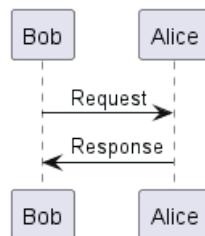
1.6 Text alignment

Text alignment on arrows can be set to left, right or center using `skinparam sequenceMessageAlign`.

You can also use `direction` or `reverseDirection` to align text depending on arrow direction. Further details and examples of this are available on the [skinparam](#) page.

```

@startuml
skinparam sequenceMessageAlign right
Bob -> Alice : Request
Alice -> Bob : Response
@enduml
  
```

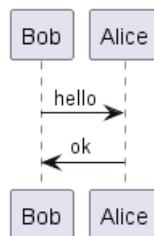


1.6.1 Text of response message below the arrow

You can put the text of the response message below the arrow, with the `skinparam responseMessageBelowArrow true` command.

```

@startuml
skinparam responseMessageBelowArrow true
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
  
```



1.7 Cambiar estilo de la flecha

Puede cambiar el estilo de la flecha de diferentes formas:

- añade una `x` al final para indicar un mensaje perdido
- utilice `\` o `/` en lugar de `<` o `>` para tener solo la parte inferior o superior de la flecha
- repite la cabeza de la flecha (por ejemplo, `>>` o `//`) para tener un trazo más fino

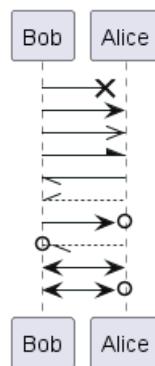


- Utilice -- en lugar de - para obtener una flecha punteada.
- añade una "o" al final de la cabeza de una flecha
- utilice flechas bidireccionales <->

```
@startuml
Bob ->x Alice
Bob -> Alice
Bob ->> Alice
Bob -\ Alice
Bob \\\- Alice
Bob //-- Alice

Bob ->o Alice
Bob o\\-- Alice

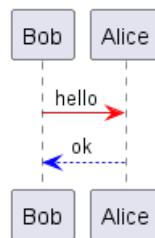
Bob <-> Alice
Bob <->o Alice
@enduml
```



1.8 Cambiar el color de la flecha

Puede cambiar el color de flechas individuales usando la siguiente notación:

```
@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml
```



1.9 Numeración de la secuencia de mensajes

La palabra clave `autonumber` es usada para añadir automáticamente números a los mensajes.

```
@startuml
autonumber
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response
@enduml
```





Puedes especificar un número de comienzo con `autonumber //número inicial//`, y también un incremento con `autonumber //número inicial// //incremento//`.

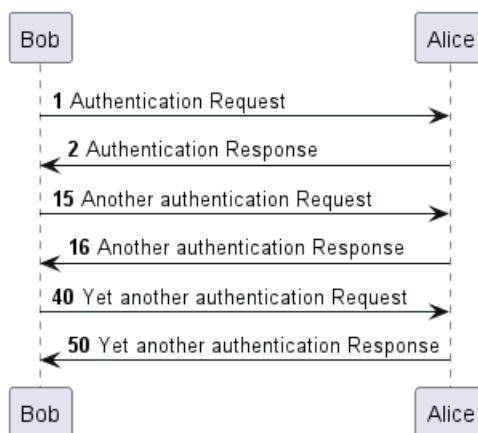
```

@startuml
autonumber
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber 15
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response
  
```

`@enduml`



Puedes especificar un formato para su número usándolo entre comillas dobles.

El formateo se hace mediante la clase Java `DecimalFormat` (0 denota un dígito, # denota un dígito y cero si está ausente).

Puedes usar alguna etiqueta HTML en el formato.

```

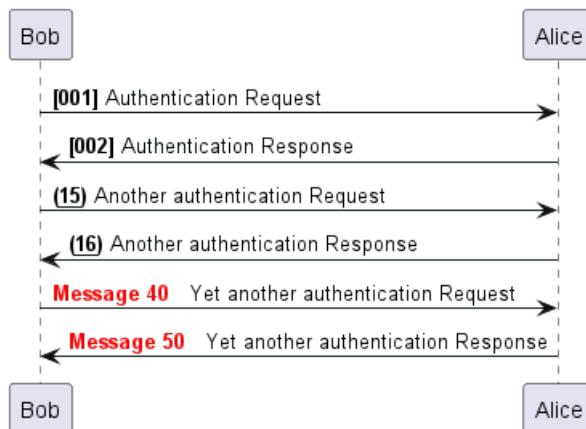
@startuml
autonumber "<b>[000]</b>"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber 15 "<b>(<u>##</u>)""
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10 "<font color=red><b>Message 0 </b></font>"
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response
  
```



@enduml



También puedes usar `autonumber stop` y `autonumber resume //increment// //format//` para pausar y continuar la numeración automática, respectivamente.

```
@startuml
autonumber 10 10 "<b>[000]"
```

```
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response
```

```
autonumber stop
Bob -> Alice : dummy
```

```
autonumber resume "<font color=red><b>Message 0  ">
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response
```

```
autonumber stop
Bob -> Alice : dummy
```

```
autonumber resume 1 "<font color=blue><b>Message 0  "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response
@enduml
```



[Ref. [QA-7119](<https://forum.plantuml.net/7119/create-links-after-creating-a-diagram?show=7137#a7137>)]



1.10 Page Title, Header and Footer

The `title` keyword is used to add a title to the page.

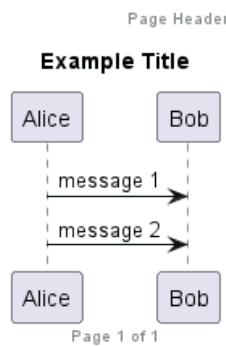
Pages can display headers and footers using `header` and `footer`.

```
@startuml
```

```
header Page Header
footer Page %page% of %lastpage%
title Example Title
```

```
Alice -> Bob : message 1
Alice -> Bob : message 2
```

```
@enduml
```



1.11 Dividiendo diagramas

The reserved word `newpage` is used to divide a diagram into several images.

You can place a title for the new page just after the reserved word `newpage`.

This is quite practical with *Word* to return large diagrams in several pages.

```
@startuml
```

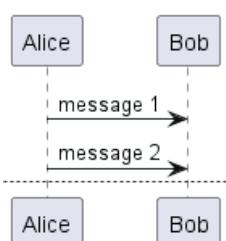
```
Alice -> Bob : message 1
Alice -> Bob : message 2
```

```
newpage
```

```
Alice -> Bob : message 3
Alice -> Bob : message 4
```

```
newpage A title for the\last page
```

```
Alice -> Bob : message 5
Alice -> Bob : message 6
@enduml
```



1.12 Agrupando mensajes

Es posible agrupar mensajes usando las siguientes palabras reservadas:

- alt/else
- opt
- loop
- par
- break
- critical
- group, seguida de un texto para mostrar

Es posible añadir un texto que será mostrado en el encabezado (excepto para group).

La palabra reservada end es usada para cerrar el grupo.

Tenga en cuenta que es posible anidar grupos.

```
@startuml  
Alice -> Bob: Authentication Request
```

```
alt successful case
```

```
    Bob -> Alice: Authentication Accepted
```

```
else some kind of failure
```

```
    Bob -> Alice: Authentication Failure
```

```
    group My own label
```

```
        Alice -> Log : Log attack start
```

```
        loop 1000 times
```

```
            Alice -> Bob: DNS Attack
```

```
        end
```

```
        Alice -> Log : Log attack end
```

```
    end
```

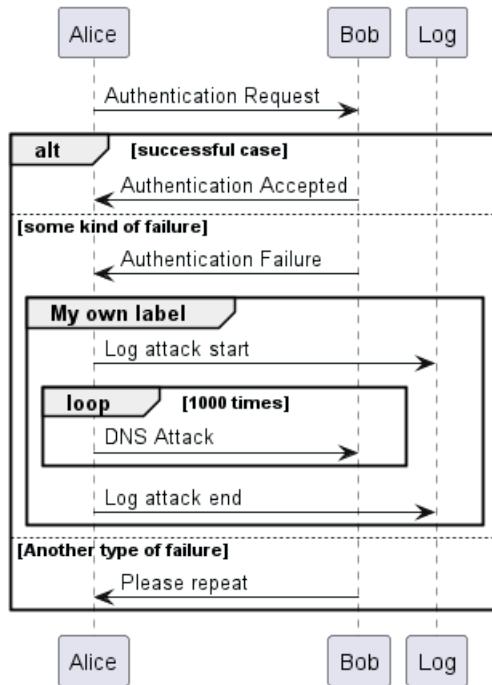
```
else Another type of failure
```

```
    Bob -> Alice: Please repeat
```

```
end
```

```
@enduml
```





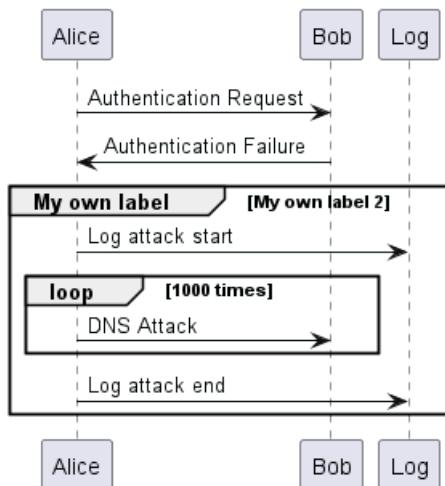
1.13 Secondary group label

For group, it is possible to add, between [and], a secondary text or label that will be displayed into the header.

```

@startuml
Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Failure
group My own label [My own label 2]
    Alice -> Log : Log attack start
    loop 1000 times
        Alice -> Bob: DNS Attack
    end
    Alice -> Log : Log attack end
end
@enduml

```



[Ref. QA-2503]



1.14 Notas en mensajes

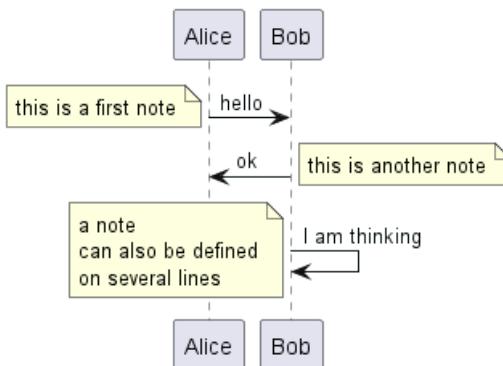
Es posible colocar notas en mensajes usando las palabras reservadas `note left` o `note right` *inmediatamente después del mensaje*.

Puedes tener una nota multi-líneas usando la palabra reservada `end note`.

```
@startuml
Alice->Bob : hello
note left: this is a first note

Bob->Alice : ok
note right: this is another note

Bob->Bob : I am thinking
note left
a note
can also be defined
on several lines
end note
@enduml
```



1.15 Algunas otras notas

También es posible colocar notas relativas al participante con las palabras reservadas `<code>note left of</code>` , `note right of` o `note over` .

Es posible resaltar una nota cambiando su color de fondo.

También puedes tener una nota multi-líneas usando la palabra reservada `end note`.

```
@startuml
participant Alice
participant Bob
note left of Alice #aqua
This is displayed
left of Alice.
end note

note right of Alice: This is displayed right of Alice.

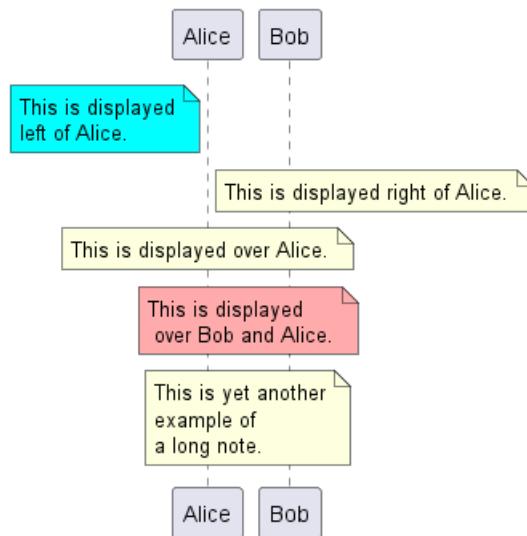
note over Alice: This is displayed over Alice.

note over Alice, Bob #FFAAAA: This is displayed\n over Bob and Alice.

note over Bob, Alice
This is yet another
example of
a long note.
```



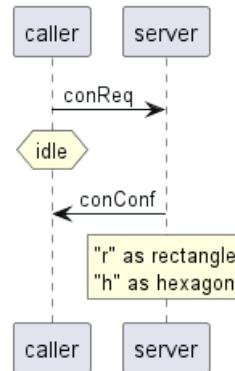
```
end note
@enduml
```



1.16 Cambiando el aspecto de las notas

Puedes usar las palabras reservadas `hnote` y `rnote` para cambiar el aspecto de las notas.

```
@startuml
caller -> server : conReq
hnote over caller : idle
caller <- server : conConf
rnote over server
    "r" as rectangle
    "h" as hexagon
endrnote
@enduml
```



[Ref. [QA-1765](<https://forum.plantuml.net/1765/is-it-possible-to-have-different-shapes-for-notes?show=1806#c1806>)]

1.17 Note over all participants [across]

You can directly make a note over all participants, with the syntax:

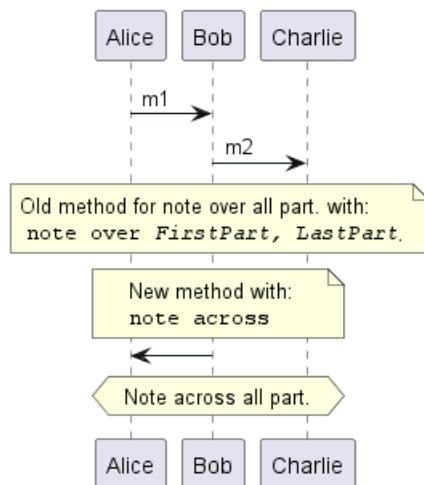
- `note across: note_description`

```
@startuml
Alice->Bob:m1
Bob->Charlie:m2
note over Alice, Charlie: Old method for note over all part. with:\n ""note over //FirstPart, LastPa
note across: New method with:\n""note across""
```



```
Bob->Alice
```

```
hnote across:Note across all part.  
@enduml
```



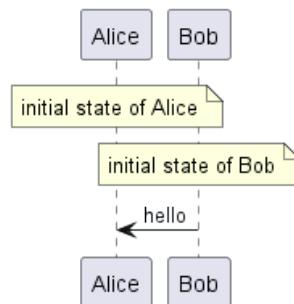
[Ref. QA-9738]

1.18 Several notes aligned at the same level [/]

You can make several notes aligned at the same level, with the syntax /:

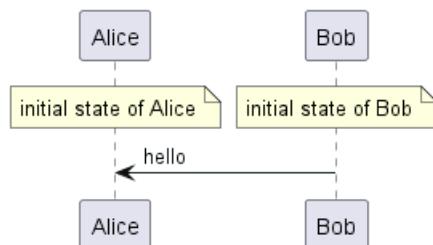
- without / (*by default, the notes are not aligned*)

```
@startuml
note over Alice : initial state of Alice
note over Bob : initial state of Bob
Bob -> Alice : hello
@enduml
```



- with / (*the notes are aligned*)

```
@startuml
note over Alice : initial state of Alice
/ note over Bob : initial state of Bob
Bob -> Alice : hello
@enduml
```



[Ref. QA-354]

1.19 Creole y HTML

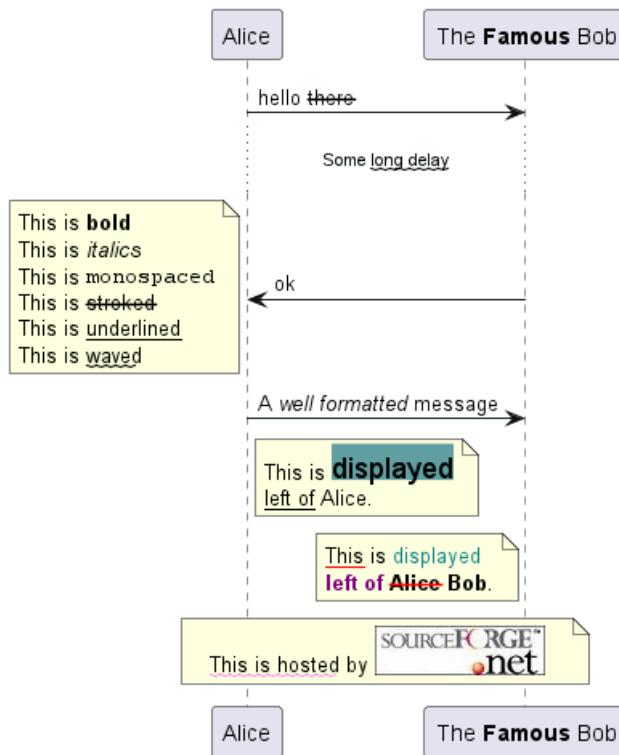
También es posible usar sintaxis de WikiCreole:

```
@startuml
participant Alice
participant "The **Famous** Bob" as Bob

Alice -> Bob : hello --there--
... Some ~~long delay~~ ...
Bob -> Alice : ok
note left
    This is **bold**
    This is //italics//
    This is ""monospaced"""
    This is --stroked--
    This is __underlined__
    This is ~~waved~~
end note

Alice -> Bob : A //well formatted// message
note right of Alice
    This is <back:cadetblue><size:18>displayed</size></back>
    __left of__ Alice.
end note
note left of Bob
    <u:red>This</u> is <color #118888>displayed</color>
    **<color purple>left of</color> <s:red>Alice</strike> Bob**.
end note
note over Alice, Bob
    <w:#FF33FF>This is hosted</w> by <img sourceforge.jpg>
end note
@enduml
```





1.20 Divisor

Siquieres, puedes dividir un diagrama usando el separador == para separar su diagrama en pasos lógicos.

@startuml

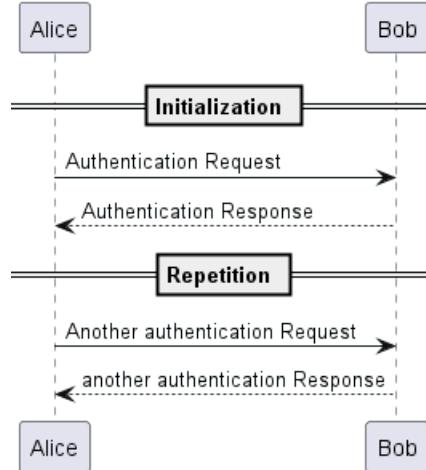
== Initialization ==

Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response

== Repetition ==

Alice -> Bob: Another authentication Request
Alice <-- Bob: another authentication Response

@enduml



1.21 Referencia

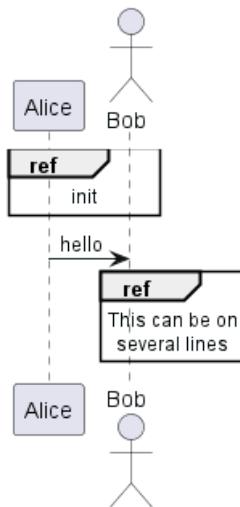
Puedes referenciar en un diagrama utilizando la palabra clave `ref over`.

```
@startuml
participant Alice
actor Bob

ref over Alice, Bob : init

Alice -> Bob : hello

ref over Bob
  This can be on
  several lines
end ref
@enduml
```



1.22 Retardo

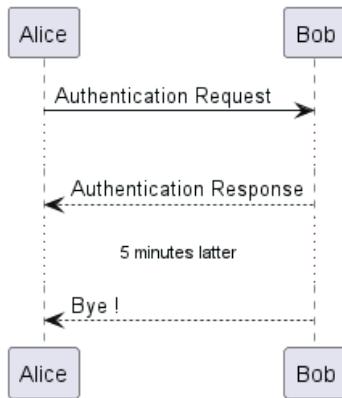
Puedes usar `...` para indicar un retardo en el diagrama. Y también es posible colocar un mensaje con ese retardo.

```
@startuml

Alice -> Bob: Authentication Request
...
Bob --> Alice: Authentication Response
...5 minutes latter...
Bob --> Alice: Bye !

@enduml
```





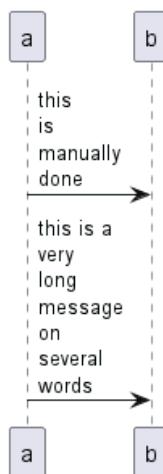
1.23 Text wrapping

To break long messages, you can manually add `\n` in your text.

Another option is to use `maxMessageSize` setting:

```

@startuml
skinparam maxMessageSize 50
participant a
participant b
a -> b :this\nis\nmanually\ndone
a -> b :this is a very long message on several words
@enduml
  
```



1.24 Espaciado

Puedes usar `|||` para indicar espaciado en el diagrama.

También es posible especificar un número de píxel para ser usado.

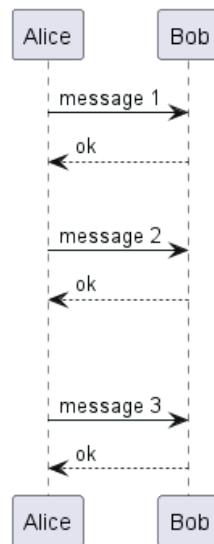
```
@startuml
```

```

Alice -> Bob: message 1
Bob --> Alice: ok
|||
Alice -> Bob: message 2
Bob --> Alice: ok
||45||
Alice -> Bob: message 3
Bob --> Alice: ok
  
```



@enduml



1.25 Activación y Destrucción de la Línea de vida

`activate` y `deactivate` son usados para denotar la activación de un participante.

Una vez que un participante es activado, su línea de vida aparece.

`activate` y `deactivate` aplica en el mensaje anterior.

`destroy` denota el final de la línea de vida de un participante.

```
@startuml
participant User
```

```
User -> A: DoWork
activate A
```

```
A -> B: << createRequest >>
activate B
```

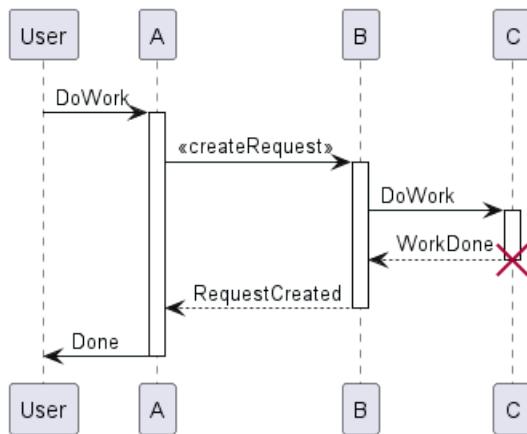
```
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C
```

```
B --> A: RequestCreated
deactivate B
```

```
A -> User: Done
deactivate A
```

```
@enduml
```





Puede usarse anidamiento de líneas de vida, y es posible agregar un color a dicha línea de vida.

```
@startuml
participant User
```

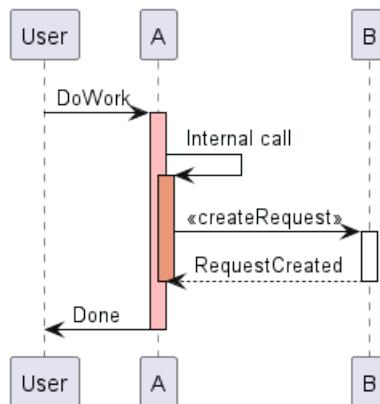
```
User -> A: DoWork
activate A #FFBBBB
```

```
A -> A: Internal call
activate A #DarkSalmon
```

```
A -> B: << createRequest >>
activate B
```

```
B --> A: RequestCreated
deactivate B
deactivate A
A -> User: Done
deactivate A
```

```
@enduml
```



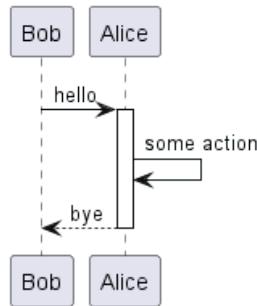
1.26 Return

A new command **return** for generating a return message with optional text label. The point returned to is the point that cause the most recently activated life-line. The syntax is simply **return label** where label, if provided, can be any string acceptable on conventional messages.

```
@startuml
Bob -> Alice : hello
activate Alice
```



```
Alice -> Alice : some action
return bye
@enduml
```



1.27 Creación de participante

Puedes usar la palabra reservada `create` justo antes de la primera recepción de un mensaje para recalcar el hecho de que ese mensaje se encuentra *creando* ese nuevo objeto.

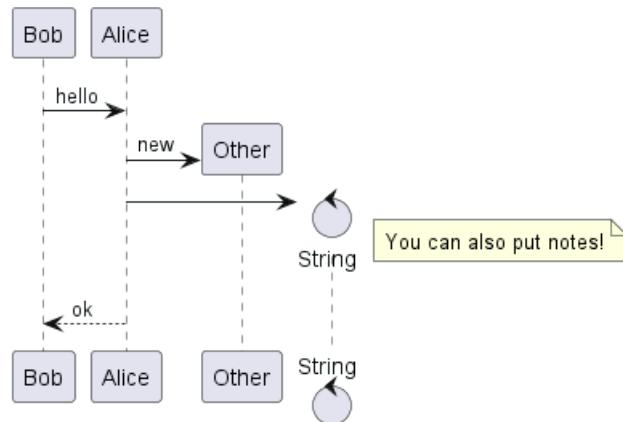
```
@startuml
Bob -> Alice : hello

create Other
Alice -> Other : new

create control String
Alice -> String
note right : You can also put notes!

Alice --> Bob : ok

@enduml
```



1.28 Shortcut syntax for activation, deactivation, creation

Immediately after specifying the target participant, the following syntax can be used:

- `++` Activate the target (optionally a color may follow this)
- `--` Deactivate the source
- `**` Create an instance of the target
- `!!` Destroy an instance of the target

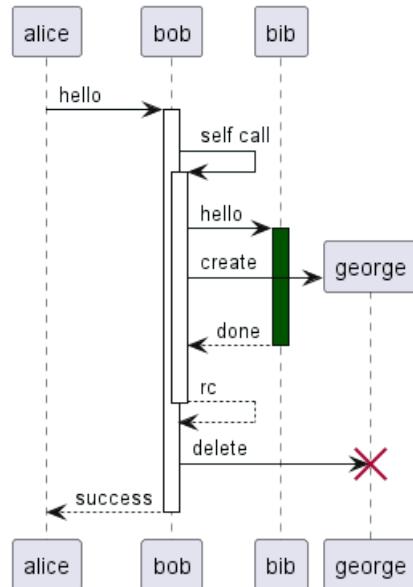
```
@startuml
```



```

alice -> bob ++ : hello
bob -> bob ++ : self call
bob -> bib ++ #005500 : hello
bob -> george ** : create
return done
return rc
bob -> george !! : delete
return success
@enduml

```

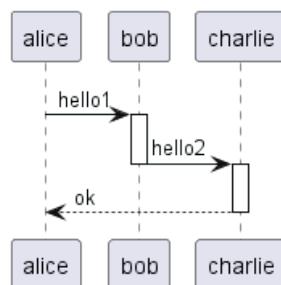


Then you can mix activation and deactivation, on same line:

```

@startuml
alice -> bob ++ : hello1
bob -> charlie ---++ : hello2
charlie --> alice -- : ok
@enduml

```

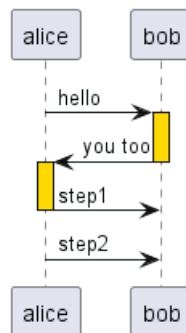


```

@startuml
@startuml
alice -> bob ---++ #gold: hello
bob -> alice ---++ #gold: you too
alice -> bob --: step1
alice -> bob : step2
@enduml
@enduml

```





[Ref. QA-4834, QA-9573 and QA-13234]

1.29 Mensajes entrantes y salientes

Puedes usar flechas entrantes y salientes si quieres centrarte en una parte del diagrama.

Utilice corchetes para denotar el lado izquierdo "[" o el lado derecho "]" del diagrama.

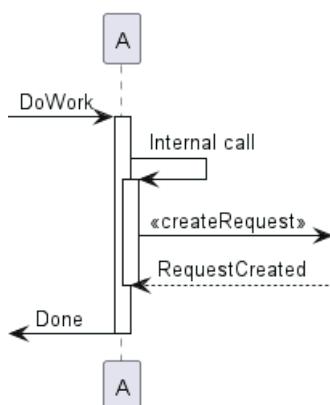
```
@startuml
[-> A: DoWork
```

```
activate A
```

```
A -> A: Internal call
activate A
```

```
A ->] : << createRequest >>
```

```
A<--] : RequestCreated
deactivate A
[<- A: Done
deactivate A
@enduml
```



También puedes tener la siguiente sintaxis:

```
@startuml
[!-> Bob
[o-> Bob
[o->o Bob
[x-> Bob
```

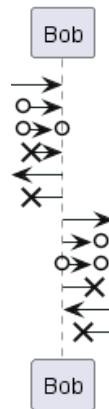
```
[<- Bob
[x<- Bob
```

```
Bob ->]
```



```
Bob ->o]
Bob o->o]
Bob ->x]
```

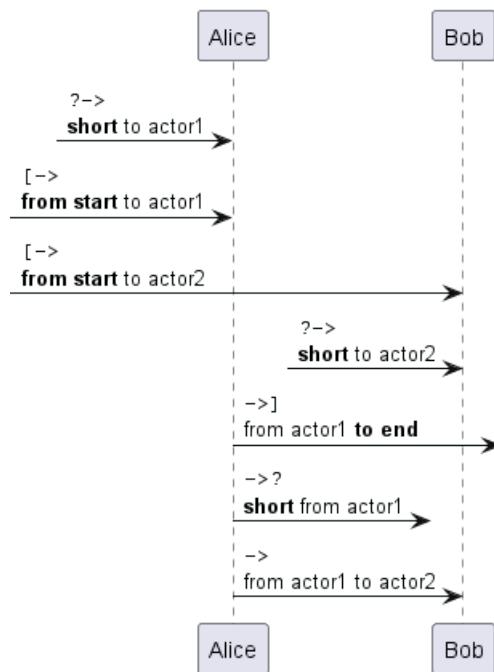
```
Bob <-]
Bob x<-]
@enduml
```



1.30 Short arrows for incoming and outgoing messages

You can have **short** arrows with using ?.

```
@startuml
?-> Alice : """?->"""\n**short** to actor1
[-> Alice : """[->"""\n**from start** to actor1
[-> Bob : """[->"""\n**from start** to actor2
?-> Bob : """?->"""\n**short** to actor2
Alice ->] : """->] """\nfrom actor1 **to end**
Alice ->? : """->?"""\n**short** from actor1
Alice -> Bob : """->"""\nfrom actor1 to actor2
@enduml
```



[Ref. QA-310]



1.31 Anchors and Duration

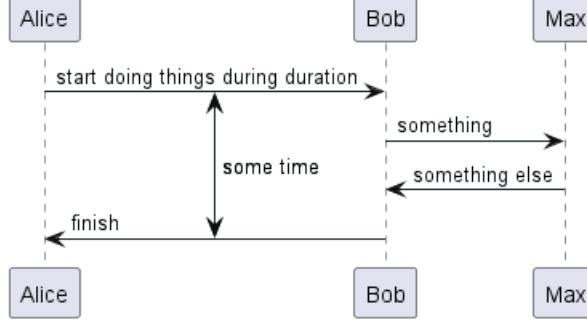
With `teoz` it is possible to add anchors to the diagram and use the anchors to specify duration time.

```
@startuml
!pragma teoz true
```

```
{start} Alice -> Bob : start doing things during duration
Bob -> Max : something
Max -> Bob : something else
{end} Bob -> Alice : finish

{start} <-> {end} : some time
```

```
@enduml
```



You can use the `-P` command-line option to specify the pragma:

```
java -jar plantuml.jar -Pteoz=true
```

[Ref. issue-582]

1.32 Estereotipos y marcas

Es posible añadir estereotipos a participantes usando `<< y >>`.

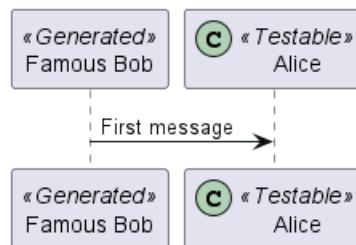
En el estereotipo, puedes añadir un carácter marcado en un círculo coloreado usando la sintaxis `(X,color)`.

```
@startuml
```

```
participant "Famous Bob" as Bob << Generated >>
participant Alice << (C,#ADD1B2) Testable >>
```

Bob->Alice: First message

```
@enduml
```



Por defecto, *guillemet* (comillas) son usadas para mostrar el estereotipo. Puedes cambiar este comportamiento usando `skinparam guillemet`:

```
@startuml
```

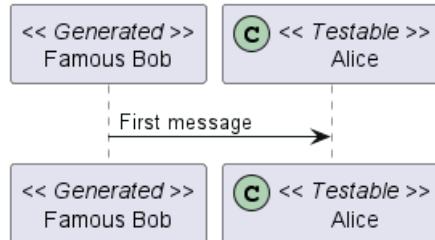
```
skinparam guillemet false
```



```
participant "Famous Bob" as Bob << Generated >>
participant Alice << (C,#ADD1B2) Testable >>
```

Bob->Alice: First message

@enduml

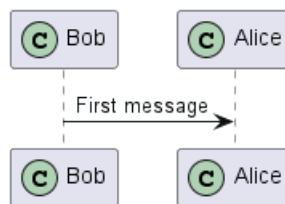


@startuml

```
participant Bob << (C,#ADD1B2) >>
participant Alice << (C,#ADD1B2) >>
```

Bob->Alice: First message

@enduml



1.33 Mayor información en los títulos

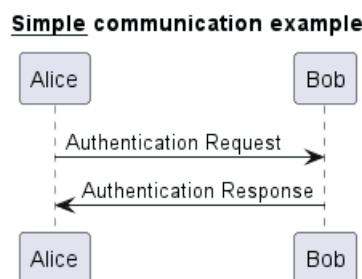
Puedes usar sintaxis de Creole en el título.

@startuml

title __Simple__ **communication** example

```
Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response
```

@enduml



Puedes añadir una nueva línea usando \n en la descripción del título.

@startuml

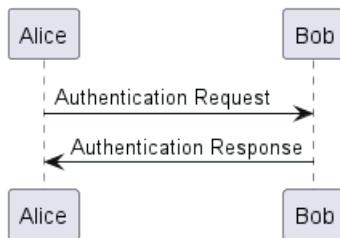
title __Simple__ communication example\nnon several lines



```
Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response
```

@enduml

Simple communication example on several lines



Además puedes definir un título en varias líneas usando las palabras reservadas `title` y `end title`.

@startuml

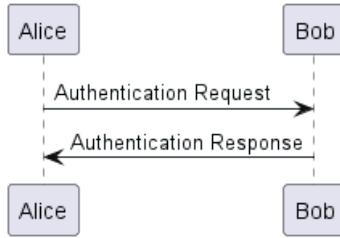
```
title
<u>Simple</u> communication example
on <i>several</i> lines and using <font color=red>html</font>
This is hosted by <img:sourceforge.jpg>
end title
```

```
Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response
```

@enduml

Simple communication example on several lines and using html

This is hosted by



1.34 Entorno de participante

Es posible dibujar una caja alrededor de algunos participantes, usando los comandos `box` y `end box`.

Puedes añadir un título opcional o un color de fondo opcional, después de la palabra reservada `box`.

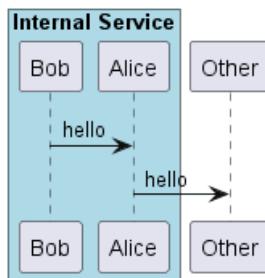
@startuml

```
box "Internal Service" #LightBlue
participant Bob
participant Alice
end box
participant Other
```

```
Bob -> Alice : hello
Alice -> Other : hello
```



@enduml



1.35 Removiendo pie de página

Puedes usar las palabras reservadas `hide footbox` para remover el pie de página del diagrama.

@startuml

```

hide footbox
title Footer removed

```

```

Alice --> Bob: Authentication Request
Bob --> Alice: Authentication Response

```

@enduml



1.36 Personalización (Skinparam)

Puedes usar el comando `skinparam` para cambiar los colores y las fuentes de los dibujos

Puedes usar este comando:

- En la definición del diagrama, como cualquier otro comando,
- En un archivo incluido,
- En un archivo de configuración, proporcionado en la consola de comandos o en el ANT task.

También puedes cambiar otros parámetros de renderización, como se ve en los siguientes ejemplos

```

@startuml
skinparam sequenceArrowThickness 2
skinparam roundcorner 20
skinparam maxmessagesize 60
skinparam sequenceParticipant underline

```

```

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

```

```

User -> A: DoWork
activate A

```



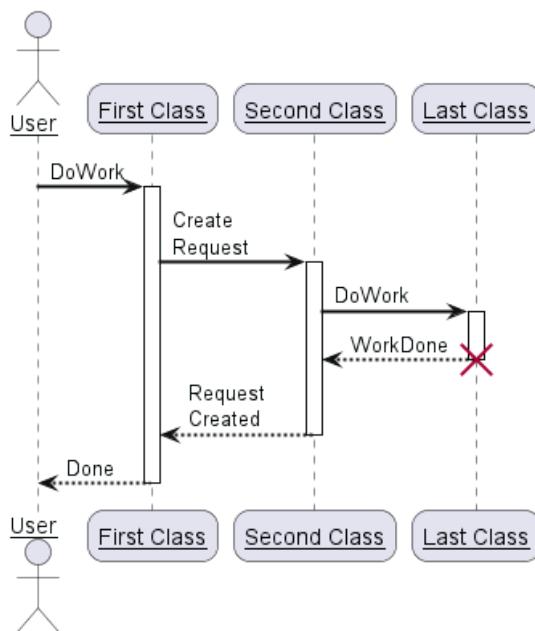
A -> B: Create Request
activate B

B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml



@startuml
skinparam backgroundColor #EEEBDC
skinparam handwritten true

```

skinparam sequence {
    ArrowColor DeepSkyBlue
    ActorBorderColor DeepSkyBlue
    LifeLineBorderColor blue
    LifeLineBackgroundColor #A9DCDF

    ParticipantBorderColor DeepSkyBlue
    ParticipantBackgroundColor DodgerBlue
    ParticipantFontName Impact
    ParticipantFontSize 17
    ParticipantFontColor #A9DCDF
}

ParticipantBackgroundColor aqua
ActorFontColor DeepSkyBlue
ActorFontSize 17
ActorFontName Aapex
}
  
```

 Guía de Referencia del Lenguaje PlantUML (1.2023.11)

```

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

```

```

User -> A: DoWork
activate A

```

```

A -> B: Create Request
activate B

```

```

B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

```

```

B --> A: Request Created
deactivate B

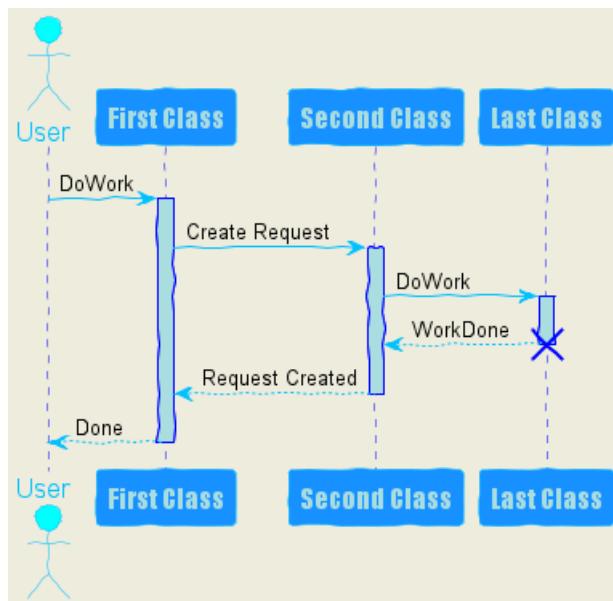
```

```

A --> User: Done
deactivate A

```

```
@enduml
```



1.37 Cambiando el relleno

Es posible ajustar algunos parámetros de relleno

```

@startuml
skinparam ParticipantPadding 20
skinparam BoxPadding 10

```

```

box "Foo1"
participant Alice1
participant Alice2
end box
box "Foo2"
participant Bob1
participant Bob2

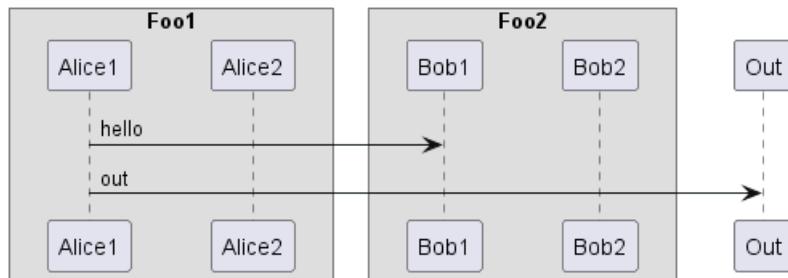
```



```

end box
Alice1 -> Bob1 : hello
Alice1 -> Out : out
@enduml

```



1.38 Appendix: Examples of all arrow type

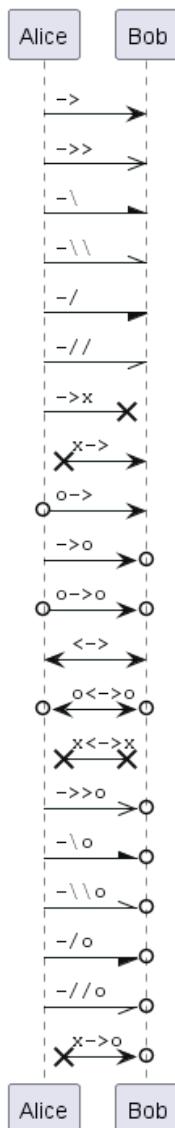
1.38.1 Normal arrow

```

@startuml
participant Alice as a
participant Bob as b
a -> b : ""-> ""
a ->> b : ""->> ""
a -\ b : ""-\ ""
a -\\\" b : ""-\\\\\""
a -/ b : ""-/ ""
a -// b : ""-///
a ->x b : ""->x ""
a x-> b : ""x-> ""
a o-> b : ""o-> ""
a ->o b : ""->o ""
a o->o b : ""o->o ""
a <-> b : ""<-> ""
a o<->o b : ""o<->o""
a x<->x b : ""x<->x"""
a ->>o b : ""->>o """
a -\o b : ""-\o ""
a -\\o b : ""-\\\\o"""
a -/o b : ""-/o """
a -//o b : ""-/o """
a x->o b : ""x->o """
@enduml

```





1.38.2 Itself arrow

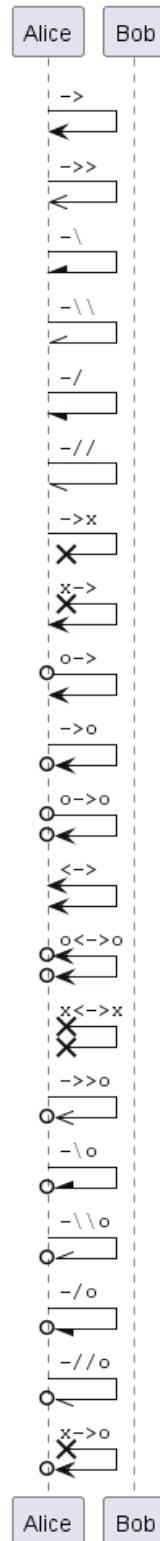
```

@startuml
participant Alice as a
participant Bob as b
a -> a : ""-> ""
a ->> a : ""->> ""
a -\ a : ""-\ ""
a -\\ a : ""-\\\\""
a -/ a : ""-/ ""
a -// a : ""-// ""
a ->x a : ""->x ""
a x-> a : ""x-> ""
a o-> a : ""o-> ""
a ->o a : ""->o ""
a o->o a : ""o->o ""
a <-> a : ""<-> ""
a o<->o a : ""o<->o ""
a x<->x a : ""x<->x ""
a ->>o a : ""->>o ""
a -\o a : ""-\o ""
a -\\o a : ""-\\\o ""

```



```
a -/o    a : ""-/o   ""
a -//o   a : ""-//o  ""
a x->o  a : ""x->o ""
@enduml
```



1.38.3 Incoming and outgoing messages (with '[', ']')

1.38.4 Incoming messages (with '[')

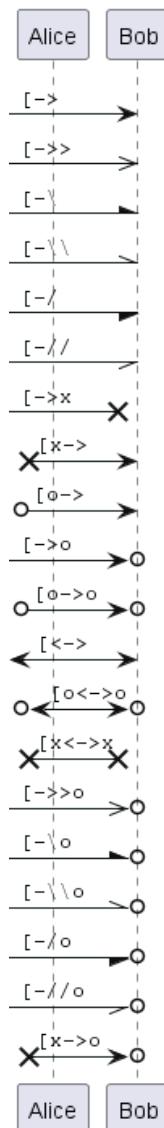
```
@startuml
```



```
participant Alice as a
participant Bob   as b
[->    b : ""[->  ""
[-->   b : ""[-->  ""
[-\    b : ""[-\   ""
[-\\   b : ""[-\\\\\""
[-/    b : ""[-/   ""
[-//   b : ""[-//  ""
[->x  b : ""[->x ""
[x->  b : ""[x-> ""
[o->  b : ""[o-> ""
[->o  b : ""[->o ""
[o->o b : ""[o->o ""
[<->  b : ""[<-> ""
[o<->o b : ""[o<->o""
[x<->x b : ""[x<->x""
[-->>o b : ""[-->>o ""
[-\o   b : ""[-\o   ""
[-\\o  b : ""[-\\\\o""
[-/o   b : ""[-/o   ""
[-//o  b : ""[-//o  ""
[x->o b : ""[x->o ""
```

@enduml



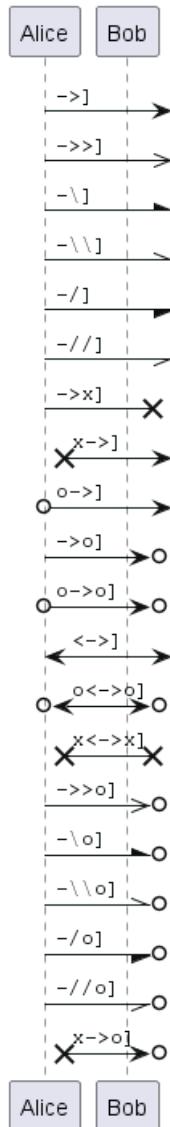


1.38.5 Outgoing messages (with ']')

```
@startuml
participant Alice as a
participant Bob as b
a ->] : ">]"
a ->>] : ">>]"
a -\] : "\]"
a -\\/] : "-\\\\]"
a -/] : "-/]"
a -//] : "-//]"
a ->x] : "->x]"
a x->] : "x->]"
a o->] : "o->]"
a ->o] : "->o]"
a o->o] : "o->o]"
a <->] : "<->]"
a o<->o] : "o<->o]"
a x<->x] : "x<->x]"
a ->>o] : "->>o]"
a -\o] : "\o]"
a -\\o] : "\\o]"
```



```
a -/o]      : """-/o]   ""
a -//o]     : """-//o]  ""
a x->o]    : """x->o]  ""
@enduml
```



1.38.6 Short incoming and outgoing messages (with '?')

1.38.7 Short incoming (with '?')

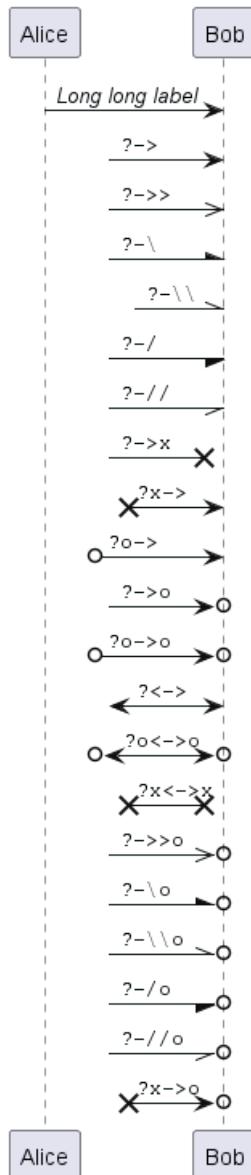
```
@startuml
participant Alice as a
participant Bob   as b
a ->    b : //Long long label//?
->    b : """?->  """
?->>  b : """?->>  """
?-\  
 b : """?-\  
  """
?-\\\  
 b : """?-\\\\\""""
?-/  
 b : """?-/-  """
?-//  
 b : """?-//  """
?->x  b : """?->x  """
?x->  b : """?x->  """
?o->  b : """?o->  """
?->o  b : """?->o  """
```



```

?o->o    b : """?o->o """
?<->    b : """?<-> """
?o<->o  b : """?o<->o"""
?x<->x  b : """?x<->x"""
?->>o   b : """?->>o """
?-\\o    b : """?-\\o """
?-//o    b : """?-//o """
?x->o   b : """?x->o """
@enduml

```



1.38.8 Short outgoing (with '?')

```

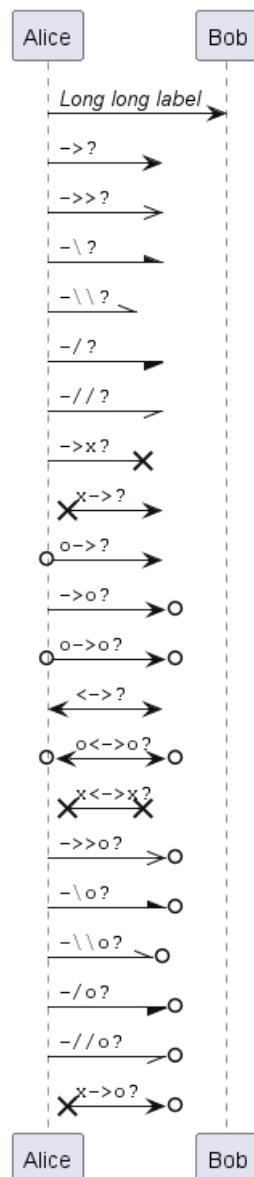
@startuml
participant Alice as a
participant Bob as b
a -> b : //Long long label// 
a ->? : """->? """
a ->>? : """->>? """
a -\\? : """-\\? """

```



```

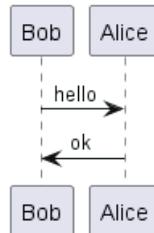
a -\?\? : """-\\\?\?""
a -/?? : """-/?\?""
a -//?\? : """-//?\?""
a ->x?\? : """->x?\?""
a x->?\? : """x->?\?""
a o->?\? : """o->?\?""
a ->o?\? : """->o?\?""
a o->o?\? : """o->o?\?""
a <->?\? : """<->?\?""
a o<->o?\? : """o<->o?\?""
a x<->x?\? : """x<->x?\?""
a ->>o?\? : """->>o?\?""
a -\o?\? : """-\o?\?""
a -\|\o?\? : """-\|\o?\?""
a -/o?\? : """-/o?\?""
a -//o?\? : """-//o?\?""
a x->o?\? : """x->o?\?""
@enduml
  
```



1.39 Specific SkinParameter

1.39.1 By default

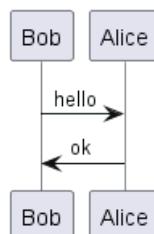
```
@startuml
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



1.39.2 LifelineStrategy

- nosolid (*by default*)

```
@startuml
skinparam lifelineStrategy nosolid
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```

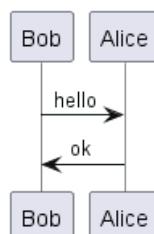


[Ref. QA-9016]

- solid

In order to have solid life line in sequence diagrams, you can use: `skinparam lifelineStrategy solid`

```
@startuml
skinparam lifelineStrategy solid
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



[Ref. QA-2794]

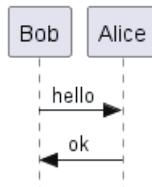
1.39.3 style strictuml

To be conform to strict UML (*for arrow style: emits triangle rather than sharp arrowheads*), you can use:

- `skinparam style strictuml`



```
@startuml
skinparam style strictuml
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



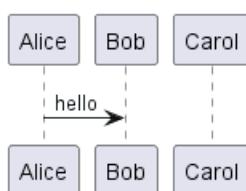
[Ref. QA-1047]

1.40 Hide unlinked participant

By default, all participants are displayed.

```
@startuml
participant Alice
participant Bob
participant Carol

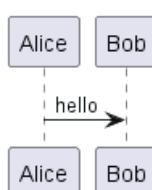
Alice -> Bob : hello
@enduml
```



But you can hide unlinked participant.

```
@startuml
hide unlinked
participant Alice
participant Bob
participant Carol

Alice -> Bob : hello
@enduml
```



[Ref. QA-4247]

1.41 Color a group message

It is possible to color a group messages:

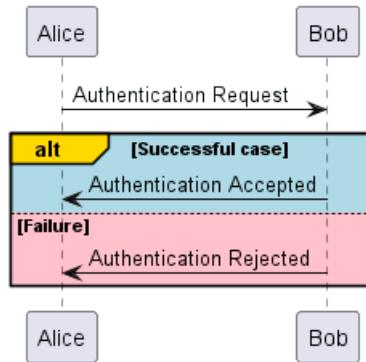
```
@startuml
Alice -> Bob: Authentication Request
alt#Gold #LightBlue Successful case
    Bob -> Alice: Authentication Accepted
```



```

else #Pink Failure
    Bob -> Alice: Authentication Rejected
end
@enduml

```



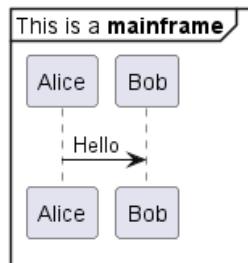
[Ref. QA-4750 and QA-6410]

1.42 Mainframe

```

@startuml
mainframe This is a **mainframe**
Alice->Bob : Hello
@enduml

```



[Ref. QA-4019 and Issue#148]

1.43 Slanted or odd arrows

You can use the (nn) option (before or after arrow) to make the arrows slanted, where nn is the number of shift pixels.

[Available only after v1.2022.6beta+]

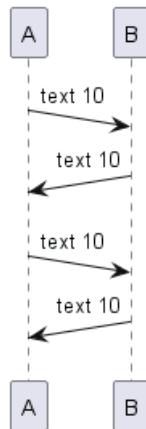
```

@startuml
A ->(10) B: text 10
B ->(10) A: text 10

A ->(10) B: text 10
A (10)<- B: text 10
@enduml

```

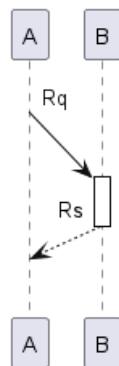




```

@startuml
A ->(40) B++: Rq
B -->(20) A--: Rs
@enduml

```

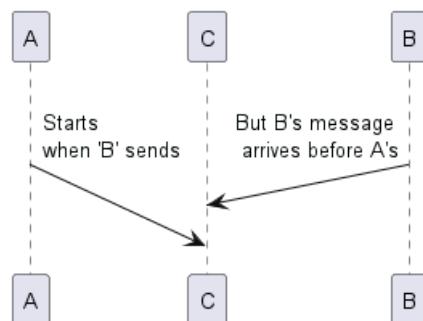


[Ref. QA-14145]

```

@startuml
!pragma teoz true
A ->(50) C: Starts\nwhen 'B' sends
& B ->(25) C: \nBut B's message\n arrives before A's
@enduml

```



[Ref. QA-6684]

```

@startuml
!pragma teoz true

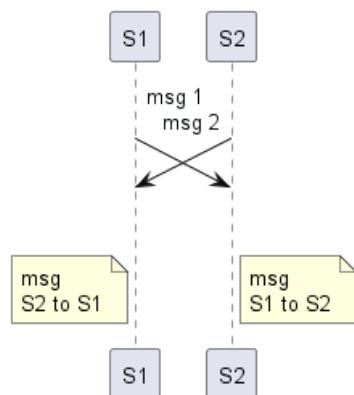
S1 ->(30) S2: msg 1\n
& S2 ->(30) S1: msg 2

note left S1: msg\nS2 to S1
& note right S2: msg\nS1 to S2

```



```
@enduml
```



[Ref. QA-1072]



2 Diagrama de Casos de Uso

Veamos algunos ejemplos:

Puedes deshabilitar el sombreado usando el comando "skinparam shadowing false"

PlantUML offers a unique approach to creating use case diagrams through its text-based language. One of the primary advantages of using PlantUML is its **simplicity and efficiency**. Instead of manually drawing shapes and connections, users can define their diagrams using intuitive and concise textual descriptions. This not only speeds up the diagram creation process but also ensures **consistency and accuracy**. The ability to integrate with various documentation platforms and its wide range of supported output formats make PlantUML a versatile tool for both developers and non-developers. Lastly, being **open-source**, PlantUML boasts a [strong community](<https://forum.plantuml.net/>) that continually contributes to its improvement and offers a wealth of resources for users at all levels.

2.1 Casos de uso

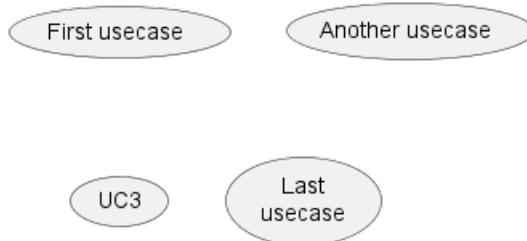
Los casos de uso se encierran entre paréntesis (porque dos paréntesis parecen un óvalo).

También puede utilizar la palabra clave `usecase` para definir un caso de uso . Y puede definir un alias, utilizando la palabra clave `as`. Este alias se utilizará más adelante, cuando se definan las relaciones.

```
@startuml
```

```
(First usecase)
(Another usecase) as (UC2)
usecase UC3
usecase (Last\nusecase) as UC4
```

```
@enduml
```



2.2 Actores

Los actores se encierran entre dos puntos.

También puedes usar la palabra reservada `actor` para definir un actor. Además puedes definir un alias, usando la palabra reservada `as`. Este alias será usado más adelante, cuando definamos relaciones.

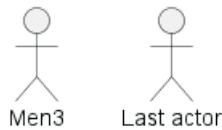
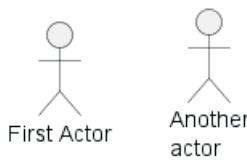
Veremos más adelante que las declaraciones de los actores son opcionales.

```
@startuml
```

```
:First Actor:
:Another\nactor: as Men2
actor Men3
actor :Last actor: as Men4
```

```
@enduml
```





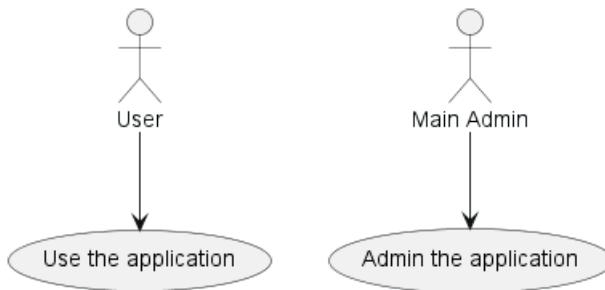
2.3 Cambiar el estilo del actor

Puedes cambiar el estilo del actor de hombre===== palo ===== (*===== por defecto =====*) a:

- un hombre impresionante con el comando `skinparam actorStyle awesome`;
- un hombre hueco con el comando `skinparam actorStyle hollow` .

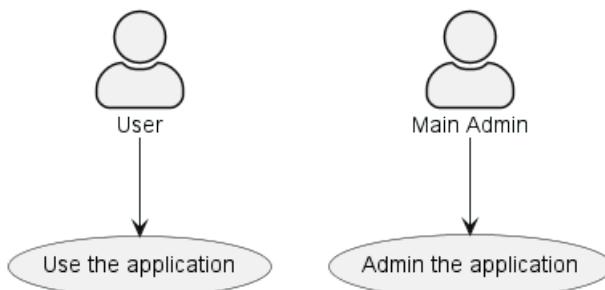
2.3.1 Hombre palo (*por defecto*)

```
@startuml
:User: --> (Use)
"Main Admin" as Admin
"Use the application" as (Use)
Admin --> (Admin the application)
@enduml
```



Hombre===== impresionante =====

```
@startuml
skinparam actorStyle awesome
:User: --> (Use)
"Main Admin" as Admin
"Use the application" as (Use)
Admin --> (Admin the application)
@enduml
```

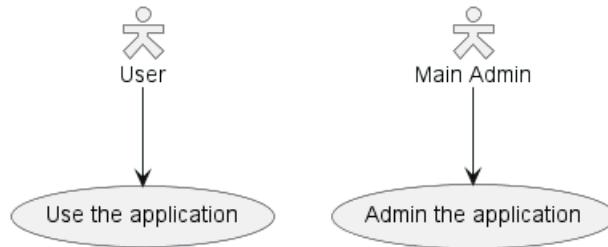


[Ref. QA-10493]



Hombre==== hueco =====

```
@startuml
skinparam actorStyle Hollow
:User: --> (Use)
"Main Admin" as Admin
"Use the application" as (Use)
Admin --> (Admin the application)
@enduml
```



[Ref. PR#396]

2.4 Descripción de Casos de uso

Si quiere realizar una descripción en varias líneas, puede usar citas (" ").

También puede usar los siguientes separadores: -- .. == ___. Y puede introducir títulos dentro de los separadores.

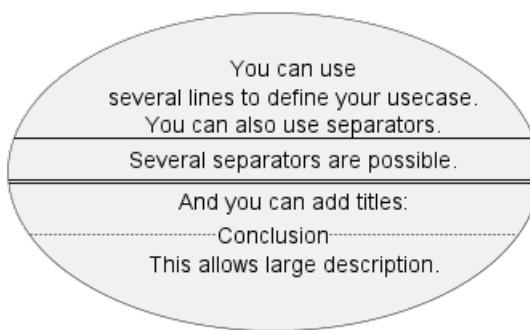
```
@startuml

usecase UC1 as "You can use
several lines to define your usecase.
You can also use separators.

-- 
Several separators are possible.

==
And you can add titles:
..Conclusion..
This allows large description."
```

@enduml



2.5 Utilice el paquete

Puede utilizar paquetes para agrupar actores o casos de uso

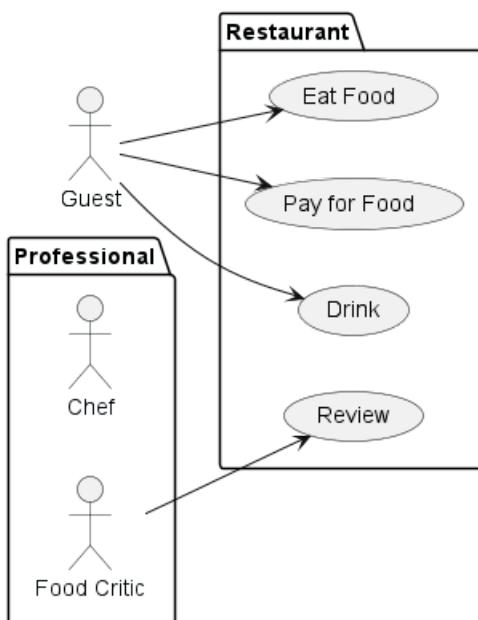
```
@startuml
left to right direction
actor Guest as g
package Professional {
```



```

actor Chef as c
actor "Food Critic" as fc
}
package Restaurant {
usecase "Eat Food" as UC1
usecase "Pay for Food" as UC2
usecase "Drink" as UC3
usecase "Review" as UC4
}
fc --> UC4
g --> UC1
g --> UC2
g --> UC3
@enduml

```



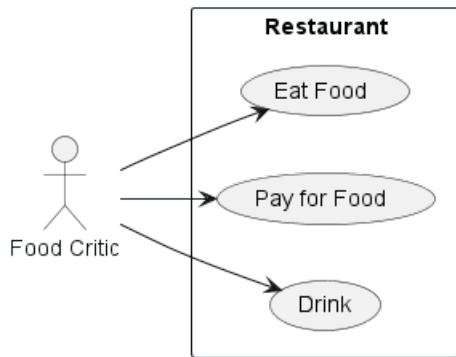
Puede utilizar `rectangle` para cambiar la visualización del paquete

```

@startuml
left to right direction
actor "Food Critic" as fc
rectangle Restaurant {
usecase "Eat Food" as UC1
usecase "Pay for Food" as UC2
usecase "Drink" as UC3
}
fc --> UC1
fc --> UC2
fc --> UC3
@enduml

```





2.6 Ejemplo básico

Para relacionar actores y casos de uso, la flecha `-->` es usada.

Cuento más guiones - en la flecha, más larga será la misma. Puedes añadir una etiqueta en la flecha, añadiendo el carácter : en la definición de la flecha.

En este ejemplo, puedes ver que *User* no ha sido definido, y es usado como un actor.

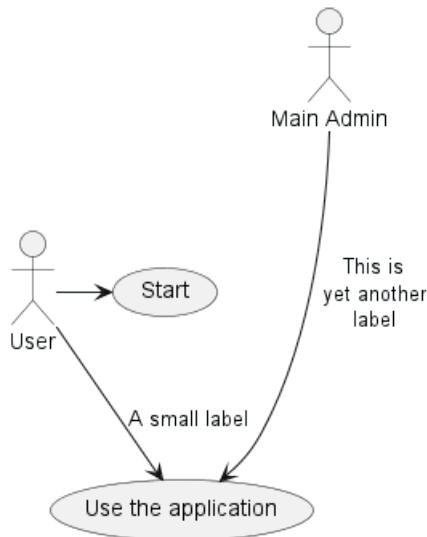
`@startuml`

```

User -> (Start)
User --> (Use the application) : A small label

:Main Admin: ---> (Use the application) : This is\nyet another\nlabel
  
```

`@enduml`



2.7 Extensión

Si un actor/caso de uso extiende a otro, puedes usar el símbolo `<|--`.

```

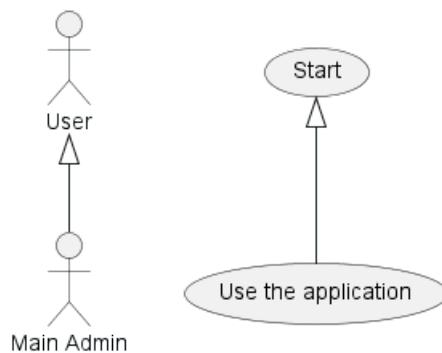
@startuml
:Main Admin: as Admin
(Use the application) as (Use)

User <|-- Admin
(Start) <|-- (Use)

@enduml
  
```

`@enduml`





2.8 Usando notas

Puedes usar las palabras claves: `note left of`, `note right of`, `note top of`, `note bottom of`, para añadir notas relacionadas a un objeto en particular.

También se puede añadir un nota solitaria con la palabra clave `note`, y después relacionarla con otro objeto usando el símbolo ...

```

@startuml
:Main Admin: as Admin
(Use the application) as (Use)

User -> (Start)
User --> (Use)

Admin ---> (Use)
  
```

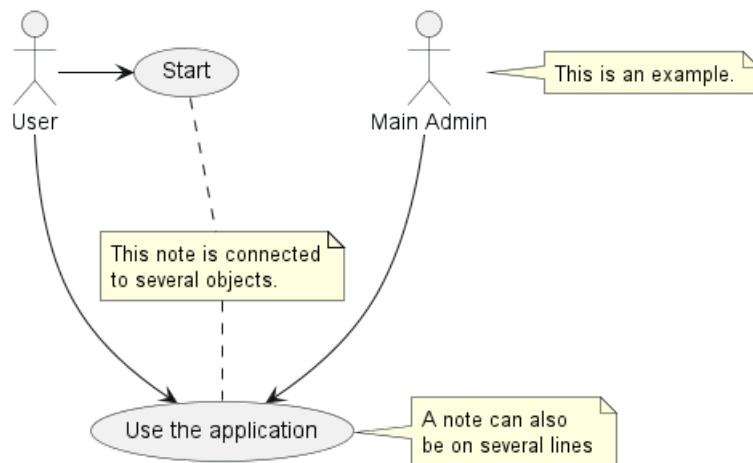
`note right of Admin : This is an example.`

```

note right of (Use)
  A note can also
  be on several lines
end note
  
```

```

note "This note is connected\nto several objects." as N2
(Start) .. N2
N2 .. (Use)
@enduml
  
```



2.9 Estereotipos

Puedes añadir estereotipos mientras defines actores y casos de uso, usando << y >>.

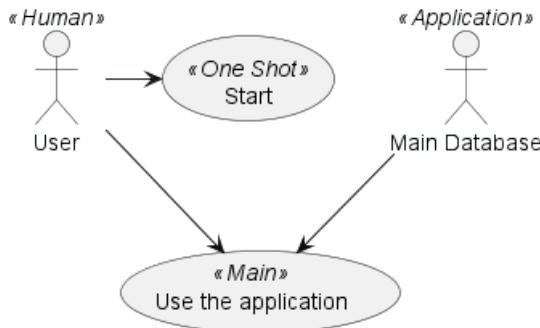


```
@startuml
User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>
```

```
User -> (Start)
User --> (Use)
```

```
MySql --> (Use)
```

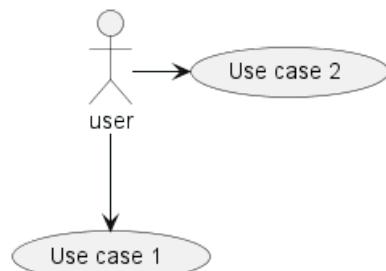
```
@enduml
```



2.10 Cambio de dirección de las flechas

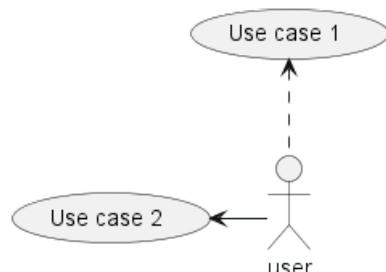
Por defecto, los enlaces entre clases tienen dos guiones -- y están orientados verticalmente. Es posible utilizar un enlace horizontal poniendo un solo guión (o punto) como este

```
@startuml
:user: --> (Use case 1)
:user: -> (Use case 2)
@enduml
```



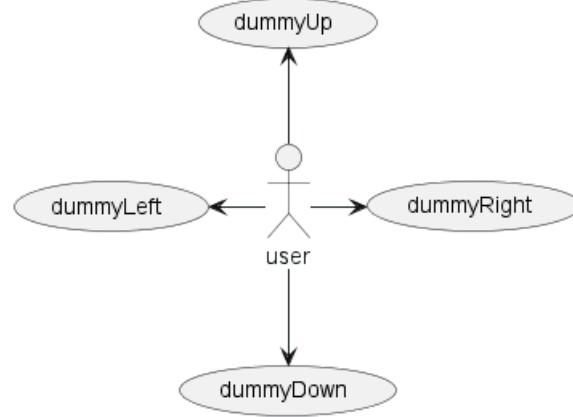
También se puede cambiar la dirección invirtiendo el enlace:

```
@startuml
(Use case 1) <.. :user:
(Use case 2) <- :user:
@enduml
```



También es posible cambiar la dirección de la flecha añadiendo las palabras clave `left`, `right`, `up` o `down` dentro de la flecha

```
@startuml
:user: -left-> (dummyLeft)
:user: -right-> (dummyRight)
:user: -up-> (dummyUp)
:user: -down-> (dummyDown)
@enduml
```

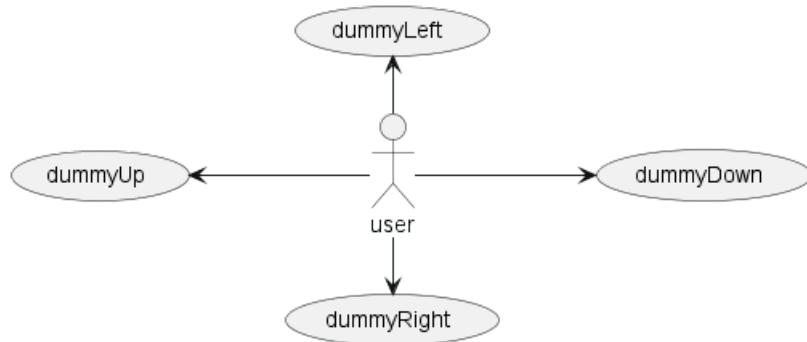


Puede acortar la flecha utilizando sólo el primer carácter de la dirección (por ejemplo, `-d-` en lugar de `-down-`) o los dos primeros caracteres (`-do-`).

Tenga en cuenta que no debe abusar de esta funcionalidad : *Graphviz* suele dar buenos resultados sin retoques.

Y con el `left to right direction` parámetro

```
@startuml
left to right direction
:user: -left-> (dummyLeft)
:user: -right-> (dummyRight)
:user: -up-> (dummyUp)
:user: -down-> (dummyDown)
@enduml
```

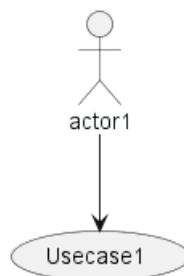


2.11 Dividiendo los diagramas

La palabra clave `newpage` divide su diagrama en varias páginas o imágenes.

```
@startuml
:actor1: --> (Usecase1)
newpage
:actor2: --> (Usecase2)
@enduml
```





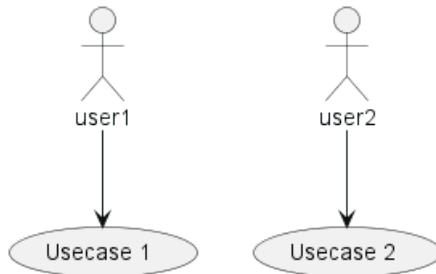
2.12 Dirección: de izquierda a derecha

El comportamiento general cuando se construye un diagrama, es **top to bottom**.

```

@startuml
'default
top to bottom direction
user1 --> (Usecase 1)
user2 --> (Usecase 2)

@enduml
  
```



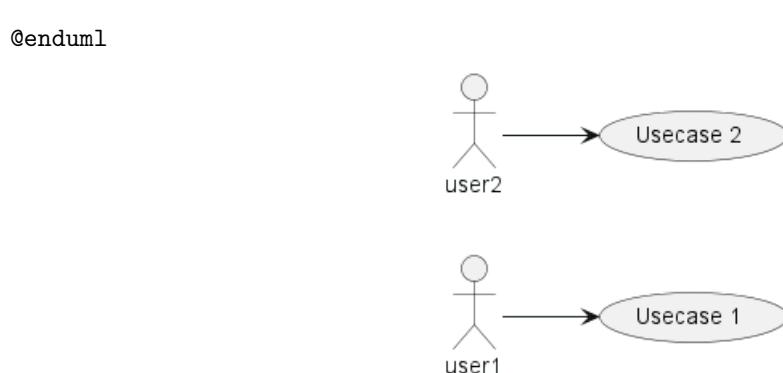
Puede cambiar a **left to right** usando el comando `left to right direction`. En ocasiones, el resultado es mejor con esta dirección.

```

@startuml

left to right direction
user1 --> (Usecase 1)
user2 --> (Usecase 2)

@enduml
  
```



2.13 Personalización (Skinparam)

Puedes usar el comando `skinparam` para cambiar los colores y las fuentes de los dibujos

Puedes usar este comando:

- En la definición del diagrama, como cualquier otro comando,



- En un archivo incluido,
- En un archivo de configuración, proporcionado en la consola de comandos o en el ANT task.

Puedes definir colores y fuentes específicas para los actores y casos de uso estereotipados.

```
@startuml
skinparam handwritten true

skinparam usecase {
BackgroundColor DarkSeaGreen
BorderColor DarkSlateGray

BackgroundColor<< Main >> YellowGreen
BorderColor<< Main >> YellowGreen

ArrowColor Olive
ActorBorderColor black
ActorFontName Courier

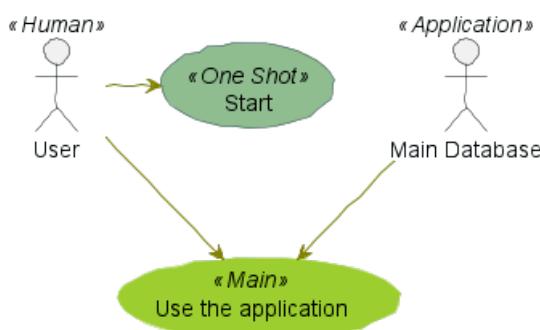
ActorBackgroundColor<< Human >> Gold
}

User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>

User -> (Start)
User --> (Use)

MySql --> (Use)

@enduml
```

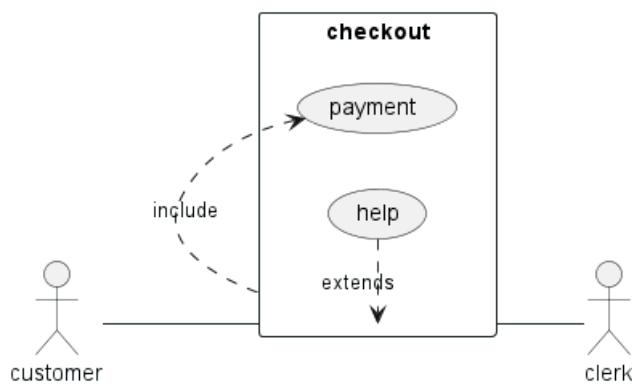


2.14 Un ejemplo completo

```
@startuml
left to right direction
skinparam packageStyle rectangle
actor customer
actor clerk
rectangle checkout {
    customer -- (checkout)
    (checkout) .> (payment) : include
    (help) .> (checkout) : extends
    (checkout) -- clerk
}
```



@enduml



2.15 Business Use Case

You can add / to make Business Use Case.

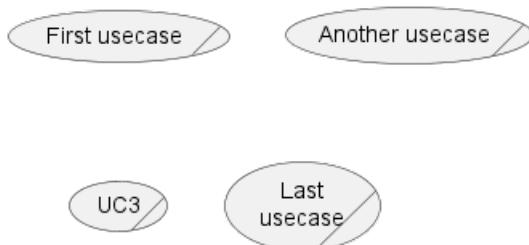
2.15.1 Business Usecase

@startuml

```

(First usecase) /
(Another usecase) / as (UC2)
usecase/ UC3
usecase/ (Last\usecase) as UC4
  
```

@enduml



2.15.2 Business Actor

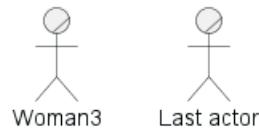
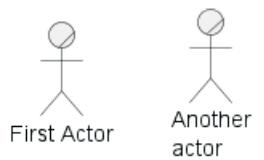
@startuml

```

:First Actor:/
:Another\actor:/ as Man2
actor/ Woman3
actor/ :Last actor: as Person1
  
```

@enduml





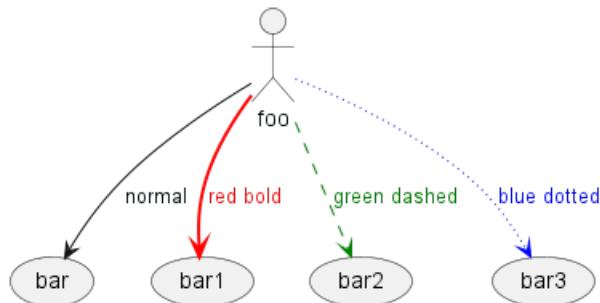
[Ref. QA-12179]

2.16 Change arrow color and style (inline style)

You can change the color or style of individual arrows using the inline following notation:

- `#color;line.[bold|dashed|dotted];text:color`

```
@startuml
actor foo
foo --> (bar) : normal
foo --> (bar1) #line:red;line.bold;text:red : red bold
foo --> (bar2) #green;line.dashed;text:green : green dashed
foo --> (bar3) #blue;line.dotted;text:blue : blue dotted
@enduml
```



[Ref. QA-3770 and QA-3816] [See similar feature on deployment-diagram or class diagram]

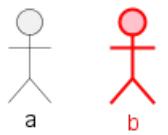
2.17 Change element color and style (inline style)

You can change the color or style of individual element using the following notation:

- `# [color|back:color];line:color;line.[bold|dashed|dotted];text:color`

```
@startuml
actor a
actor b #pink;line:red;line.bold;text:red
usecase c #palegreen;line:green;line.dashed;text:green
usecase d #aliceblue;line:blue;line.dotted;text:blue
@enduml
```





[Ref. QA-5340 and adapted from QA-6852]

2.18 Display JSON Data on Usecase diagram

2.18.1 Simple example

```
@startuml
allowmixing

actor      Actor
usecase    Usecase
```

```
json JSON {
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}
@enduml
```



JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

For another example, see on JSON page.



3 Diagrama de clase

Los diagramas de clase se diseñan utilizando una sintaxis que refleja la empleada tradicionalmente en los lenguajes de programación. Este parecido fomenta un entorno familiar para los desarrolladores, facilitando así un proceso de creación de diagramas más sencillo e intuitivo.

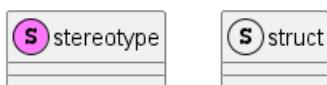
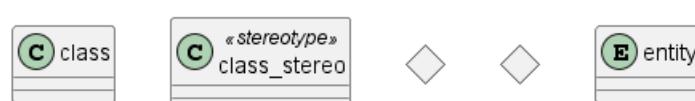
Este enfoque de diseño no sólo es sucinto, sino que también permite crear representaciones que son a la vez concisas y expresivas. Por otra parte, permite la representación de las relaciones entre las clases a través de una sintaxis que se hace eco de la de los diagramas de secuencia, allanando el camino para una representación fluida y perspicaz de las interacciones de clase.

Más allá de las representaciones estructurales y relacionales, la sintaxis de los diagramas de clase soporta enriquecimientos adicionales, tales como la inclusión de notas y la aplicación de colores, permitiendo a los usuarios crear diagramas que son a la vez informativos y visualmente atractivos.

Usted puede aprender más acerca de algunos de los comandos comunes en PlantUML para mejorar su experiencia de creación de diagramas.

3.1 Elemento declarante

```
@startuml
abstract      abstract
abstract class "abstract class"
annotation    annotation
circle        circle
()            circle_short_form
class         class
class         class_stereo  <<stereotype>>
diamond       diamond
<>           diamond_short_form
entity        entity
enum          enum
exception     exception
interface     interface
metaclass    metaclass
protocol      protocol
stereotype   stereotype
struct        struct
@enduml
```



[Ref. para protocol y struct: GH-1028, para exception: QA-16258]

3.2 Relación entre clases

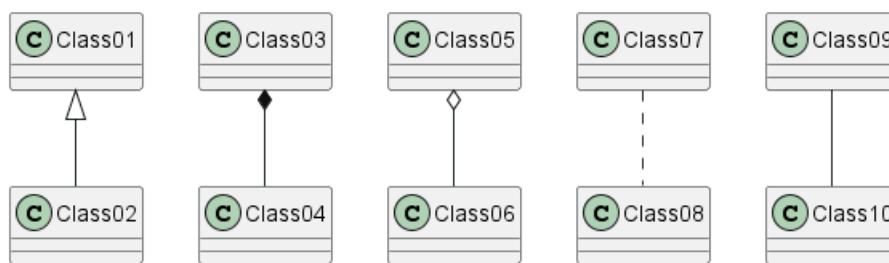
Las relaciones entre clases se definen usando los siguientes símbolos:

Type	Symbol	Drawing
Extension	< --	
Composition	*---	
Aggregation	o---	

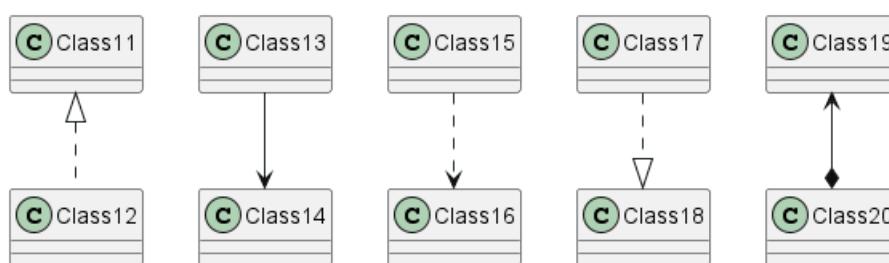
Es posible intercambiar -- por .. para tener lineas punteadas.

Sabiendo esas reglas, es posible sacar los siguientes dibujos:

```
@startuml
Class01 <|-- Class02
Class03 *--- Class04
Class05 o--- Class06
Class07 .. Class08
Class09 -- Class10
@enduml
```

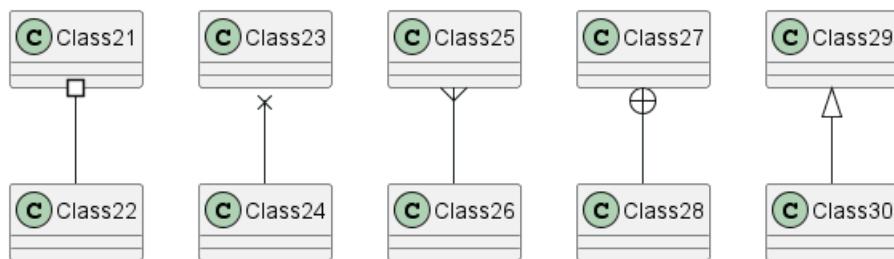


```
@startuml
Class11 <|.. Class12
Class13 --> Class14
Class15 ..> Class16
Class17 ..|> Class18
Class19 <--* Class20
@enduml
```



```
@startuml
Class21 #--- Class22
Class23 x--- Class24
Class25 }--- Class26
Class27 +--- Class28
Class29 ^--- Class30
@enduml
```





3.3 Etiquetas en las relaciones

Es posible añadir etiquetas en las relaciones, usando `:`, seguido del texto de la etiqueta.

Para la cardinalidad, puede usar comillas dobles "" en cada lado de la relación.

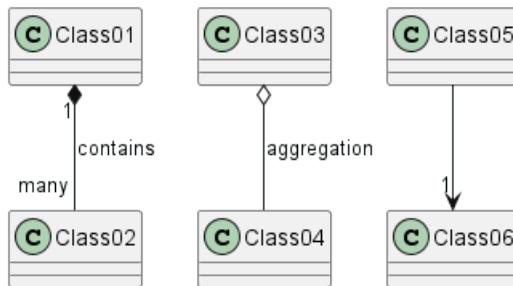
```
@startuml
```

```
Class01 "1" *-- "many" Class02 : contains
```

```
Class03 o-- Class04 : aggregation
```

```
Class05 --> "1" Class06
```

```
@enduml
```

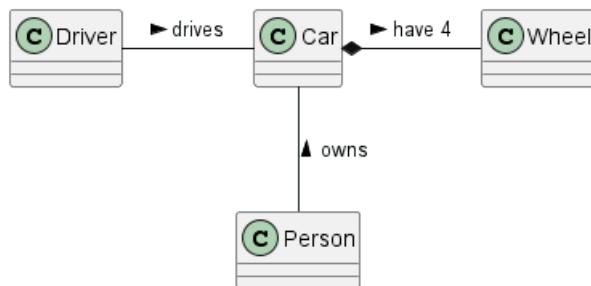


Se puede añadir una flecha extra apuntando a un objeto, mostrando que objeto actúa sobre el otro objeto, usando < o > al inicio o al final de la etiqueta.

```
@startuml
class Car
```

```
Driver -> Car : drives >
Car *--> Wheel : have 4 >
Car --> Person : < owns
```

```
@enduml
```



3.4 Sin usar letras

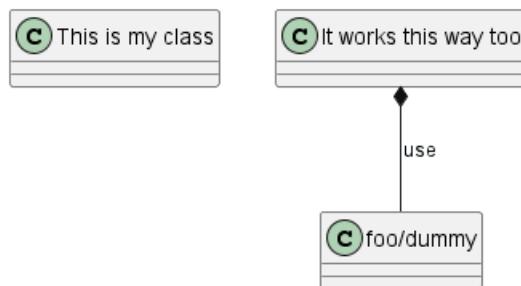
Si no desea usar letras en la visualización de la clase (o enum...), puede:



- Utilizar la palabra reservada `as` en la definición de la clase
- Colocar comillas "" alrededor del nombre de la clase

```
@startuml
class "This is my class" as class1
class class2 as "It works this way too"

class2 *-- "foo/dummy" : use
@enduml
```



Also note that names starting with "\$" are valid, but to assign an alias to such element the name must be put between quotes """".

3.5 Añadir métodos

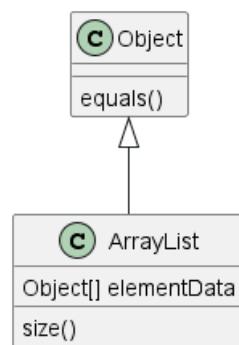
Para declarar campos y métodos, se puede utilizar el símbolo `:` seguido del nombre del campo o del método.

El sistema busca paréntesis para elegir entre métodos y campos

```
@startuml
Object <|-- ArrayList

Object : equals()
ArrayList : Object[] elementData
ArrayList : size()
```

`@enduml`



También es posible agrupar entre paréntesis {} todos los campos y métodos.

Tenga en cuenta que la sintaxis es muy flexible en cuanto al orden tipo/nombre.

```
@startuml
class Dummy {
    String data
    void methods()
}

class Flight {
```



```

flightNumber : Integer
departureTime : Date
}
@enduml

```



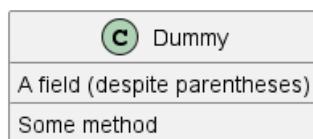
Puedes utilizar los modificadores `{field}` y `{method}` para anular el comportamiento por defecto del analizador sintáctico sobre los campos y métodos

```

@startuml
class Dummy {
    {field} A field (despite parentheses)
    {method} Some method
}

@enduml

```



3.6 Definiendo la visibilidad

Cuando defines propiedades o métodos, puedes usar caracteres para establecer la visibilidad que les correspondan:

Character	Icon for field	Icon for method	Visibility
-	□	■	private
#	◊	◊	protected
~	△	△	package private
+	○	●	public

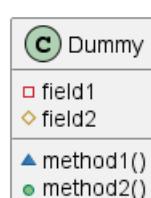
```

@startuml

class Dummy {
    -field1
    #field2
    ~method1()
    +method2()
}

@enduml

```



Puedes desactivar esta característica usando el comando `skinparam classAttributeIconSize 0`:

```

@startuml
skinparam classAttributeIconSize 0
class Dummy {

```



```
-field1
```

```
#field2
```

```
~method1()
```

```
+method2()
```

```
}
```

```
@enduml
```

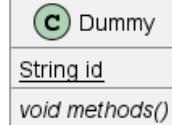


3.7 Abstracto y Estático

Puedes definir métodos o propiedades abstractas y estáticas usando los modificadores `{static}` o `{abstract}`

Esos modificadores pueden ser usado al comienzo o al final de un línea. También puedes usar `{classifier}` en lugar de `{static}`.

```
@startuml
class Dummy {
    {static} String id
    {abstract} void methods()
}
@enduml
```



3.8 Cuerpo avanzado de las clases

Por defecto, las propiedades y los métodos son agrupados automáticamente por PlantUML. Puedes usar separadores para definir tu propia manera de ordenar las propiedades y los métodos. Son posibles los siguientes separadores: `--` .. `==` `__`.

También puedes definir títulos dentro de los separadores:

```
@startuml
class Foo1 {
    You can use
    several lines
    ..
    as you want
    and group
    ==
    things together.

    --
    You can have as many groups
    as you want
    --
    End of class
}
```

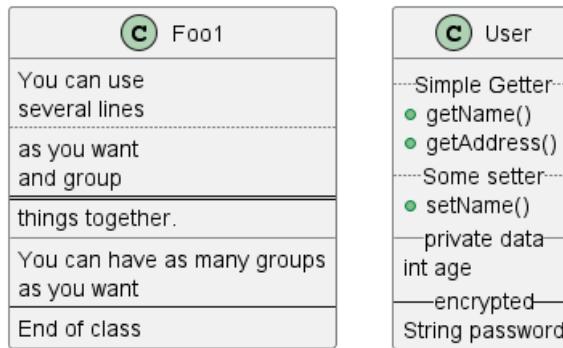


```

class User {
    ... Simple Getter ...
    + getName()
    + getAddress()
    ... Some setter ...
    + setName()
    -- private data --
    int age
    -- encrypted --
    String password
}

```

@enduml



3.9 Notas y estereotipos

Los estereotipos son definidos con la palabra clave `class`, `<<` and `>>`.

También puedes definir notas usando las palabras claves `note left of`, `note right of`, `note top of`, `note bottom of`.

Además puedes definir una nota en la última clase definida usando `note left`, `note right`, `note top`, `note bottom`.

Una nota también puede definirse solitariamente con la palabra clave `note`, y a continuación relacionarla con otro objeto usando el símbolo `...`

```

@startuml
class Object << general >>
Object <|-- ArrayList

note top of Object : In java, every class\nextends this one.

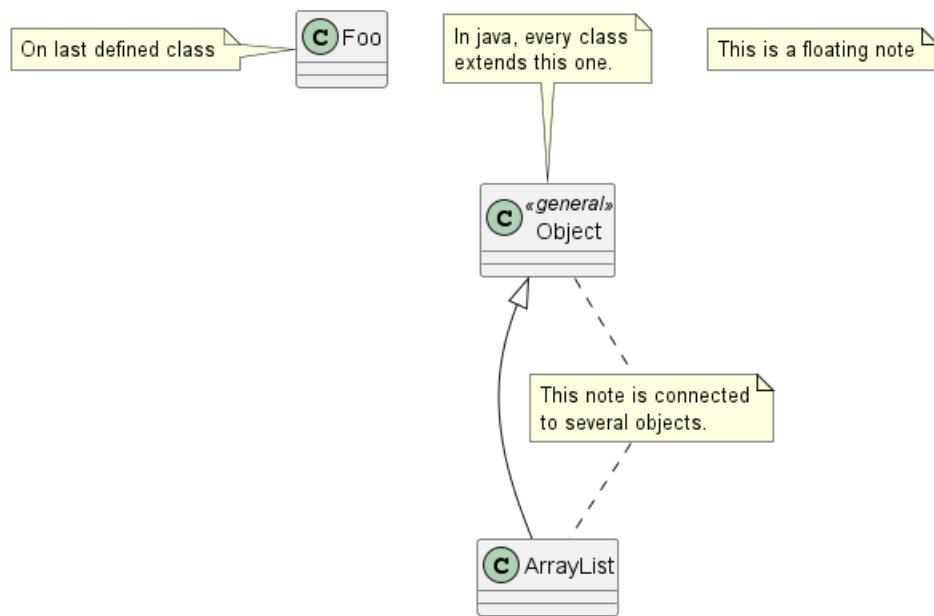
note "This is a floating note" as N1
note "This note is connected\nto several objects." as N2
Object .. N2
N2 .. ArrayList

class Foo
note left: On last defined class

@enduml

```





3.10 Más acerca de notas

También es posible usar algunas etiquetas HTML como (See Creole expression):

-
- <u>
- <i>
- <s>, , <strike>
- or
- <color:#AAAAAA> or <color:colorName>
- <size:nn> to change font size
- or <img:file>: the file must be accessible by the filesystem

También puedes tener una nota en varias líneas.

Es posible definir una nota en la última clase definida usando `note left`, `note right`, `note top`, `note bottom`.

`@startuml`

```

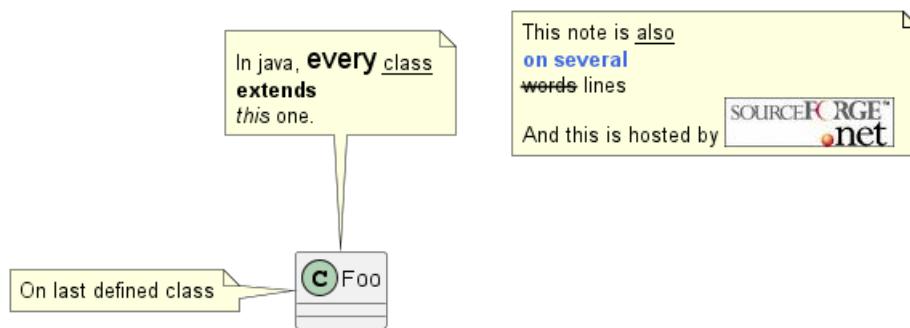
class Foo
note left: On last defined class

note top of Foo
  In java, <size:18>every</size> <u>class</u>
  <b>extends</b>
  <i>this</i> one.
end note

note as N1
  This note is <u>also</u>
  <b><color:royalBlue>on several</color>
  <s>words</s> lines
  And this is hosted by <img:sourceforge.jpg>
end note
  
```



```
@enduml
```



3.11 Note on field (field, attribute, member) or method

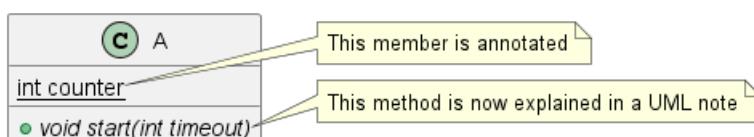
It is possible to add a note on field (field, attribute, member) or on method.

3.11.1 Constraint

- This cannot be used with top or bottom (*only left and right are implemented*)
- This cannot be used with namespaceSeparator ::

3.11.2 Note on field or method

```
@startuml
class A {
{static} int counter
+void {abstract} start(int timeout)
}
note right of A::counter
    This member is annotated
end note
note right of A::start
    This method is now explained in a UML note
end note
@enduml
```

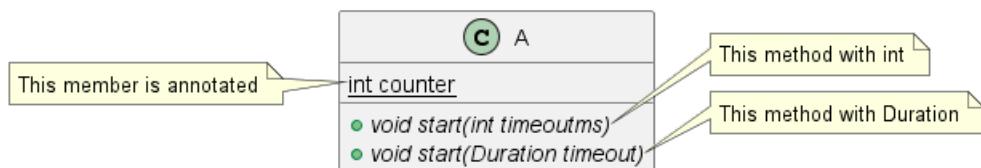


3.11.3 Note on method with the same name

```
@startuml
class A {
{static} int counter
+void {abstract} start(int timeoutms)
+void {abstract} start(Duration timeout)
}
note left of A::counter
    This member is annotated
end note
note right of A::"start(int timeoutms)"
    This method with int
end note
note right of A::"start(Duration timeout)"
    This method with Duration
@enduml
```



```
end note
@enduml
```



[Ref. QA-3474 and QA-5835]

3.12 Notas en enlaces

Es posible añadir una nota en un enlace, justo después de la definición de dicho enlace, usando `note on link`.

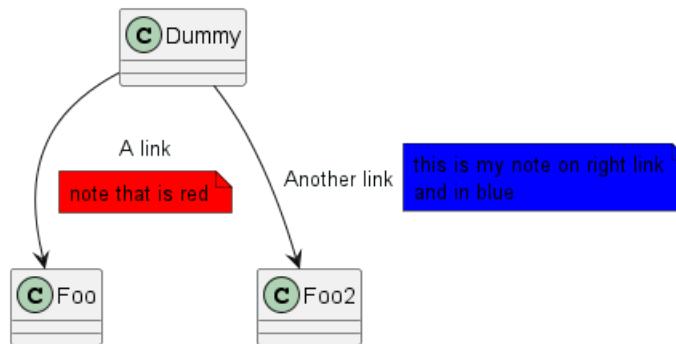
También puedes usar `note left on link`, `note right on link`, `note top on link`, `note bottom on link` si quieres cambiar la posición de la nota, en relación a una etiqueta.

```
@startuml
```

```
class Dummy
Dummy --> Foo : A link
note on link #red: note that is red

Dummy --> Foo2 : Another link
note right on link #blue
this is my note on right link
and in blue
end note
```

```
@enduml
```



3.13 Clases abstractas e interfaces

Puedes declarar una clase como abstracta usando las palabras claves `abstract` or `abstract class`.

La clase será impresa en *italic*.

Puedes usar también las palabras claves `interface`, `annotation` and `enum`.

```
@startuml
```

```
abstract class AbstractList
abstract AbstractCollection
interface List
interface Collection

List <|-- AbstractList
Collection <|-- AbstractCollection
```



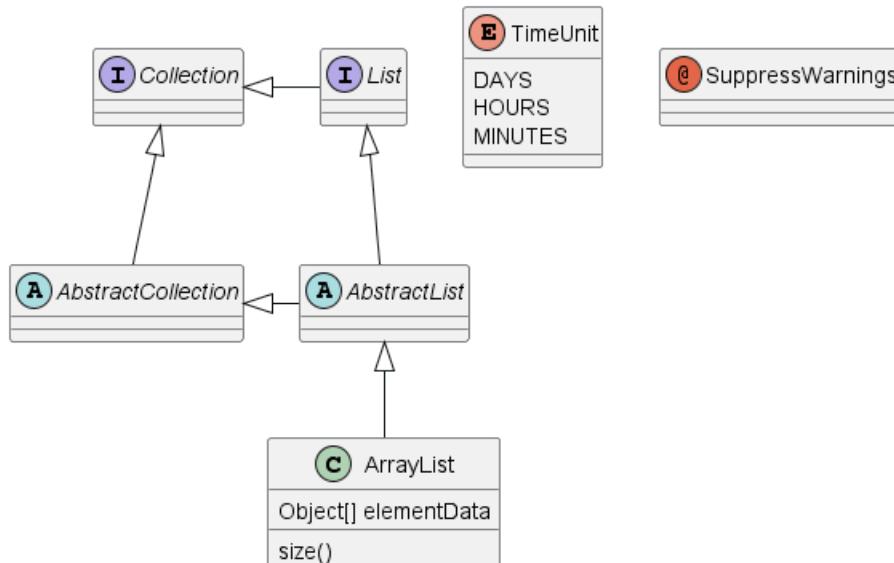
```
Collection <|- List
AbstractCollection <|- AbstractList
AbstractList <|-- ArrayList
```

```
class ArrayList {
    Object[] elementData
    size()
}
```

```
enum TimeUnit {
    DAYS
    HOURS
    MINUTES
}
```

```
@annotation SuppressWarnings
```

```
@enduml
```



[Ref. 'Annotation with members' [Issue#458](<https://github.com/plantuml/plantuml/issues/458>)]

3.14 Atributos, métodos... ocultos

Puede parametrizar la visualización de las clases usando el comando `hide/show`.

El comando básico es: `hide empty members`. Este comando ocultará atributos y métodos si están vacíos.

En lugar de `empty members`, puedes usar:

- `empty fields` o `empty attributes` para atributos vacíos.
- `empty methods` para métodos vacíos,
- `fields` o `attributes` que ocultará atributos, incluso si son descritos,
- `methods` que ocultará métodos, incluso si son descritos,
- `members` que ocultará atributos y métodos, incluso si son descritos.
- `circle` para el carácter encerrado en un círculo, en frente del nombre de clase.
- `stereotype` para el estereotipo.

También puede proporcionar, justo después las palabras clave `hide` o `show`:



- `class` para todas las clases,
- `interface` para todas las interfaces,
- `enum` para todos los enums,
- `<>foo1>` para clases que son estereotipadas con *foo1*,
- un nombre de clase existente.

Puedes usar varios comandos `show/hide` para definir reglas y excepciones.

`@startuml`

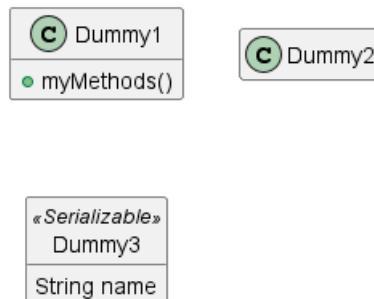
```
class Dummy1 {
    +myMethods()
}

class Dummy2 {
    +hiddenMethod()
}

class Dummy3 <<Serializable>> {
String name
}

hide members
hide <<Serializable>> circle
show Dummy1 methods
show <<Serializable>> fields
```

`@enduml`



3.15 Clases ocultas

También puedes usar el comando `show/hide` para ocultar clases.

Esto puede llegar a ser útil si defines una archivo `!included` muy grande y si deseas ocultar algunas clases después de la inclusión de dicho archivo.

`@startuml`

```
class Foo1
class Foo2

Foo2 *-- Foo1

hide Foo2

@enduml
```





3.16 Remove classes

You can also use the `remove` commands to remove classes.

This may be useful if you define a large `!included` file, and if you want to remove some classes after file inclusion.

```
@startuml
```

```
class Foo1
class Foo2

Foo2 *-- Foo1

remove Foo2

@enduml
```

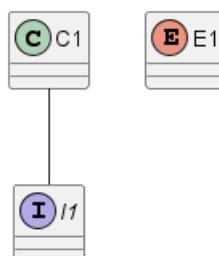


3.17 Hide, Remove or Restore tagged element or wildcard

You can put `$tags` (using `$`) on elements, then remove, hide or restore components either individually or by tags.

By default, all components are displayed:

```
@startuml
class C1 $tag13
enum E1
interface I1 $tag13
C1 -- I1
@enduml
```



But you can:

- hide `$tag13` components:

```
@startuml
class C1 $tag13
enum E1
interface I1 $tag13
```



C1 -- I1

```
hide $tag13
@enduml
```



- or remove \$tag13 components:

```
@startuml
class C1 $tag13
enum E1
interface I1 $tag13
C1 -- I1
```

```
remove $tag13
@enduml
```



- or remove \$tag13 and restore \$tag1 components:

```
@startuml
class C1 $tag13 $tag1
enum E1
interface I1 $tag13
C1 -- I1
```

```
remove $tag13
restore $tag1
@enduml
```



- or remove * and restore \$tag1 components:

```
@startuml
class C1 $tag13 $tag1
enum E1
interface I1 $tag13
C1 -- I1
```

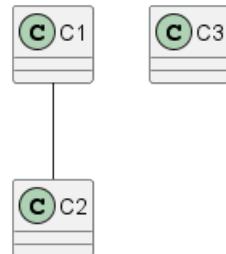
```
remove *
restore $tag1
@enduml
```

3.18 Hide or Remove unlinked class

By default, all classes are displayed:



```
@startuml
class C1
class C2
class C3
C1 --> C2
@enduml
```



But you can:

- hide @unlinked classes:

```
@startuml
class C1
class C2
class C3
C1 --> C2

hide @unlinked
@enduml
```



- or remove @unlinked classes:

```
@startuml
class C1
class C2
class C3
C1 --> C2

remove @unlinked
@enduml
```



[Adapted from QA-11052]



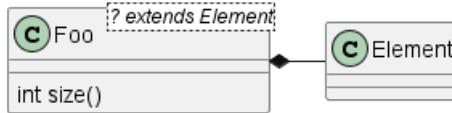
3.19 Uso de clases genéricas

También puedes usar los signos menor < y mayor > para definir el uso de clases genéricas.

```
@startuml
```

```
class Foo<? extends Element> {
    int size()
}
Foo *- Element
```

```
@enduml
```



Es posible desactivar este dibujo con el comando `skinparam genericDisplay old`.

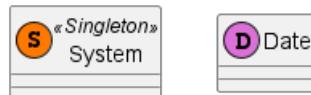
3.20 Círculo enmarcador específico

Usualmente, un carácter enmarcado en un círculo (C,I,E o A) es usado por clases, interfaces, enum y clases abstractas.

Pero puedes definir tu propio enmarcado para una clase cuando defines un estereotipo, añadiendo un carácter y un color, así como en el ejemplo:

```
@startuml
```

```
class System << (S,#FF7700) Singleton >>
class Date << (D,orchid) >>
@enduml
```



3.21 Paquetes

Puedes definir un paquete usando la palabra reservada `package`, y opcionalmente declarar un color de fondo para tu paquete (Usando el nombre o el código HTML del color).

Tenga en cuenta que las definiciones de paquetes pueden ser anidadas.

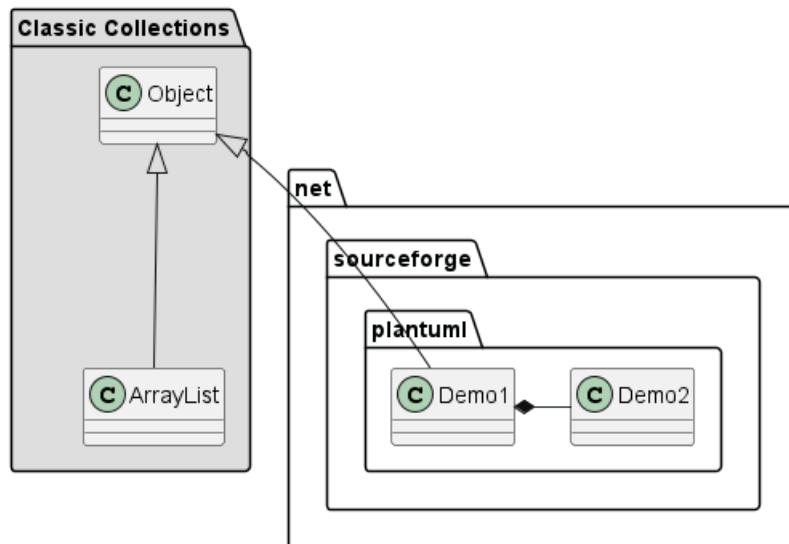
```
@startuml
```

```
package "Classic Collections" #DDDDDD {
    Object <|-- ArrayList
}
```

```
package net.sourceforge.plantuml {
    Object <|-- Demo1
    Demo1 *- Demo2
}
```

```
@enduml
```





3.22 Estilos de paquetes

Hay diferentes estilos disponibles para paquetes.

Puedes especificarlos, ya sea configurando un estilo por defecto con el comando : `skinparam packageStyle` , o usando un estereotipo en el paquete.

```

@startuml
scale 750 width
package foo1 <<Node>> {
    class Class1
}

package foo2 <<Rectangle>> {
    class Class2
}

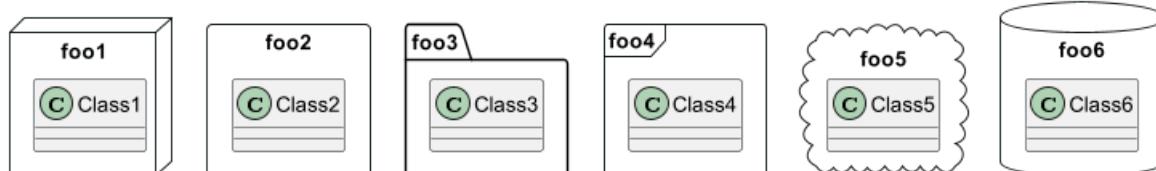
package foo3 <<Folder>> {
    class Class3
}

package foo4 <<Frame>> {
    class Class4
}

package foo5 <<Cloud>> {
    class Class5
}

package foo6 <<Database>> {
    class Class6
}

@enduml
  
```



Puedes también definir enlaces entre paquetes, como en el siguiente ejemplo:

```
@startuml

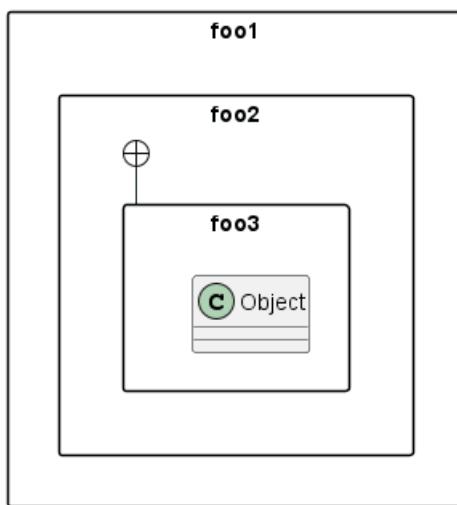
skinparam packageStyle rectangle

package foo1.foo2 {
}

package foo1.foo2.foo3 {
    class Object
}

foo1.foo2 +-- foo1.foo2.foo3

@enduml
```



3.23 Espacios de nombre

En los paquetes, el nombre de una clase es el único identificador de esta clase. Quiere decir que no puedes tener dos clases con el mismo nombre en diferentes paquetes.

En este caso, deberías usar espacios de nombres en lugar de paquetes.

Puedes referir a clases de otros espacios de nombre describiendo su ruta completamente. A clases del espacio de nombre por defecto son descritas colocando un punto al inicio.

Tenga en cuenta que no tiene que especificar explícitamente un espacio de nombre : una clase altamente clasificada es automáticamente colocada en el espacio de nombre correcto.

```
@startuml

class BaseClass

namespace net.dummy #DDDDDD {
    .BaseClass <|-- Person
    Meeting o-- Person

    .BaseClass <|- Meeting
}

namespace net.foo {
    net.dummy.Person <|- Person
    .BaseClass <|-- Person
  }
```



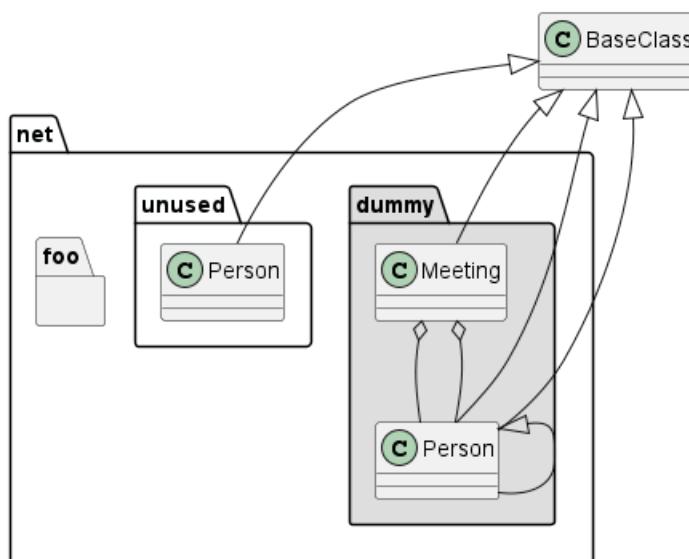
```

    net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person

@enduml

```



There won't be any difference between namespaces and packages anymore: both keywords are now synonymous.

3.24 Creación automática del espacio de nombre

Puedes definir otro separador (otro además del punto) usando el comando : `set namespaceSeparator ???.`

```
@startuml
```

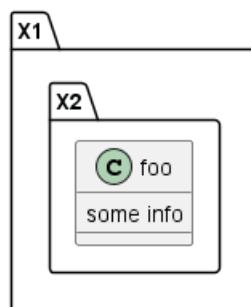
```

set namespaceSeparator ::

class X1::X2::foo {
    some info
}

```

```
@enduml
```



Puedes deshabilitar la creación automática de paquetes usando el comando `set namespaceSeparator none`.

```
@startuml
```

```
set namespaceSeparator none
```



```
class X1.X2.foo {
    some info
}

@enduml
```



3.25 Interface Lollipop

También puedes definir interfaces lollipops en clases, usando la siguiente sintaxis:

- bar ()- foo
- bar ()-- foo
- foo -(() bar

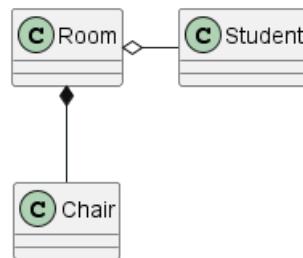
```
@startuml
class foo
bar ()- foo
@enduml
```



3.26 Cambiando la dirección de las flechas

Por defecto, enlaces entre clases tienen dos guiones -- y son verticalmente orientados. Es posible usar un enlace horizontal colocando un solo guión (o punto), así:

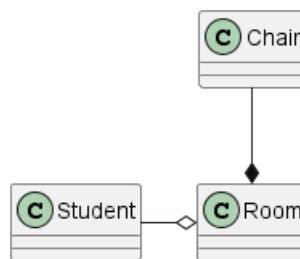
```
@startuml
Room o- Student
Room *-- Chair
@enduml
```



También puedes cambiar las direcciones revirtiendo el enlace:

```
@startuml
Student -o Room
Chair --* Room
@enduml
```

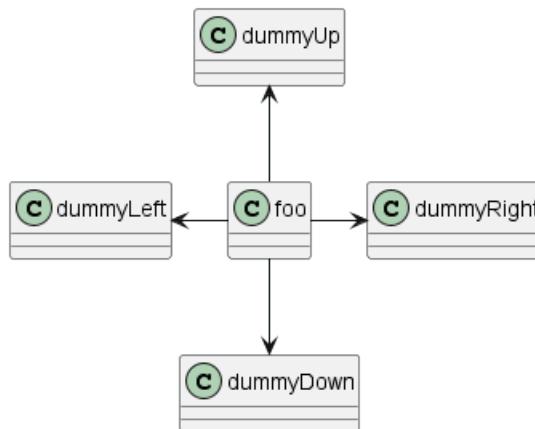




También es posible cambiar la dirección de la flecha añadiendo las palabras claves `left`, `right`, `up` or `down` dentro de la flecha:

```

@startuml
foo -left-> dummyLeft
foo -right-> dummyRight
foo -up-> dummyUp
foo -down-> dummyDown
@enduml
  
```



Puedes acortar la flecha usando el primer carácter de la dirección (por ejemplo, `-d-` en lugar de `-down-`) o los primeros dos caracteres.

Por favor tenga en cuenta que no debería abusar de esta funcionalidad : *Graphviz* usualmente otorga buenos resultados sin necesidad de ajustar.

3.27 Asociación de clases

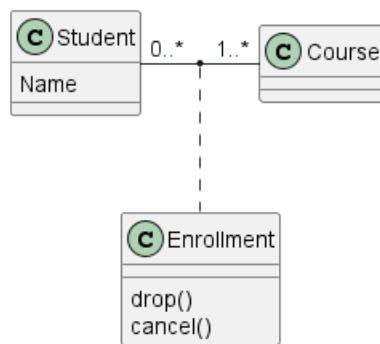
Puedes definir *association class* después de que una relación haya sido establecida entre dos clases, como en este ejemplo:

```

@startuml
class Student {
    Name
}
Student "0..*" - "1..*" Course
(Student, Course) .. Enrollment

class Enrollment {
    drop()
    cancel()
}
@enduml
  
```





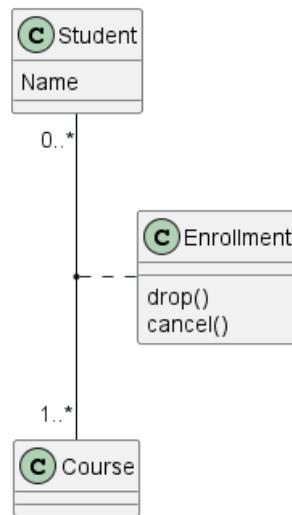
Puedes definirla en otra dirección:

```

@startuml
class Student {
    Name
}
Student "0..*" -- "1..*" Course
(Student, Course) . Enrollment

class Enrollment {
    drop()
    cancel()
}
@enduml

```



3.28 Association on same class

```

@startuml
class Station {
    +name: string
}

class StationCrossing {
    +cost: TimeInterval
}

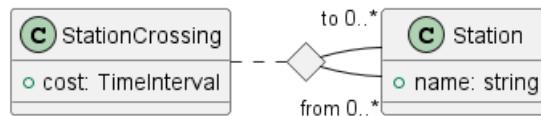
<> diamond

StationCrossing . diamond
diamond - "from 0..*" Station

```



```
diamond - "to 0..*" Station
@enduml
```



[Ref. Incubation: Associations]

3.29 Personalización (Skinparam)

Puedes usar el comando skinparam para cambiar los colores y fuentes en el diagrama.

Puedes usar este comando:

- En la definición del diagrama, como cualquier otro comando,
- En un archivo incluido,
- En un archivo de configuración, proporcionado en la consola de comandos o en el ANT task.

```
@startuml
```

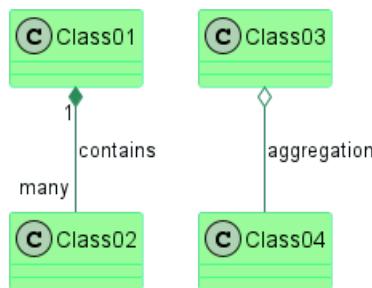
```

skinparam class {
    BackgroundColor PaleGreen
    ArrowColor SeaGreen
    BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen

Class01 "1" *-- "many" Class02 : contains
Class03 o-- Class04 : aggregation

```

```
@enduml
```



3.30 Estereotipos personalizados

Puedes definir colores y fuentes específicas para clases esterotipadas.

```
@startuml
```

```

skinparam class {
    BackgroundColor PaleGreen
    ArrowColor SeaGreen
    BorderColor SpringGreen
    BackgroundColor<<Foo>> Wheat
    BorderColor<<Foo>> Tomato
}
skinparam stereotypeCBackgroundColor YellowGreen
skinparam stereotypeCBackgroundColor<< Foo >> DimGray

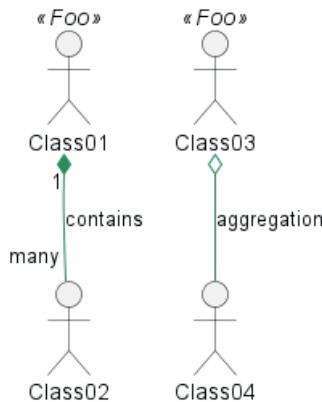
```



```
Class01 <<Foo>>
Class03 <<Foo>>
Class01 "1" *-- "many" Class02 : contains
```

```
Class03 o-- Class04 : aggregation
```

```
@enduml
```



3.31 Degrado de colores

Es posible declarar colores individuales para clases o notas usando la notación #.

Puedes usar el nombre estándar del color o el código RGB.

También puedes usar degradación de color en el fondo, con la siguiente sintaxis: dos nombres de colores separados por cualquier de los siguientes:

- |,
- /,
- \,
- o -

dependiendo de la dirección del degradado.

Por ejemplo, podrías tener:

```
@startuml
```

```

skinparam backgroundColor AntiqueWhite/Gold
skinparam classBackgroundColor Wheat|CornflowerBlue

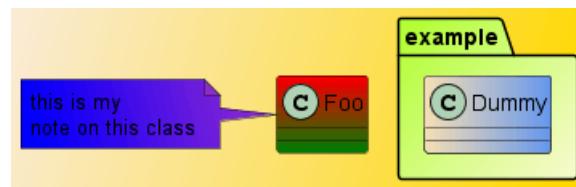
class Foo #red-green
note left of Foo #blue\9932CC
    this is my
    note on this class
end note

package example #GreenYellow/LightGoldenRodYellow {
    class Dummy
}

@enduml

```





3.32 Ayudar en el diseño

Sometimes, the default layout is not perfect...

You can use `together` keyword to group some classes together : the layout engine will try to group them (as if they were in the same package).

You can also use `hidden` links to force the layout.

```
@startuml
```

```
class Bar1
class Bar2
together {
    class Together1
    class Together2
    class Together3
}
Together1 - Together2
Together2 - Together3
Together2 -[hidden]--> Bar1
Bar1 -[hidden]> Bar2
```

```
@enduml
```



3.33 Dividiendo archivos grandes

A veces, puedes obtener imágenes bastante grandes.

Puedes usar el comando `page (hpages)x(vpages)` para dividir la imagen generada, en varias imágenes: `hpages` es el número que indica la cantidad de páginas horizontales, y `vpages` es el número que indica la cantidad de páginas verticales.

También puede utilizar algunos ajustes específicos `skinparam` poner fronteras en las páginas splitted (ver ejemplo).

```
@startuml
' Split into 4 pages
page 2x2
skinparam pageMargin 10
skinparam pageExternalColor gray
```



```

skinparam pageBorderColor black

class BaseClass

namespace net.dummy #DDDDDD {
    .BaseClass <|-- Person
    Meeting o-- Person

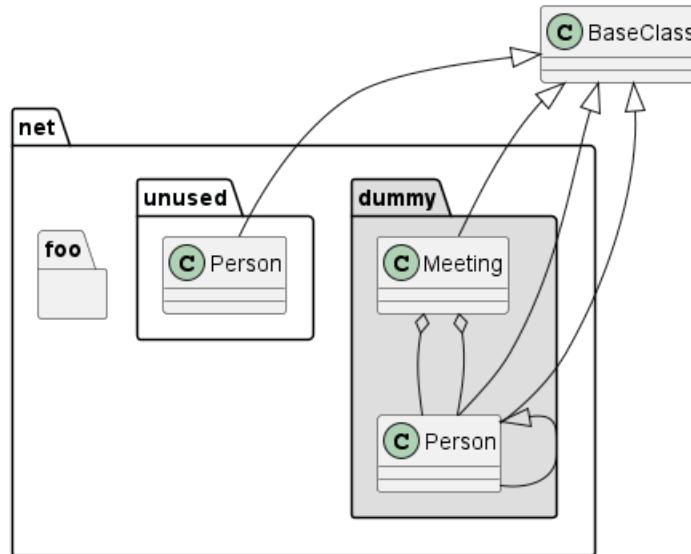
    .BaseClass <|- Meeting
}

namespace net.foo {
    net.dummy.Person <|- Person
    .BaseClass <|-- Person

    net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person
@enduml

```



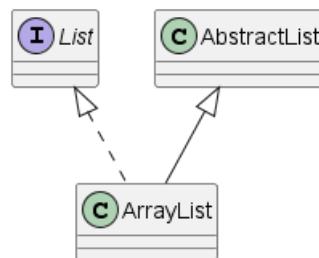
3.34 Extends and implements

It is also possible to use `extends` and `implements` keywords.

```

@startuml
class ArrayList implements List
class ArrayList extends AbstractList
@enduml

```



3.35 Bracketed relations (linking or arrow) style

3.35.1 Line style

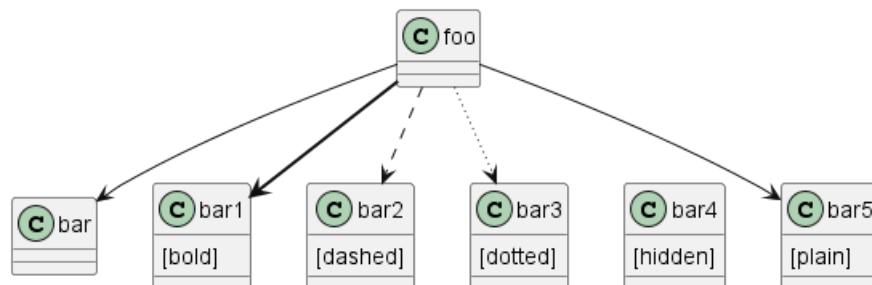
It's also possible to have explicitly **bold**, **dashed**, **dotted**, **hidden** or **plain** relation, links or arrows:

- without label

```
@startuml
title Bracketed line style without label
class foo
class bar
bar1 : [bold]
bar2 : [dashed]
bar3 : [dotted]
bar4 : [hidden]
bar5 : [plain]

foo --> bar
foo -[bold]-> bar1
foo -[dashed]-> bar2
foo -[dotted]-> bar3
foo -[hidden]-> bar4
foo -[plain]-> bar5
@enduml
```

Bracketed line style without label



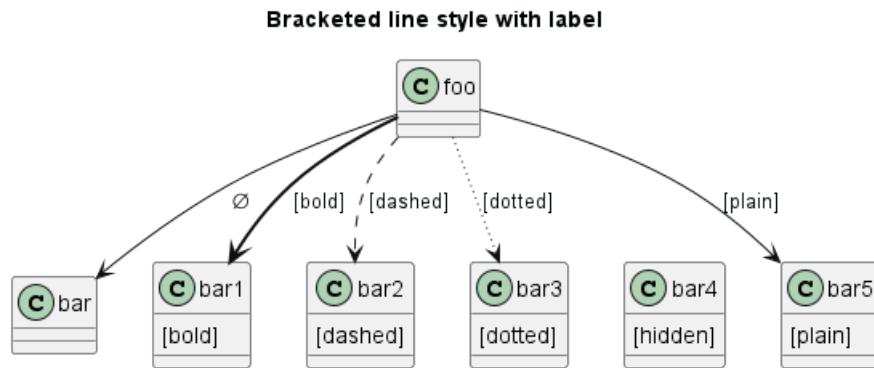
- with label

```
@startuml
title Bracketed line style with label
class foo
class bar
bar1 : [bold]
bar2 : [dashed]
bar3 : [dotted]
bar4 : [hidden]
bar5 : [plain]

foo --> bar      :
foo -[bold]-> bar1 : [bold]
foo -[dashed]-> bar2 : [dashed]
foo -[dotted]-> bar3 : [dotted]
foo -[hidden]-> bar4 : [hidden]
foo -[plain]-> bar5 : [plain]

@enduml
```





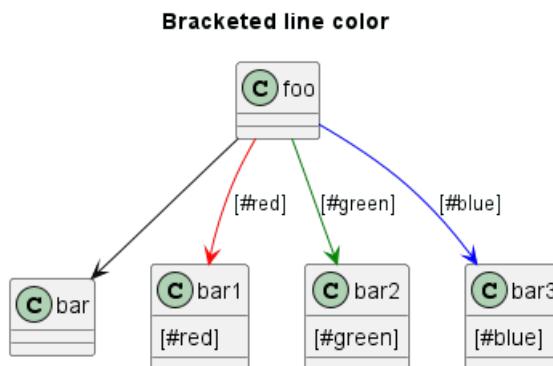
[Adapted from QA-4181]

3.35.2 Line color

```

@startuml
title Bracketed line color
class foo
class bar
bar1 : [#red]
bar2 : [#green]
bar3 : [#blue]

foo --> bar
foo -[#red]-> bar1 : [#red]
foo -[#green]-> bar2 : [#green]
foo -[#blue]-> bar3 : [#blue]
'foo -[#blue;#yellow;#green]-> bar4
@enduml
  
```



3.35.3 Line thickness

```

@startuml
title Bracketed line thickness
class foo
class bar
bar1 : [thickness=1]
bar2 : [thickness=2]
bar3 : [thickness=4]
bar4 : [thickness=8]
bar5 : [thickness=16]

foo --> bar
foo -[thickness=1]-> bar1 : [1]
  
```

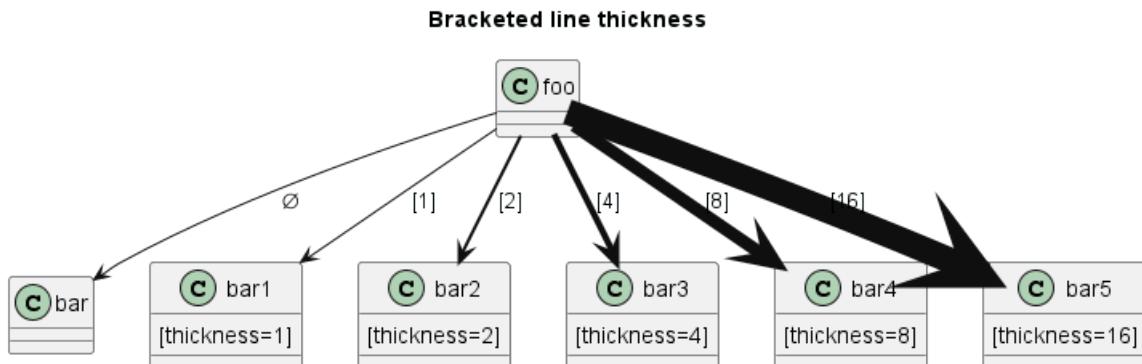


```

foo -[thickness=2]-> bar2 : [2]
foo -[thickness=4]-> bar3 : [4]
foo -[thickness=8]-> bar4 : [8]
foo -[thickness=16]-> bar5 : [16]

```

@enduml



[Ref. QA-4949]

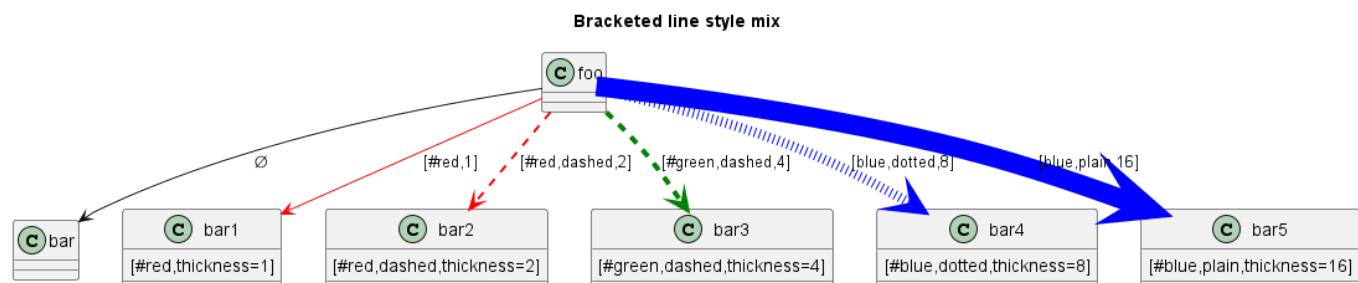
3.35.4 Mix

```

@startuml
title Bracketed line style mix
class foo
class bar
bar1 : [#red,thickness=1]
bar2 : [#red,dashed,thickness=2]
bar3 : [#green,dashed,thickness=4]
bar4 : [#blue,dotted,thickness=8]
bar5 : [#blue,plain,thickness=16]

foo --> bar : 
foo -[#red,thickness=1]-> bar1 : [#red,1]
foo -[#red,dashed,thickness=2]-> bar2 : [#red,dashed,2]
foo -[#green,dashed,thickness=4]-> bar3 : [#green,dashed,4]
foo -[#blue,dotted,thickness=8]-> bar4 : [blue,dotted,8]
foo -[#blue,plain,thickness=16]-> bar5 : [blue,plain,16]
@enduml

```



3.36 Change relation (linking or arrow) color and style (inline style)

You can change the color or style of individual relation or arrows using the inline following notation:

- #color;line.[bold|dashed|dotted];text:color

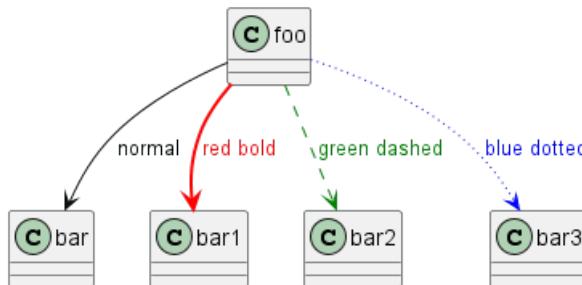
@startuml



```

class foo
foo --> bar : normal
foo --> bar1 #line:red;line.bold;text:red : red bold
foo --> bar2 #green;line.dashed;text:green : green dashed
foo --> bar3 #blue;line.dotted;text:blue : blue dotted
@enduml

```



[See similar feature on deployment]

3.37 Change class color and style (inline style)

You can change the color or style of individual class using the two following notations:

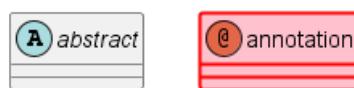
- **#color ##[style]color**

With background color first (**#color**), then line style and line color (**##[style]color**)

```

@startuml
abstract abstract
annotation annotation #pink ##[bold]red
class class      #palegreen ##[dashed]green
interface interface #aliceblue ##[dotted]blue
@enduml

```



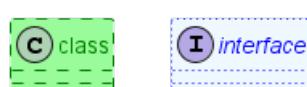
[Ref. QA-1487]

- **# [color|back:color];header:color;line:color;line. [bold|dashed|dotted];text:color**

```

@startuml
abstract abstract
annotation annotation #pink;line:red;line.bold;text:red
class class      #palegreen;line:green;line.dashed;text:green
interface interface #aliceblue;line:blue;line.dotted;text:blue
@enduml

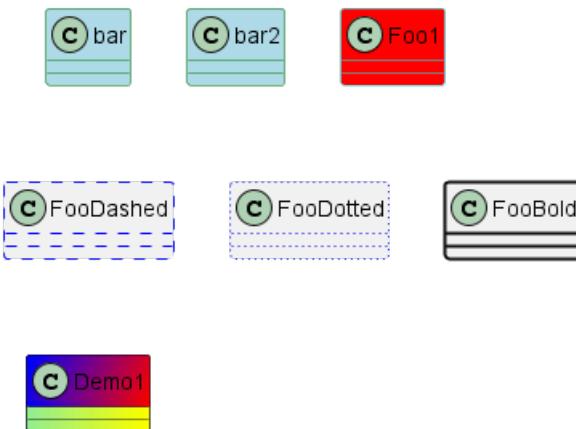
```



First original example:

```
@startuml
class bar #line:green;back:lightblue
class bar2 #lightblue;line:green

class Foo1 #back:red;line:00FFFF
class FooDashed #line.dashed:blue
class FooDotted #line.dotted:blue
class FooBold #line.bold
class Demo1 #back:lightgreen|yellow;header:blue/red
@enduml
```



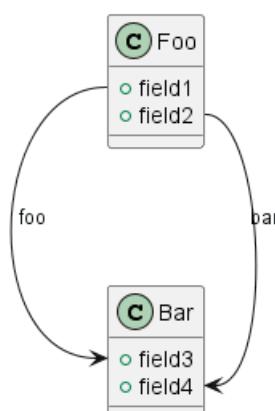
[Ref. QA-3770]

3.38 Arrows from/to class members

```
@startuml
class Foo {
+ field1
+ field2
}

class Bar {
+ field3
+ field4
}

Foo::field1 --> Bar::field3 : foo
Foo::field2 --> Bar::field4 : bar
@enduml
```



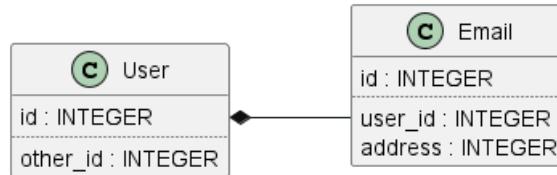
[Ref. QA-3636]

```
@startuml
left to right direction
```

```
class User {
    id : INTEGER
    ..
    other_id : INTEGER
}

class Email {
    id : INTEGER
    ..
    user_id : INTEGER
    address : INTEGER
}

User::id *-- Email::user_id
@enduml
```



[Ref. QA-5261]

3.39 Grouping inheritance arrow heads

You can merge all arrow heads using the `skinparam groupInheritance 1`, with a threshold as parameter.

3.39.1 GroupInheritance 1 (no grouping)

```
@startuml
skinparam groupInheritance 1

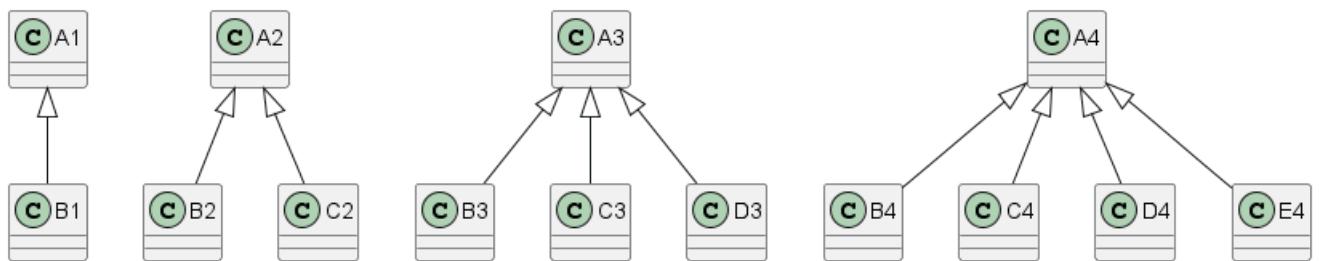
A1 <|-- B1

A2 <|-- B2
A2 <|-- C2

A3 <|-- B3
A3 <|-- C3
A3 <|-- D3

A4 <|-- B4
A4 <|-- C4
A4 <|-- D4
A4 <|-- E4
@enduml
```





3.39.2 GroupInheritance 2 (grouping from 2)

```
@startuml
skinparam groupInheritance 2
```

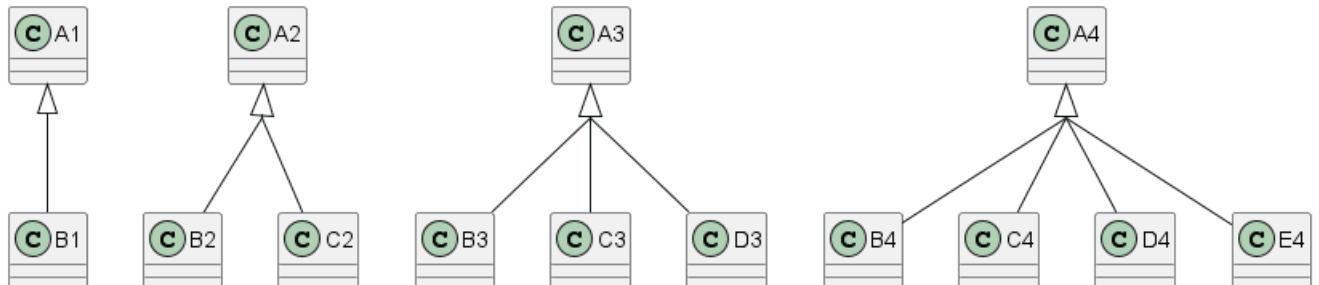
A1 <|-- B1

A2 <|-- B2
A2 <|-- C2

A3 <|-- B3
A3 <|-- C3
A3 <|-- D3

A4 <|-- B4
A4 <|-- C4
A4 <|-- D4
A4 <|-- E4

```
@enduml
```



3.39.3 GroupInheritance 3 (grouping only from 3)

```
@startuml
skinparam groupInheritance 3
```

A1 <|-- B1

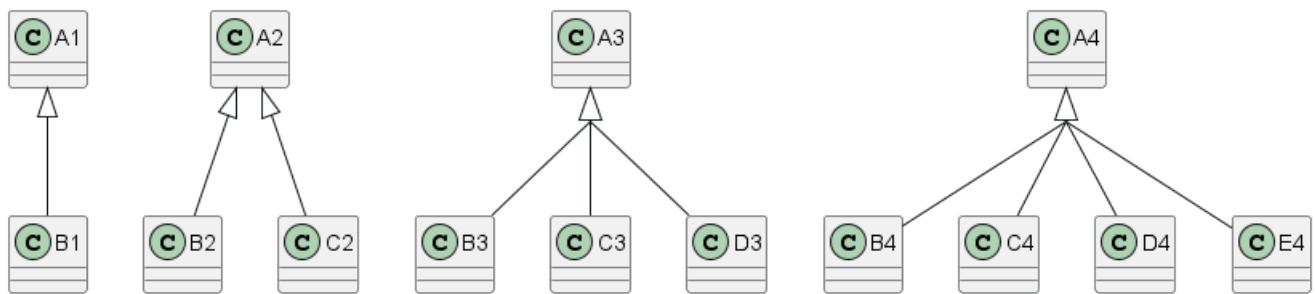
A2 <|-- B2
A2 <|-- C2

A3 <|-- B3
A3 <|-- C3
A3 <|-- D3

A4 <|-- B4
A4 <|-- C4
A4 <|-- D4
A4 <|-- E4



@enduml

**3.39.4 GroupInheritance 4 (grouping only from 4)**

@startuml

skinparam groupInheritance 4

A1 <|-- B1

A2 <|-- B2

A2 <|-- C2

A3 <|-- B3

A3 <|-- C3

A3 <|-- D3

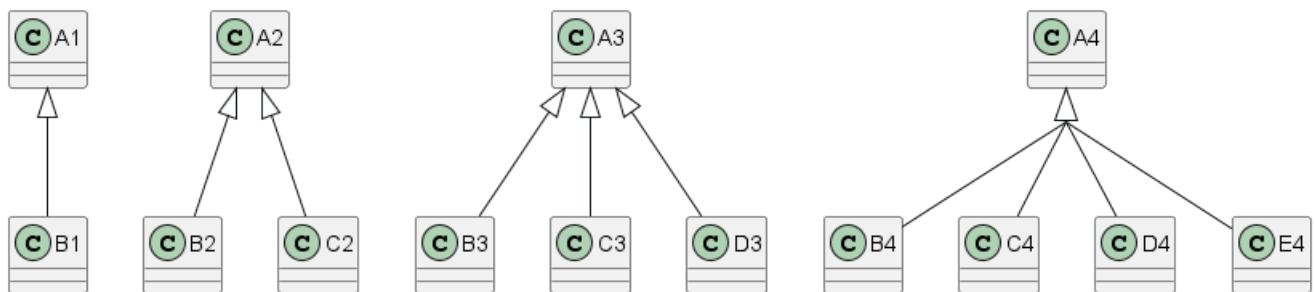
A4 <|-- B4

A4 <|-- C4

A4 <|-- D4

A4 <|-- E4

@enduml



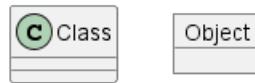
[Ref. QA-3193, and Defect QA-13532]

3.40 Display JSON Data on Class or Object diagram**3.40.1 Simple example**

```

@startuml
class Class
object Object
json JSON {
    "fruit":"Apple",
    "size":"Large",
    "color": ["Red", "Green"]
}
@enduml
  
```





JSON	
fruit	Apple
size	Large
color	Red
	Green

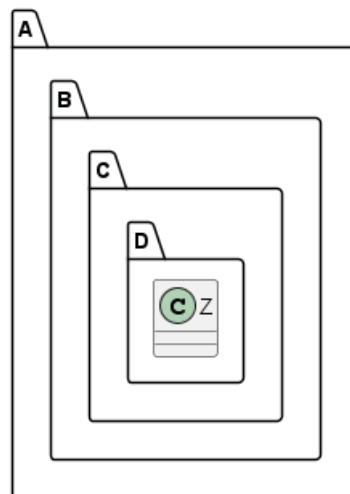
[Ref. QA-15481]

For another example, see on JSON page.

3.41 Packages and Namespaces Enhancement

[From V1.2023.2+, and V1.2023.5]

```
@startuml
class A.B.C.D.Z {
}
@enduml
```

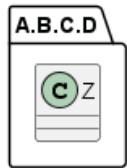


```
@startuml
set separator none
class A.B.C.D.Z {
}
@enduml
```

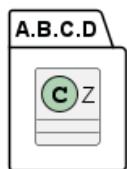


```
@startuml
!pragma useIntermediatePackages false
class A.B.C.D.Z {
}
@enduml
```





```
@startuml  
set separator none  
package A.B.C.D {  
    class Z {  
    }  
}  
@enduml
```



[Ref. GH-1352]



4 Diagrama de Objetos

PlantUML offers a simple and intuitive way to create object diagrams using plain text. Its user-friendly syntax allows for quick diagram creation without the need for complex GUI tools. Moreover, the [PlantUML forum](<https://forum.plantuml.net/>) provides a platform for users to discuss, share, and seek assistance, fostering a collaborative community. By choosing PlantUML, users benefit from both the efficiency of markdown-based diagramming and the support of an active community.

4.1 Definición de objetos

Puedes definir instancias de objetos usando la palabra reservada `object`.

```
@startuml
object firstObject
object "My Second Object" as o2
@enduml
```



4.2 Relaciones entre objetos

Las relaciones entre objetos son definidas usando los siguientes símbolos:

Type	Symbol	Image
Extension	< --	
Composition	*--	
Aggregation	o--	

Es posible reemplazar `--` con `...` para obtener una línea de puntos.

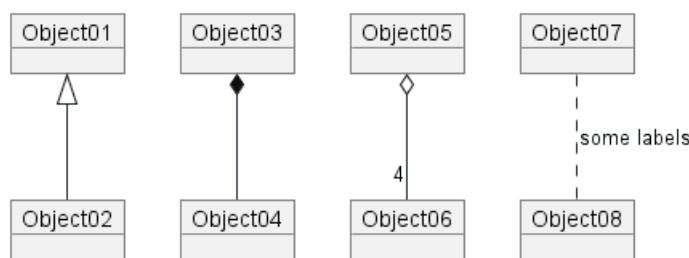
Sabiendo estas reglas, es posible dibujar los siguientes diagramas.

Es posible agregar una etiqueta sobre una relación usando `:`, seguido del texto de la etiqueta.

Para la cardinalidad puedes usar doble comillas `" "` en cada lado de la relación.

```
@startuml
object Object01
object Object02
object Object03
object Object04
object Object05
object Object06
object Object07
object Object08

Object01 <|-- Object02
Object03 *-- Object04
Object05 o-- "4" Object06
Object07 ... Object08 : some labels
@enduml
```



4.3 Asociaciones de objetos

```
@startuml
object o1
object o2
diamond dia
object o3

o1 --> dia
o2 --> dia
dia --> o3
@enduml
```



4.4 Agregando campos

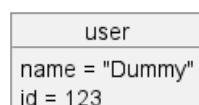
Para declarar campos, puedes usar el símbolo : seguido del nombre del campo.

```
@startuml

object user

user : name = "Dummy"
user : id = 123

@enduml
```

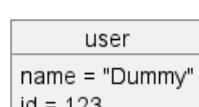


También es posible declarar entre llaves {} todos los campos.

```
@startuml

object user {
    name = "Dummy"
    id = 123
}

@enduml
```



4.5 Características comunes en diagramas de clases

- Visibilidad
- Definición de notas
- Uso de paquetes
- Personalización de la salida (skinparam)

4.6 Tabla o arreglo asociativo

Puede definir la "tabla" o associative array, con la palabra `map` y utilizando `=>` como separador.

```
@startuml
map CapitalCity {
    UK => London
    USA => Washington
    Germany => Berlin
}
@enduml
```

CapitalCity	
UK	London
USA	Washington
Germany	Berlin

```
@startuml
map "Map **Country => CapitalCity**" as CC {
    UK => London
    USA => Washington
    Germany => Berlin
}
@enduml
```

Map Country => CapitalCity	
UK	London
USA	Washington
Germany	Berlin

```
@startuml
map "map: Map<Integer, String>" as users {
    1 => Alice
    2 => Bob
    3 => Charlie
}
@enduml
```

map: Map<Integer, String>	
1	Alice
2	Bob
3	Charlie

And add link with object.

```
@startuml
object London

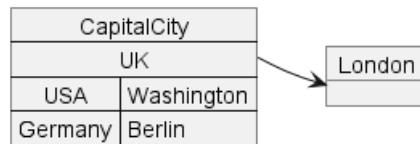
map CapitalCity {
    UK *-> London
```



```

USA => Washington
Germany => Berlin
}
@enduml

```



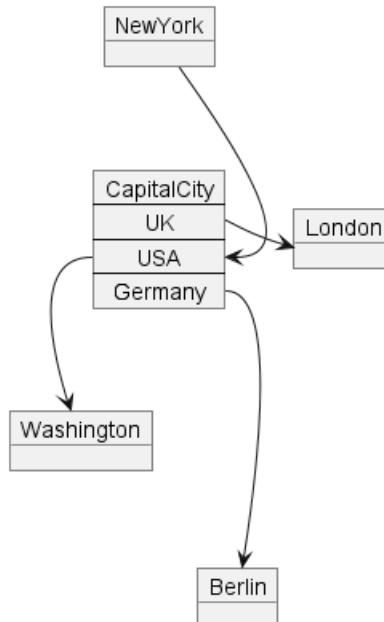
```

@startuml
object London
object Washington
object Berlin
object NewYork

map CapitalCity {
    UK *-> London
    USA *--> Washington
    Germany *---> Berlin
}

NewYork --> CapitalCity::USA
@enduml

```



[Ref. #307]

```

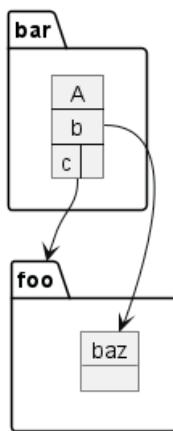
@startuml
package foo {
    object baz
}

package bar {
    map A {
        b *-> foo.baz
        c =>
    }
}

```



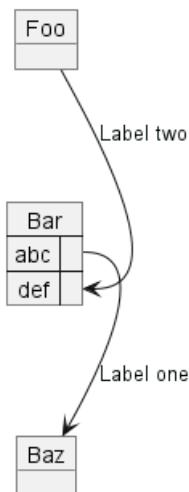
```
A::c --> foo
@enduml
```



[Ref. QA-12934]

```
@startuml
object Foo
map Bar {
    abc=>
    def=>
}
object Baz

Bar::abc --> Baz : Label one
Foo --> Bar::def : Label two
@enduml
```



[Ref. #307]

4.7 Program (or project) evaluation and review technique (PERT) with map

You can use map table in order to make Program (or project) evaluation and review technique (PERT) diagram.

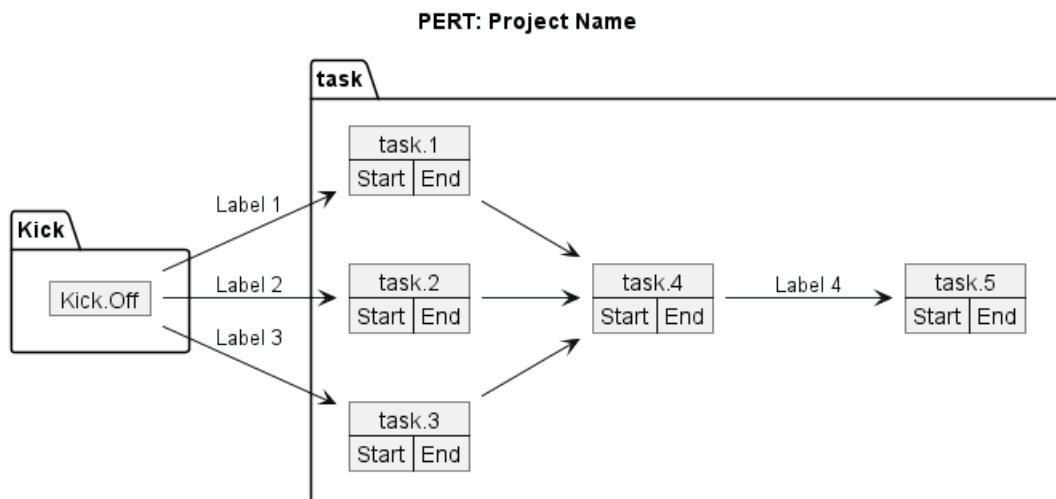
```
@startuml PERT
left to right direction
' Horizontal lines: -->, <--, <-->
' Vertical lines: ->, <-, <->
title PERT: Project Name
```



```

map Kick.Off {
}
map task.1 {
    Start => End
}
map task.2 {
    Start => End
}
map task.3 {
    Start => End
}
map task.4 {
    Start => End
}
map task.5 {
    Start => End
}
}
Kick.Off --> task.1 : Label 1
Kick.Off --> task.2 : Label 2
Kick.Off --> task.3 : Label 3
task.1 --> task.4
task.2 --> task.4
task.3 --> task.4
task.4 --> task.5 : Label 4
@enduml

```



[Ref. QA-12337]

4.8 Display JSON Data on Class or Object diagram

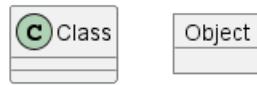
4.8.1 Simple example

```

@startuml
class Class
object Object
json JSON {
    "fruit":"Apple",
    "size":"Large",
    "color": ["Red", "Green"]
}
@enduml

```





JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

For another example, see on JSON page.



5 Diagrama de Actividades

5.1 Actividades simples

Puedes usar (*) para el punto de inicio y de final en un diagrama de actividades.

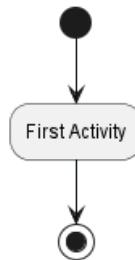
En algunos casos, quizás quieras usar (*top) para forzar a que el punto de inicio se ubique en la parte superior del diagrama.

Utilice --> para las flechas.

```
@startuml
```

```
(*) --> "First Activity"
"First Activity" --> (*)
```

```
@enduml
```



5.2 Etiquetas en las flechas

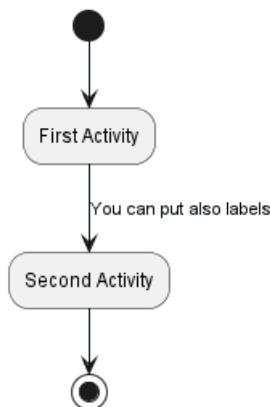
Por defecto, una flecha comienza en la última actividad usada.

Puedes colocar una etiqueta sobre una flecha usando corchetes, [], justo después de la definición de la flecha.

```
@startuml
```

```
(*) --> "First Activity"
-->[You can put also labels] "Second Activity"
--> (*)
```

```
@enduml
```



5.3 Cambiando la dirección de la flecha

Puedes usar -> para flechas horizontales. Es posible formzar la dirección de una flecha usando la siguiente sintaxis:

- -down-> (default arrow)

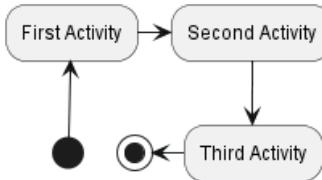


- -right-> or ->
- -left->
- -up->

@startuml

```
(*) -up-> "First Activity"
-right-> "Second Activity"
--> "Third Activity"
-left-> (*)
```

@enduml



5.4 Ramas (bifurcaciones)

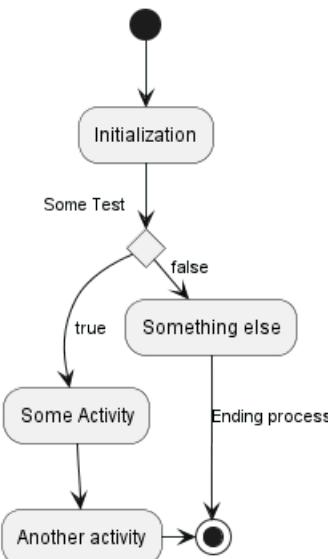
Puedes usar las palabras reservadas `if/then/else` para definir ramas o caminos alternos.

@startuml

```
(*) --> "Initialization"

if "Some Test" then
    -->[true] "Some Activity"
    --> "Another activity"
    -right-> (*)
else
    ->[false] "Something else"
    -->[Ending process] (*)
endif
```

@enduml



Desafortunadamente, a veces tendrás que repetir la misma actividad en el texto del diagrama:

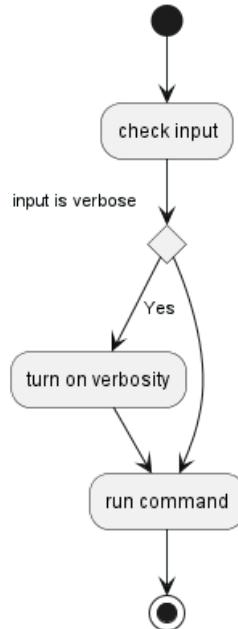
@startuml



```

(*) --> "check input"
If "input is verbose" then
--> [Yes] "turn on verbosity"
--> "run command"
else
--> "run command"
Endif
-->(*)
@enduml

```



5.5 Más acerca de las Ramas

Por defecto, una rama es conectada con la última actividad definida, pero es posible sobrescribir esto y definir un enlace con la palabra reservada `if` .

También es posible anidar ramas.

```

@startuml

(*) --> if "Some Test" then

    -->[true] "activity 1"

    if "" then
        -> "activity 3" as a3
    else
        if "Other test" then
            -left-> "activity 5"
        else
            --> "activity 6"
        endif
    endif

else

    ->[false] "activity 2"

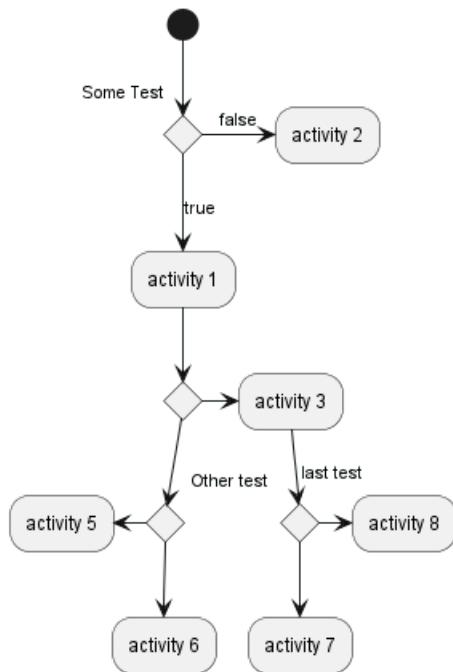
endif

```



```
a3 --> if "last test" then
    --> "activity 7"
else
    -> "activity 8"
endif

@enduml
```



5.6 Sincronización

Puedes usar === code === para mostrar barras de sincronización.

```
@startuml
```

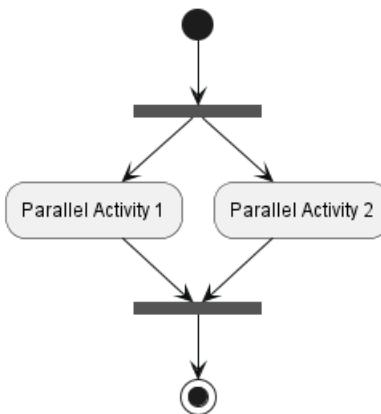
```
(*) --> ===B1===
--> "Parallel Activity 1"
--> ===B2===

==B1== --> "Parallel Activity 2"
--> ==B2==

--> (*)
```

```
@enduml
```





5.7 Descripción de actividades de gran contenido

Cuando declaras actividades, puedes abarcar la descripción del texto, en varias líneas. También puedes añadir en la descripción.

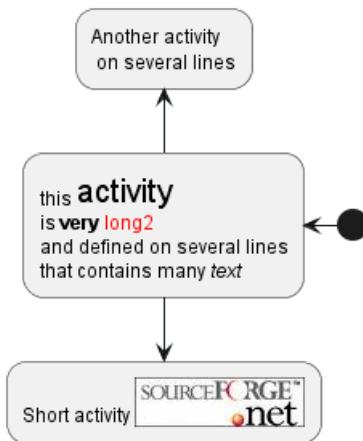
También puedes colocar una pequeña cantidad de código en la actividad, con la palabra reservada `as`. Este código puede usarse más adelante en la descripción del diagrama.

```

@startuml
(*) -left-> "this <size:20>activity</size>
is <b>very</b> <color:red>long2</color>
and defined on several lines
that contains many <i>text</i>" as A1

-up-> "Another activity\n on several lines"

A1 --> "Short activity <img:sourceforge.jpg>"
@enduml
  
```



5.8 Notas

Puedes añadir notas sobre la actividad usando los comandos `note left`, `note right`, `note top` or `note bottom`, justo después de la descripción de la actividad a la que quieras añadirle la nota.

Si quieres colocar una nota sobre el punto de inicio, define la nota al comienzo de la descripción del diagrama.

También puedes tener una nota de varias líneas, usando la palabra reservada `endnote`.

```
@startuml
```

```
(*) --> "Some Activity"
```

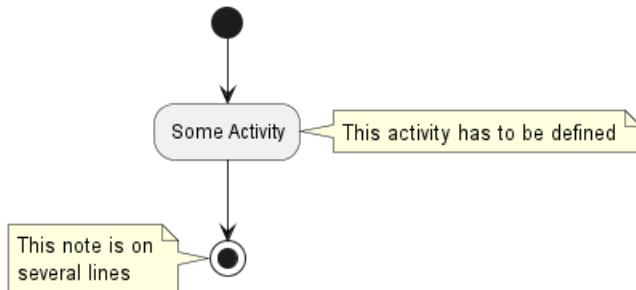


```

note right: This activity has to be defined
"Some Activity" --> (*)
note left
  This note is on
  several lines
end note

@enduml

```



5.9 Partición

Puedes definir una partición usando la palabra reservada `partition` y opcionalmente declarar un color de fondo para tu partición (Usando el nombre o código HTML del color)

Cuando declaras actividades, éstas son automáticamente colocadas en la última partición usada.

Puedes cerrar la partición usando una llave de cierre `}`.

```
@startuml
```

```

partition Conductor {
    (*) --> "Climbs on Platform"
    --> === S1 ===
    --> Bows
}

partition Audience #LightSkyBlue {
    === S1 === --> Applauds
}

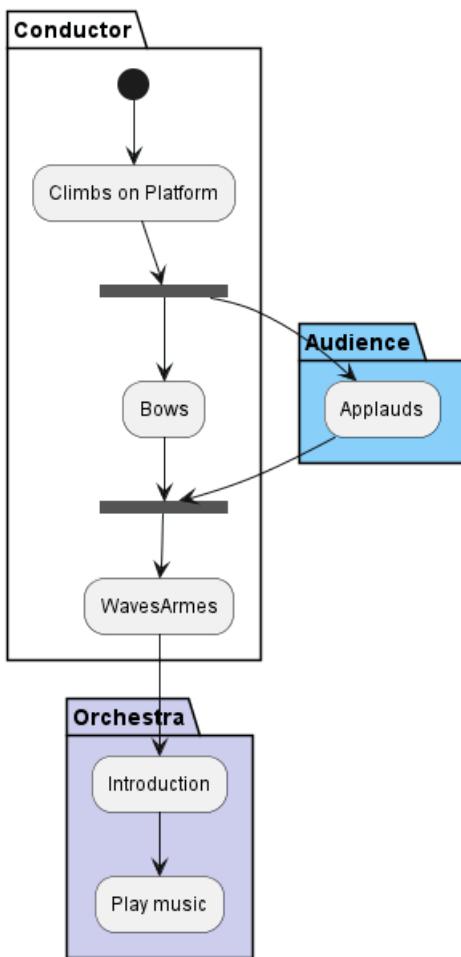
partition Conductor {
    Bows --> === S2 ===
    --> WavesArmes
    Applauds --> === S2 ===
}

partition Orchestra #CCCCEE {
    WavesArmes --> Introduction
    --> "Play music"
}

```

```
@enduml
```





5.10 Personalización (Skinparam)

Puedes usar el comando `skinparam` para cambiar las fuentes y colores en el diagrama.

Puedes usar este comando :

- En la definición del diagrama, como cualquier otro comando,
- En un archivo incluido,
- En un archivo de configuración, proporcionado en la consola de comandos o en el ANT task.

Puedes definir colores y fuentes específicas para actividades estereotipadas.

`@startuml`

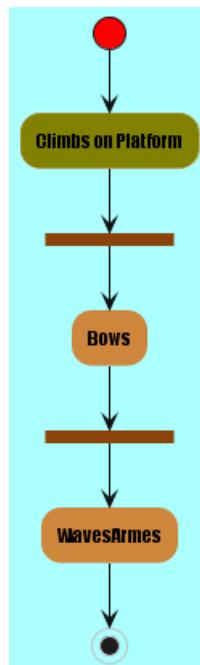
```

skinparam backgroundColor #AAFFFF
skinparam activity {
    StartColor red
    BarColor SaddleBrown
    EndColor Silver
    BackgroundColor Peru
    BackgroundColor<< Begin >> Olive
    BorderColor Peru
    FontName Impact
}
(*) --> "Climbs on Platform" << Begin >>
--> === S1 ===
--> Bows
  
```



```
--> === S2 ===
--> WavesArmes
--> (*)
```

@enduml



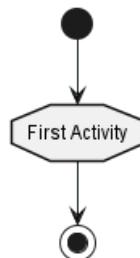
5.11 Octágono

Puedes cambiar la forma de las actividades a un octágono usando el comando `skinparam activityShape octagon`.

```
@startuml
'Default is skinparam activityShape roundBox
skinparam activityShape octagon

(*) --> "First Activity"
"First Activity" --> (*)
```

@enduml



5.12 Ejemplo completo

```
@startuml
title Servlet Container

(*) --> "ClickServlet.handleRequest()"
--> "new Page"

if "Page.onSecurityCheck" then
```



```
->[true] "Page.onInit()"

if "isForward?" then
->[no] "Process controls"

if "continue processing?" then
-->[yes] ===RENDERING===
else
-->[no] ===REDIRECT_CHECK===
endif

else
-->[yes] ===RENDERING===
endif

if "is Post?" then
-->[yes] "Page.onPost()"
--> "Page.onRender()" as render
--> ===REDIRECT_CHECK===
else
-->[no] "Page.onGet()"
--> render
endif

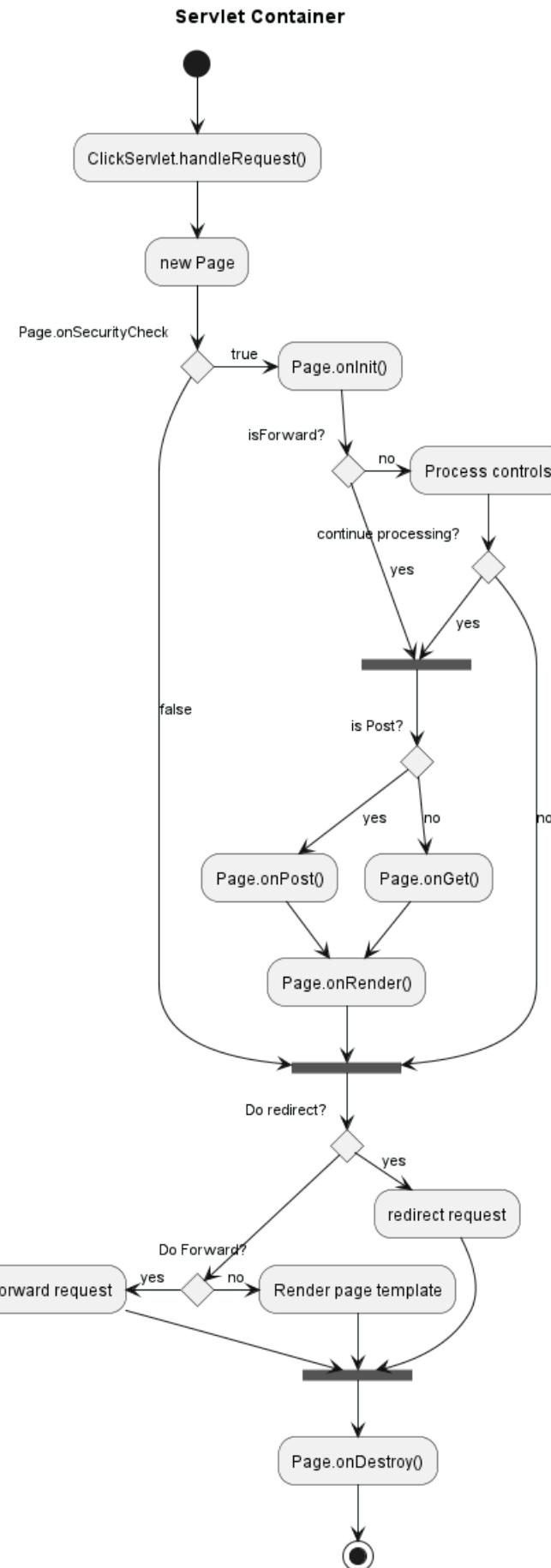
else
-->[false] ===REDIRECT_CHECK===
endif

if "Do redirect?" then
->[yes] "redirect request"
--> ==BEFORE_DESTROY===
else
if "Do Forward?" then
-left->[yes] "Forward request"
--> ==BEFORE_DESTROY===
else
-right->[no] "Render page template"
--> ==BEFORE_DESTROY===
endif
endif

--> "Page.onDestroy()"
-->(*)
```

@enduml





6 Diagrama de actividad (nueva sintaxis)

La sintaxis anterior utilizada para los diagramas de actividad presentaba varias limitaciones y problemas de mantenimiento. Reconociendo estos inconvenientes, hemos introducido una sintaxis e implementación totalmente renovadas que no sólo son fáciles de usar, sino también más estables.

6.0.1 Ventajas de la nueva sintaxis

- Sin dependencia de Graphviz: Al igual que con los diagramas de secuencia, la nueva sintaxis elimina la necesidad de instalar Graphviz, simplificando así el proceso de configuración.
- Facilidad de mantenimiento: La* naturaleza intuitiva de la nueva sintaxis significa que es más fácil de manejar y mantener sus diagramas.

6.0.2 Transición a la Nueva Sintaxis

Mientras que continuaremos apoyando la sintaxis antigua para mantener la compatibilidad, animamos a los usuarios a migrar a la nueva sintaxis para aprovechar las características mejoradas y los beneficios que ofrece.

Haga el cambio hoy y experimente un proceso de diagramación más ágil y eficiente con la nueva sintaxis de diagrama de actividad.

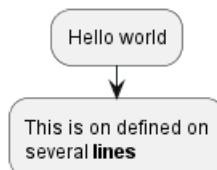
6.1 Una Actividad simple

Las etiquetas de las actividades inician con un dos puntos (:) y terminan con un punto y coma (;).

Se puede aplicar formato a un texto usando sintaxis de WikiCreole.

Están implícitamente enlazados en el orden de su definición.

```
@startuml
:Hello world;
:This is on defined on
several **lines**;
@enduml
```

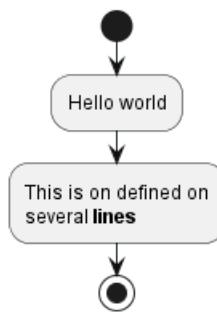


6.2 Start/Stop

Puedes usar las palabras reservadas **start** y **stop** para denotar el comienzo y el final del diagrama.

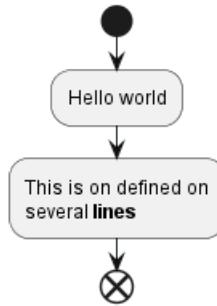
```
@startuml
start
:Hello world;
:This is on defined on
several **lines**;
stop
@enduml
```





También puedes usar la palabra reservada `end`.

```
@startuml
start
:Hello world;
:This is on defined on
several **lines**;
end
@enduml
```



6.3 Condicionales

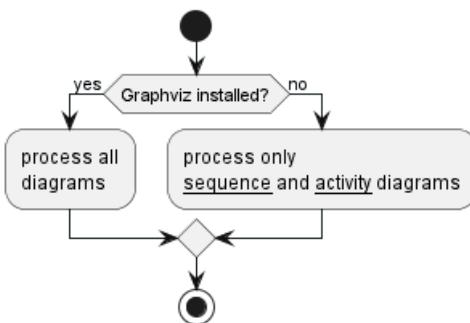
Puedes usar las palabras reservadas `if`, `then` y `else` para colocar condiciones en tus diagramas. Las etiquetas pueden ser proporcionadas usando paréntesis.

```
@startuml
start

if (Graphviz installed?) then (yes)
  :process all\ndiagrams;
else (no)
  :process only
  __sequence__ and __activity__ diagrams;
endif

stop
@enduml
```

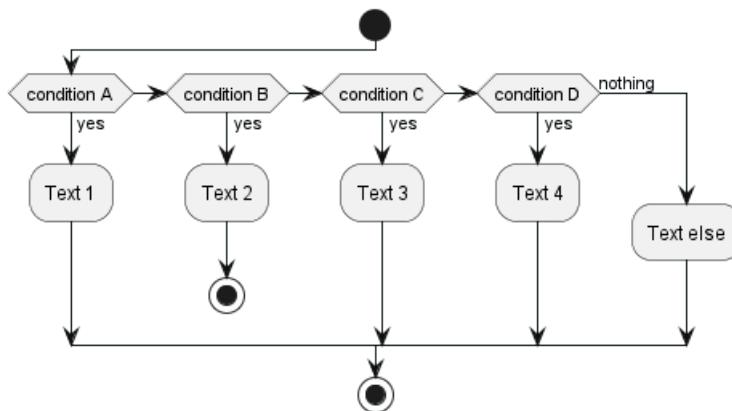




Puedes usar la palabra reservada `elseif` para tener varias condiciones :

```

@startuml
start
if (condition A) then (yes)
  :Text 1;
elseif (condition B) then (yes)
  :Text 2;
  stop
elseif (condition C) then (yes)
  :Text 3;
elseif (condition D) then (yes)
  :Text 4;
else (nothing)
  :Text else;
endif
stop
@enduml
  
```



[Refs. [QA-3931](<https://forum.plantuml.net/3931/please-provide-elseif-structure-vertically-activity-diagrams>), [issue-582](<https://github.com/plantuml/plantuml/issues/582>)]

6.4 Switch and case [switch, case, endswitch]

You can use `switch`, `case` and `endswitch` keywords to put switch in your diagram.

Labels can be provided using parentheses.

```

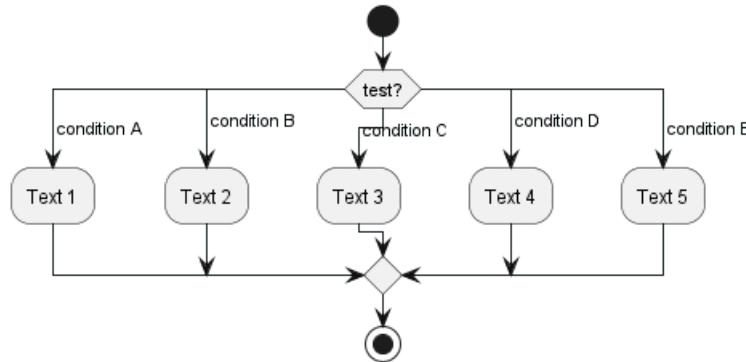
@startuml
start
switch (test?)
case ( condition A )
  :Text 1;
case ( condition B )
  :Text 2;
  
```



```

case ( condition C )
:Text 3;
case ( condition D )
:Text 4;
case ( condition E )
:Text 5;
endswitch
stop
@enduml

```



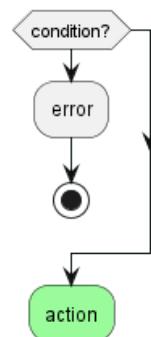
6.5 Conditional with stop on an action [kill, detach]

You can stop action on a if loop.

```

@startuml
if (condition?) then
:error;
stop
endif
#palegreen:action;
@enduml

```



But if you want to stop at the precise action, you can use the `kill` or `detach` keyword:

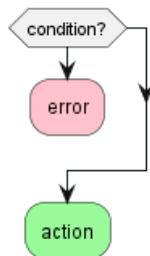
- kill

```

@startuml
if (condition?) then
:#pink:error;
kill
endif
#palegreen:action;
@enduml

```

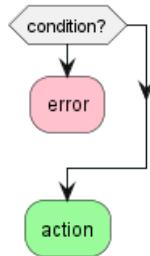




[Ref. QA-265]

- detach

```
@startuml
if (condition?) then
  #pink:error;
  detach
endif
#palegreen:action;
@enduml
```



6.6 El ciclo Repeat

Puedes usar las palabras reservadas `repeat` y `repeatwhile` para colocar bucles.

```
@startuml

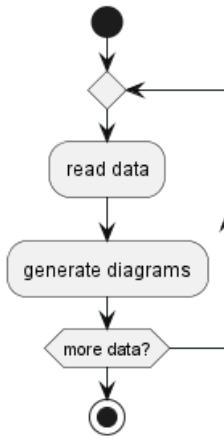
start

repeat
  :read data;
  :generate diagrams;
repeat while (more data?)

stop

@enduml
```





It is also possible to use a full action as `repeat` target and insert an action in the return path using the `backward` keyword.

```

@startuml

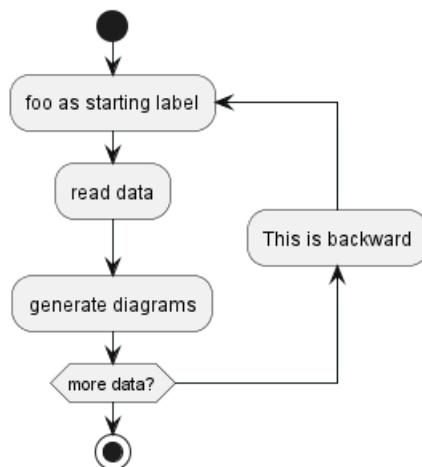
start

repeat :foo as starting label;
    :read data;
    :generate diagrams;
backward:This is backward;
repeat while (more data?)

stop

@enduml

```



*[Ref. [QA-5826](<https://forum.plantuml.net/5826/please-provide-action-repeat-loop-start-instead-condition?show=5831>)]

6.7 Break on a repeat loop [break]

You can use the `break` keyword after an action on a loop.

```

@startuml
start
repeat
    :Test something;
    if (Something went wrong?) then (no)
        #palegreen:OK;

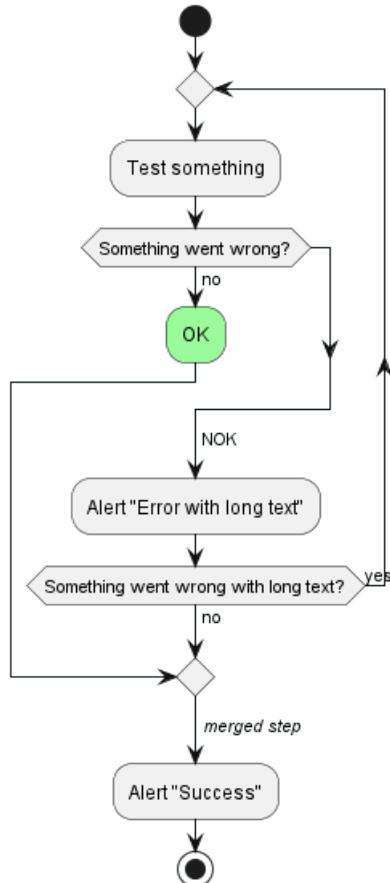
```



```

break
endif
->NOK;
:Alert "Error with long text";
repeat while (Something went wrong with long text?) is (yes) not (no)
->//merged step//;
:Alert "Success";
stop
@enduml

```



[Ref. QA-6105]

6.8 Goto and Label Processing [label, goto]

It is currently only experimental

You can use `label` and `goto` keywords to denote goto processing, with:

- `label <label_name>`
- `goto <label_name>`

```

@startuml
title Point two queries to same activity\nwith `goto`
start
if (Test Question?) then (yes)
'space label only for alignment
label sp_lab0
label sp_lab1
'real label
label lab
:shared;

```

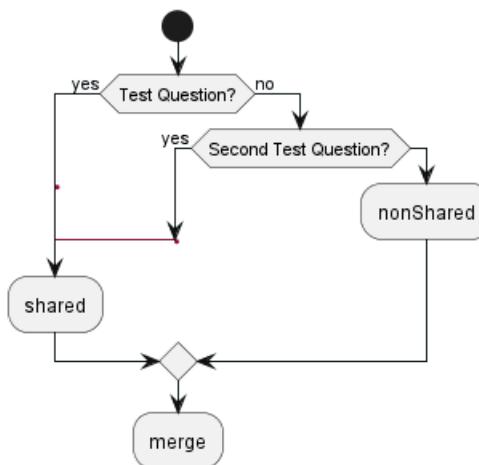


```

else (no)
if (Second Test Question?) then (yes)
label sp_lab2
goto sp_lab1
else
:nonShared;
endif
endif
:merge;
@enduml

```

**Point two queries to same activity
with `goto`**



[Ref. QA-15026, QA-12526 and initially QA-1626]

6.9 El ciclo While

Puedes usar las palabras reservadas `while` y `end while` para un ciclo repetitivo.

`@startuml`

`start`

```

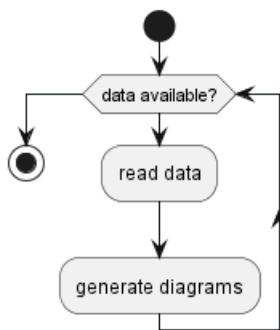
while (data available?)
    :read data;
    :generate diagrams;
endwhile

```

`stop`

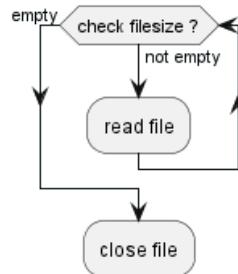
`@enduml`





Es posible proporcionar una etiqueta después de la palabra reservada `endwhile`, o usar la palabra reservada `is`.

```
@startuml
while (check filesize ?) is (not empty)
    :read file;
endwhile (empty)
:close file;
@enduml
```



6.10 Procesamiento paralelo

Puedes usar las palabras reservadas `fork`, `fork again` y `end fork` para denotar procesamientos paralelos.

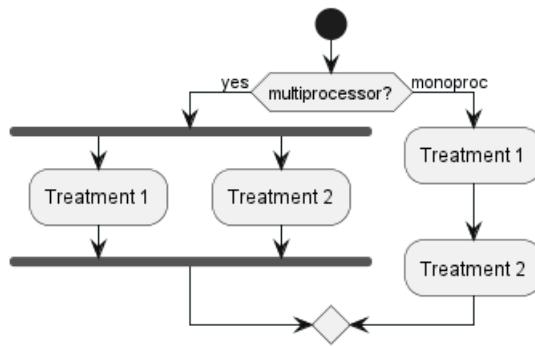
```
@startuml

start

if (multiprocessor?) then (yes)
  fork
    :Treatment 1;
  fork again
    :Treatment 2;
  end fork
else (monoproc)
  :Treatment 1;
  :Treatment 2;
endif

@enduml
```





6.11 Split processing

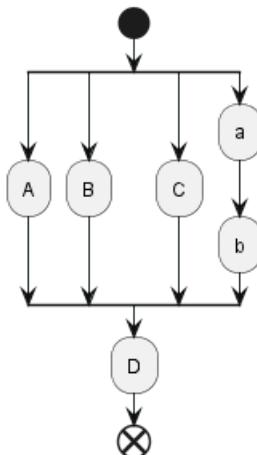
6.11.1 Split

You can use `split`, `split again` and `end split` keywords to denote split processing.

```

@startuml
start
split
  :A;
split again
  :B;
split again
  :C;
split again
  :a;
  :b;
end split
:D;
end
@enduml

```



6.11.2 Input split (multi-start)

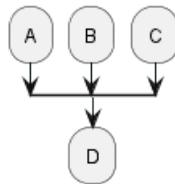
You can use `hidden` arrows to make an input split (multi-start):

```

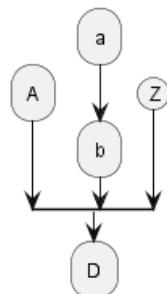
@startuml
split
  -[hidden]->
  :A;
split again
  -[hidden]->
  
```



```
:B;
split again
-[hidden]->
:C;
end split
:D;
@enduml
```



```
@startuml
split
-[hidden]->
:A;
split again
-[hidden]->
:a;
:b;
split again
-[hidden]->
(Z)
end split
:D;
@enduml
```



[Ref. QA-8662]

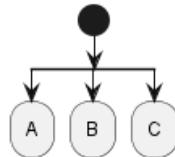
6.11.3 Output split (multi-end)

You can use `kill` or `detach` to make an output split (multi-end):

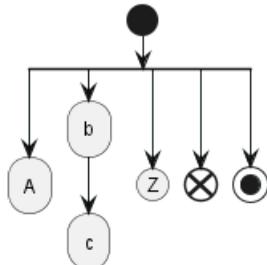
```
@startuml
start
split
:A;
kill
split again
:B;
detach
split again
:C;
kill
```



```
end split
@enduml
```



```
@startuml
start
split
: A;
kill
split again
: b;
: c;
detach
split again
(Z)
detach
split again
end
split again
stop
end split
@enduml
```



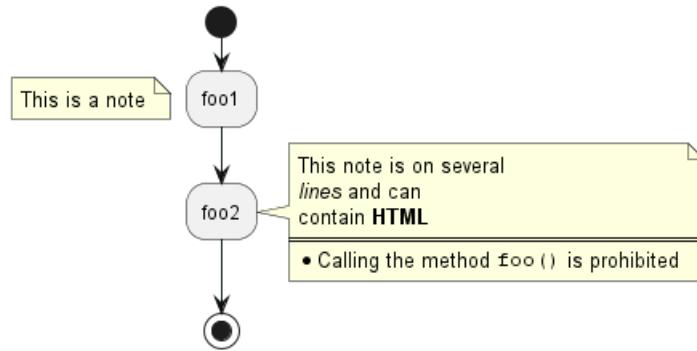
6.12 Notas

Se puede aplicar formato a un texto usando sintaxis de WikiCreole.

Una nota puede ser flotante, usando la palabra clave **floating**.

```
@startuml
start
: foo1;
floating note left: This is a note
: foo2;
note right
This note is on several
//lines// and can
contain <b>HTML</b>
=====
* Calling the method ""foo()"" is prohibited
end note
stop

@enduml
```



[Ref. [QA-2398](<https://forum.plantuml.net/2398/is-it-possible-to-add-a-comment-on-top-of-a-activity-partition?show=2403#a2403>)]

6.13 Colores

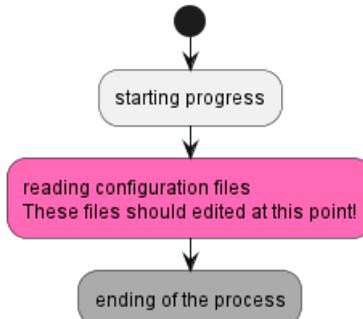
Puedes especificar colores en algunas actividades.

@startuml

```

start
:starting progress;
#HotPink:reading configuration files
These files should edited at this point!;
#AAAAAA:ending of the process;
  
```

@enduml



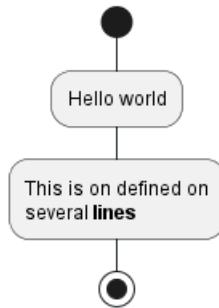
[Ref. [QA-4906](<https://forum.plantuml.net/4906/setting-ad-hoc-gradient-backgrounds-in-activity?show=4917#a4917>)]

6.14 Lines without arrows

You can use `skinparam ArrowHeadColor none` in order to connect activities using lines only, without arrows.

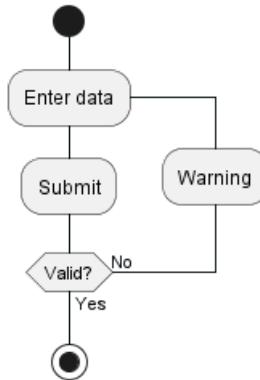
@startuml
`skinparam ArrowHeadColor none`
 start
 :Hello world;
 :This is on defined on
 several **lines**;
 stop
@enduml





```

@startuml
skinparam ArrowHeadColor none
start
repeat :Enter data;
:Submit;
backward :Warning;
repeat while (Valid?) is (No) not (Yes)
stop
@enduml
  
```



6.15 Flechas

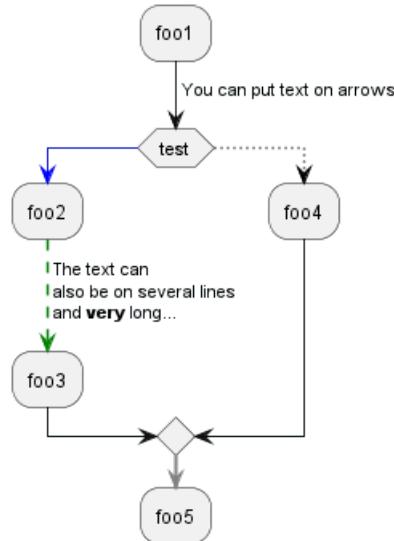
Usando la notación `->`, puedes añadir texto a una flecha y cambiar su color.

También es posible tener flechas punteadas, en linea discontinua, en negrita u ocultas.

```

@startuml
:foo1;
-> You can put text on arrows;
if (test) then
-[#blue]->
:foo2;
-[#green,dashed]-> The text can
also be on several lines
and **very** long...;
:foo3;
else
-[#black,dotted]->
:foo4;
endif
-[#gray,bold]->
:foo5;
@enduml
  
```



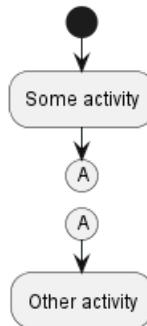


6.16 Connector

You can use parentheses to denote connector.

```

@startuml
start
:Some activity;
(A)
detach
(A)
:Other activity;
@enduml
  
```



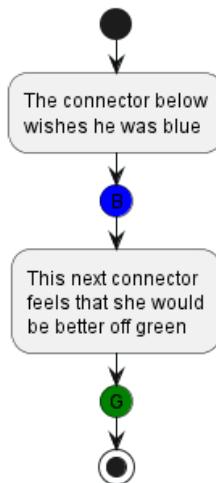
6.17 Color on connector

You can add color on connector.

```

@startuml
start
:The connector below
wishes he was blue;
#blue:(B)
:This next connector
feels that she would
be better off green;
#green:(G)
stop
@enduml
  
```





[Ref. QA-10077]

6.18 Agrupación

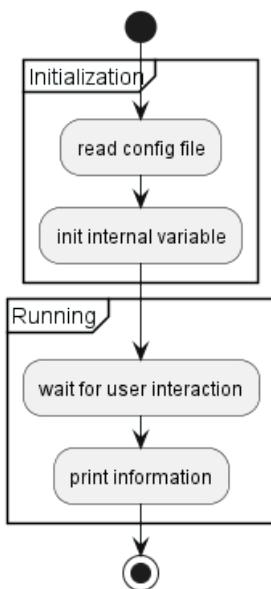
Puedes agrupar actividades definiendo particiones:

```

@startuml
start
partition Initialization {
    :read config file;
    :init internal variable;
}
partition Running {
    :wait for user interaction;
    :print information;
}

stop
@enduml

```



[Ref. [QA-2793](<https://forum.plantuml.net/2793/activity-beta-partition-name-more-than-one-word-does-not-work?show=2798#a2798>)] *[Ref. [QA-542](<https://forum.plantuml.net/542/ability-to-define-hyperlink-on-diagram-elements?show=14003#c14003>)]*

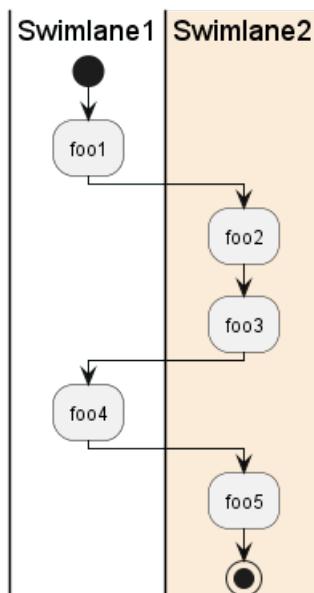


6.19 Carriles

Usando la tecla pipe |, puedes definir carriles.

También es posible cambiar el color de los carriles.

```
@startuml
|Swimlane1|
start
:foo1;
|#AntiqueWhite|Swimlane2|
:foo2;
:foo3;
|Swimlane1|
:foo4;
|Swimlane2|
:foo5;
stop
@enduml
```



[Ref. [QA-2681](<https://forum.plantuml.net/2681/possible-define-alias-swimlane-place-alias-everywhere-else?show=2685#a2685>)]

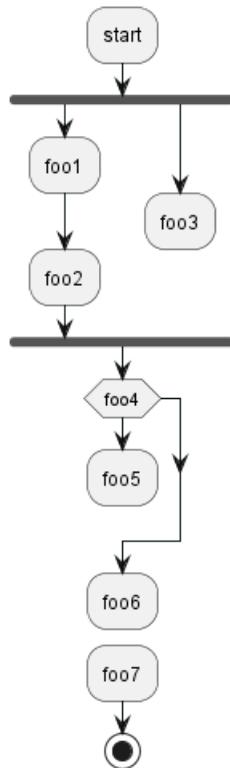
6.20 Desacoplar y remover

Es posible remover una flecha usando la palabra reservada `detach`.

```
@startuml
:start;
fork
:foo1;
:foo2;
fork again
:foo3;
detach
endfork
if (foo4) then
:foo5;
detach
endif
:foo6;
```



```
detach
:foo7;
stop
@enduml
```



6.21 Otras formas de representación de actividades

Cambiando el separador final, ; , puedes configurar diferentes representaciones para una actividad:

- |
- <
- >
- /
-]
- }

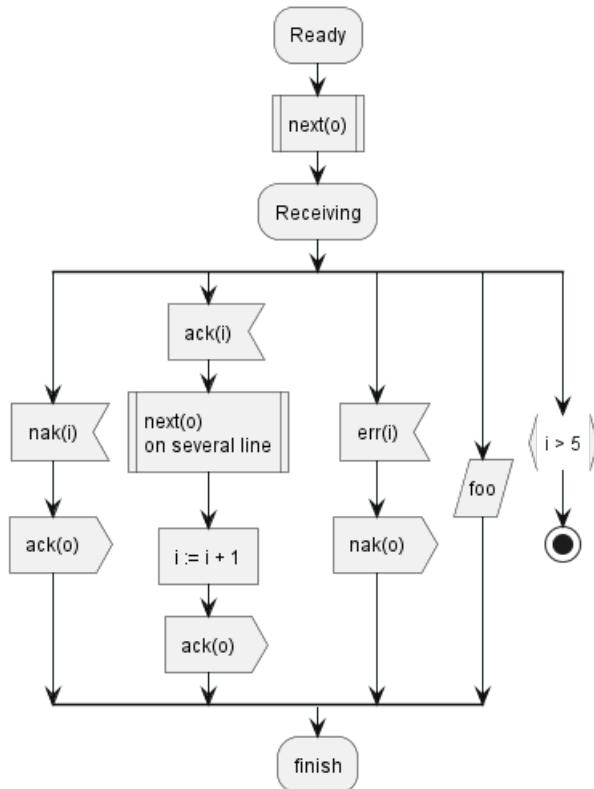
```
@startuml
:Ready;
:next(o)| 
:Receiving;
split
:nak(i)<
:ack(o)>
split again
:ack(i)<
:next(o)
on several line|
:i := i + 1]
:ack(o)>
split again
:err(i)<
```



```

:nak(o)>
split again
:foo/
split again
:i > 5}
stop
end split
:finish;
@enduml

```



6.22 Un ejemplo completo

@startuml

```

start
:ClickServlet.handleRequest();
:new page;
if (Page.onSecurityCheck) then (true)
:Page.onInit();
if (isForward?) then (no)
:Process controls;
if (continue processing?) then (no)
stop
endif

if (isPost?) then (yes)
:Page.onPost();
else (no)
:Page.onGet();
endif
:Page.onRender();
endif

```



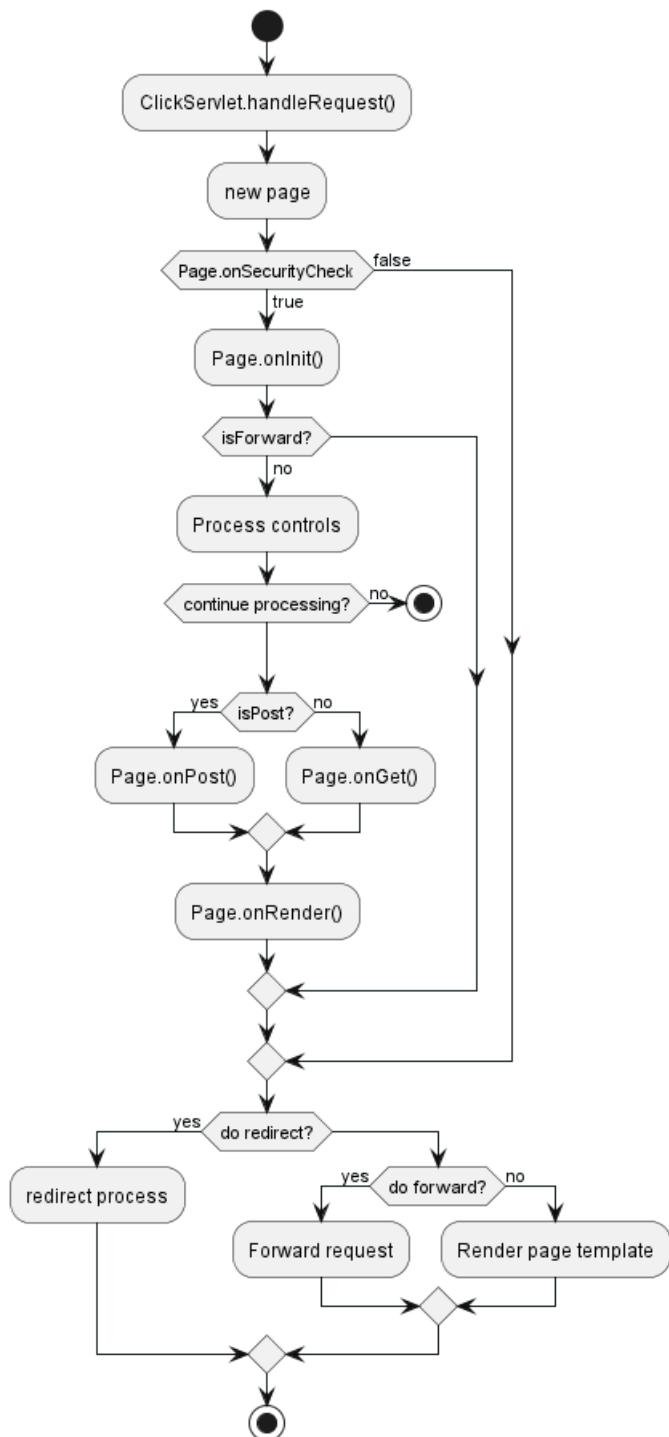
```
else (false)
endif

if (do redirect?) then (yes)
    :redirect process;
else
    if (do forward?) then (yes)
        :Forward request;
    else (no)
        :Render page template;
    endif
endif

stop

@enduml
```





6.23 Condition Style

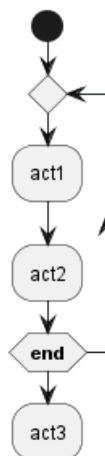
6.23.1 Inside style (by default)

```

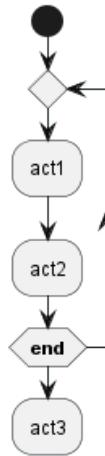
@startuml
skinparam conditionStyle inside
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end</b>)
  :act3;
  
```



```
@enduml
```



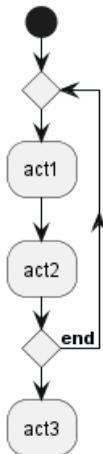
```
@startuml
start
repeat
    :act1;
    :act2;
repeatwhile (<b>end</b>)
    :act3;
@enduml
```



6.23.2 Diamond style

```
@startuml
skinparam conditionStyle diamond
start
repeat
    :act1;
    :act2;
repeatwhile (<b>end</b>)
    :act3;
@enduml
```



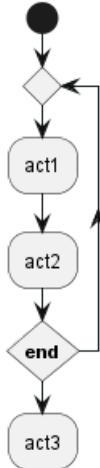


6.23.3 InsideDiamond (or *Foo1*) style

```

@startuml
skinparam conditionStyle InsideDiamond
start
repeat
    :act1;
    :act2;
repeatwhile (<b>end</b>)
    :act3;
@enduml

```

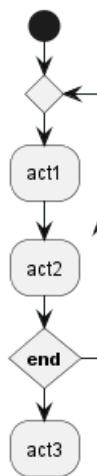


```

@startuml
skinparam conditionStyle foo1
start
repeat
    :act1;
    :act2;
repeatwhile (<b>end</b>)
    :act3;
@enduml

```





[Ref. QA-1290 and #400]

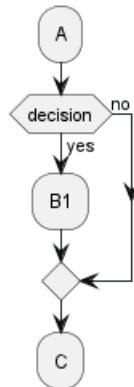
6.24 Condition End Style

6.24.1 Diamond style (by default)

- With one branch

```

@startuml
skinparam ConditionEndStyle diamond
:A;
if (decision) then (yes)
  :B1;
else (no)
endif
:C;
@enduml
  
```

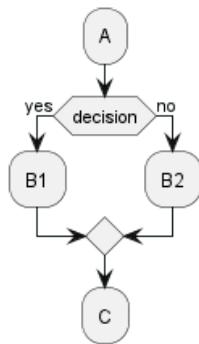


- With two branches (B1, B2)

```

@startuml
skinparam ConditionEndStyle diamond
:A;
if (decision) then (yes)
  :B1;
else (no)
  :B2;
endif
:C;
@enduml
@enduml
  
```



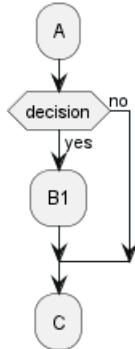


6.24.2 Horizontal line (hline) style

- With one branch

```

@startuml
skinparam ConditionEndStyle hline
:A;
if (decision) then (yes)
  :B1;
else (no)
endif
:C;
@enduml
  
```

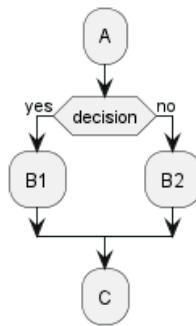


- With two branches (B1, B2)

```

@startuml
skinparam ConditionEndStyle hline
:A;
if (decision) then (yes)
  :B1;
else (no)
  :B2;
endif
:C;
@enduml
@enduml
  
```





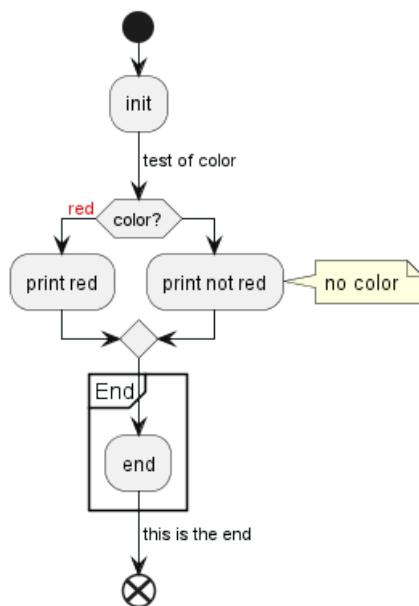
[Ref. QA-4015]

6.25 Using (global) style

6.25.1 Without style (by default)

```

@startuml
start
:init;
-> test of color;
if (color?) is (<color:red>red) then
:print red;
else
:print not red;
note right: no color
endif
partition End {
:end;
}
-> this is the end;
end
@enduml
  
```



6.25.2 With style

You can use style to change rendering of elements.

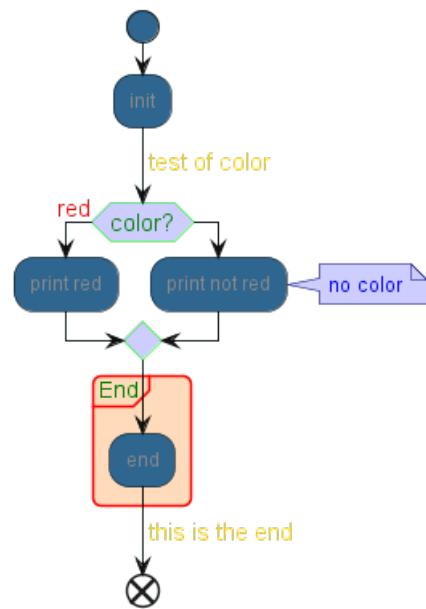
```
@startuml
```



```
<style>
activityDiagram {
    BackgroundColor #33668E
    BorderColor #33668E
    FontColor #888
    FontName arial

    diamond {
        BackgroundColor #ccf
        LineColor #00FF00
        FontColor green
        FontName arial
        FontSize 15
    }
    arrow {
        FontColor gold
        FontName arial
        FontSize 15
    }
    partition {
        LineColor red
        FontColor green
        RoundCorner 10
        BackgroundColor PeachPuff
    }
    note {
        FontColor Blue
        LineColor Navy
        BackgroundColor #ccf
    }
}
document {
    BackgroundColor transparent
}
</style>
start
: init;
-> test of color;
if (color?) is (<color:red>red) then
: print red;
else
: print not red;
note right: no color
endif
partition End {
: end;
}
-> this is the end;
end
@enduml
```





7 Diagrama de Componentes

Veamos algunos ejemplos : Let's have few examples.

Advantages of PlantUML:

- **Simplicity:** With PlantUML, you can create component diagrams using simple and intuitive text-based descriptions, eliminating the need for complex drawing tools.
- **Integration:** PlantUML seamlessly integrates with various tools and platforms, making it a versatile choice for developers and architects.
- **Collaboration:** The [PlantUML forum](<https://forum.plantuml.net/>) offers a platform for users to discuss, share, and seek assistance on their diagrams, fostering a collaborative community.

7.1 Componentes

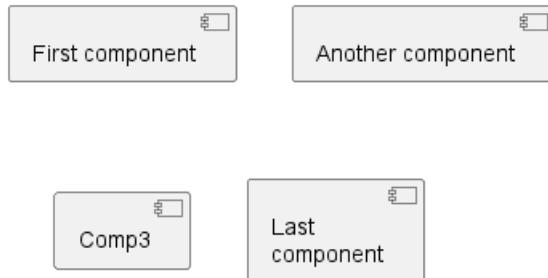
Los componentes deberían ser encerrados entre corchetes [] .

También puedes usar la palabra reservada **component** para definir un componente. Y puedes definir un alias, usando la palabra reservada **as**. Este alias será usado más adelante, cuando definamos relaciones.

@startuml

```
[First component]
[Another component] as Comp2
component Comp3
component [Last\ncomponent] as Comp4
```

@enduml



7.2 Interfaces

Puedes definir una interfaz usando el símbolo () (porque esto luce como un circulo).

Puedes usar también la palabra reservada **interface** para definir una interfaz. Y puedes definir un alias, usando la palabra reservada **as**. Este alias sera usado luego, definiendo las relaciones.

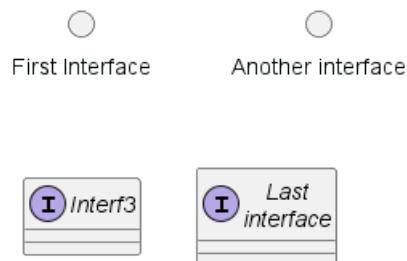
Nosotros veremos mas adelante que la definición de interfaz es opcional.

@startuml

```
() "First Interface"
() "Another interface" as Interf2
interface Interf3
interface "Last\ninterface" as Interf4
```

@enduml





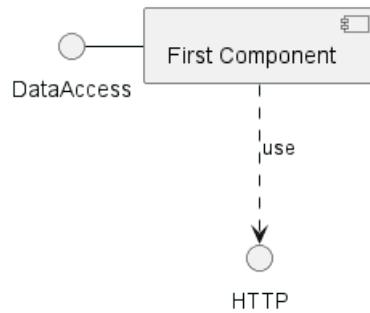
7.3 Ejemplos basicos

Los links entre los elementos se hacen usando la combinación de símbolos de linea de puntos (..), linea recta (--), y flechas (-->)

@startuml

```
DataAccess - [First Component]
[First Component] ..> HTTP : use
```

@enduml



7.4 Usando notas

Puedes usar el `note left of`, `note right of`, `note top of`, `note bottom of` Las palabras reservadas para definir notas relacionadas a un objeto simple.

Una nota puede ser definida sola usando la palabra reservada `note`, luego linkea a otro objeto usando el símbolo ...

@startuml

```
interface "Data Access" as DA

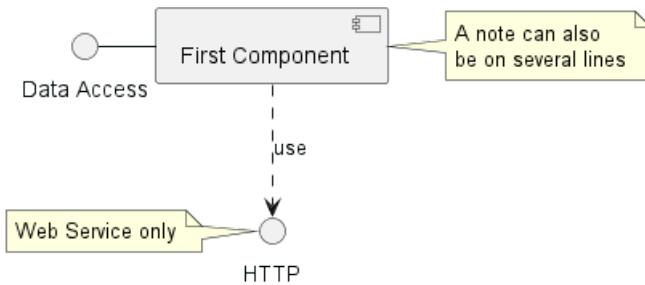
DA - [First Component]
[First Component] ..> HTTP : use

note left of HTTP : Web Service only

note right of [First Component]
  A note can also
  be on several lines
end note

@enduml
```





7.5 Agrupando componentes

Puedes usar varias palabras reservadas para agrupar componentes e interfaces juntos:

- package
- node
- folder
- frame
- cloud
- database

@startuml

```

package "Some Group" {
    HTTP - [First Component]
    [Another Component]
}

node "Other Groups" {
    FTP - [Second Component]
    [First Component] --> FTP
}

cloud {
    [Example 1]
}

database "MySql" {
    folder "This is my folder" {
        [Folder 3]
    }
    frame "Foo" {
        [Frame 4]
    }
}

```

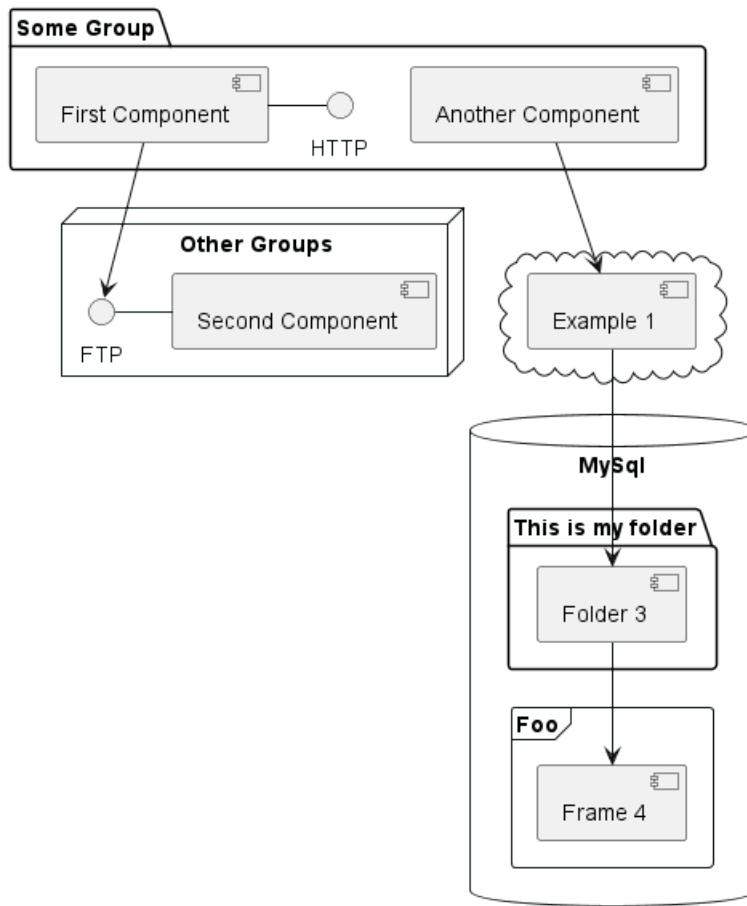
```

[Another Component] --> [Example 1]
[Example 1] --> [Folder 3]
[Folder 3] --> [Frame 4]

```

@enduml



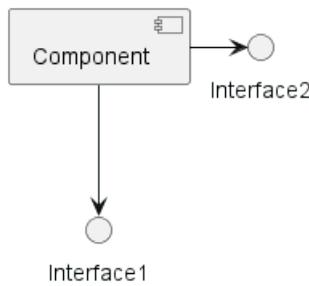


7.6 Cambiando la dirección de las flechas

Por defecto los links entre clases tienen dos guiones --y son orientados verticalmente. Puedes usar la orientacion horizontal para un link poniendo un guion (o punto) como en el siguiente ejemplo:

```

@startuml
[Component] --> Interface1
[Component] -> Interface2
@enduml
  
```

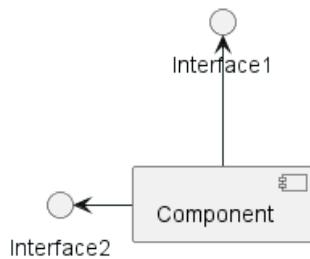


Puedes también cambiar direcciones invirtiendo el link:

```

@startuml
Interface1 <-- [Component]
Interface2 <- [Component]
@enduml
  
```

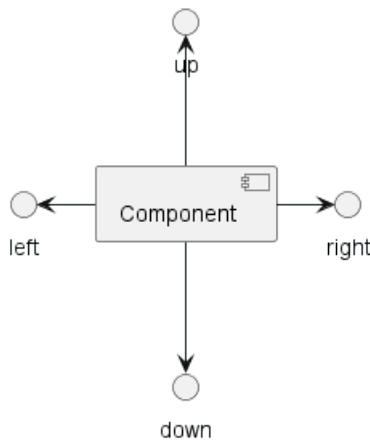




También es posible cambiar la dirección de las flechas agregando la palabra reservada `left`, `right`, `up` o `down` dentro de la flecha:

```

@startuml
[Component] -left-> left
[Component] -right-> right
[Component] -up-> up
[Component] -down-> down
@enduml
  
```



Puedes acortar la flecha usando el primer carácter (por ejemplo, `-d-` en lugar de `-down-`) o los dos primeros caracteres (`-do-`).

Por favor nota que no puedes abusar de esta funcionalidad *Graphviz* que usualmente otorga buenos resultados sin ajustes.

7.7 Use UML2 notation

By default (*from v1.2020.13-14*), UML2 notation is used.

```
@startuml
```

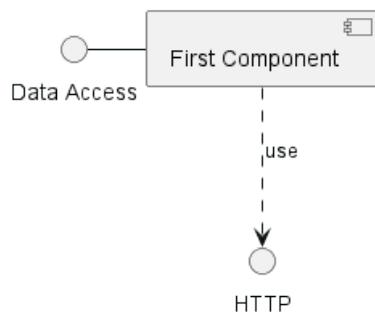
```

interface "Data Access" as DA

DA - [First Component]
[First Component] ..> HTTP : use

@enduml
  
```





7.8 Utiliza la notación UML1

El comando `skinparam componentStyle uml1` es usado para cambiar hacia la notación UML1.

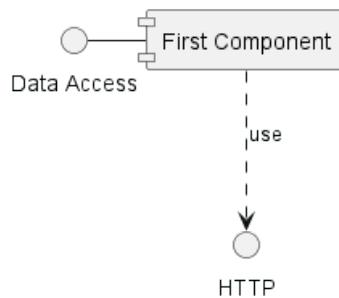
```

@startuml
skinparam componentStyle uml1

interface "Data Access" as DA

DA - [First Component]
[First Component] ..> HTTP : use

@enduml
  
```



7.9 Use rectangle notation (remove UML notation)

The `skinparam componentStyle rectangle` command is used to switch to rectangle notation (*without any UML notation*).

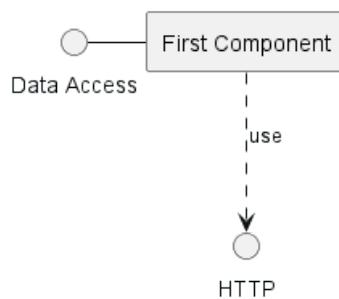
```

@startuml
skinparam componentStyle rectangle

interface "Data Access" as DA

DA - [First Component]
[First Component] ..> HTTP : use

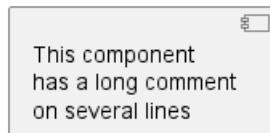
@enduml
  
```



7.10 Long description

It is possible to put description on several lines using square brackets.

```
@startuml
component comp1 [
This component
has a long comment
on several lines
]
@enduml
```



7.11 Colores individuales

Puedes especificar un color despues de la definición del componente.

```
@startuml
component [Web Server] #Yellow
@enduml
```

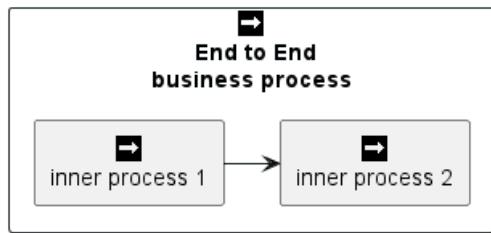


7.12 Using Sprite in Stereotype

You can use sprites within stereotype components.

```
@startuml
sprite $businessProcess [16x16/16] {
FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFOFFFFF
FFFFFFFFFFFFFOOOFFF
FF00000000000000FF
FF00000000000000FF
FF00000000000000FF
FFFFFFFFFFFFFOFFFFF
FFFFFFFFFFFFFOFFFFF
FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF
}
rectangle " End to End\nbusiness process" <<$businessProcess>> {
rectangle "inner process 1" <<$businessProcess>> as src
rectangle "inner process 2" <<$businessProcess>> as tgt
src -> tgt
}
@enduml
```





7.13 Personalización (Skinparam)

Puedes usar el comando skinparam para cambiar los colores y las fuentes de los dibujos

Puedes usar este comando:

- En la definición del diagrama, como cualquier otro comando,
- En un archivo incluido,
- En un archivo de configuración, proporcionado en la consola de comandos o en el ANT task.

Puedes definir colores y fuentes específicas para interfaces y componentes estereotipados.

@startuml

```

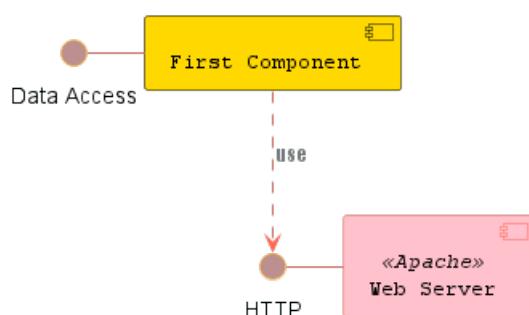
skinparam interface {
    backgroundColor RosyBrown
    borderColor orange
}

skinparam component {
    FontSize 13
    BackgroundColor<<Apache>> Pink
    BorderColor<<Apache>> #FF6655
    FontName Courier
    BorderColor black
    BackgroundColor gold
    ArrowFontName Impact
    ArrowColor #FF6655
    ArrowFontColor #777777
}

() "Data Access" as DA
Component "Web Server" as WS << Apache >>

DA - [First Component]
[First Component] ..> () HTTP : use
HTTP - WS
  
```

@enduml



```

@startuml

skinparam component {
    backgroundColor<<static_lib>> DarkKhaki
    backgroundColor<<shared_lib>> Green
}

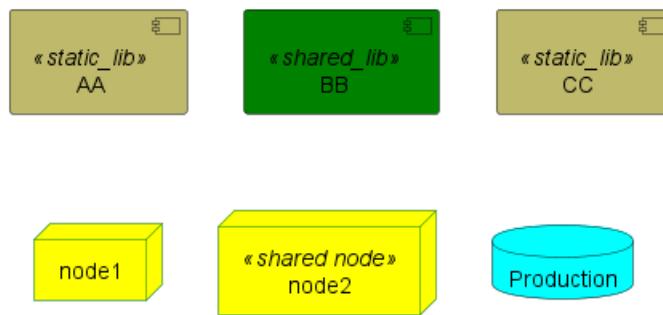
skinparam node {
borderColor Green
backgroundColor Yellow
backgroundColor<<shared_node>> Magenta
}
skinparam databaseBackgroundColor Aqua

[AA] <<static_lib>>
[BB] <<shared_lib>>
[CC] <<static_lib>>

node node1
node node2 <<shared node>>
database Production

@enduml

```



7.14 Specific SkinParameter

7.14.1 componentStyle

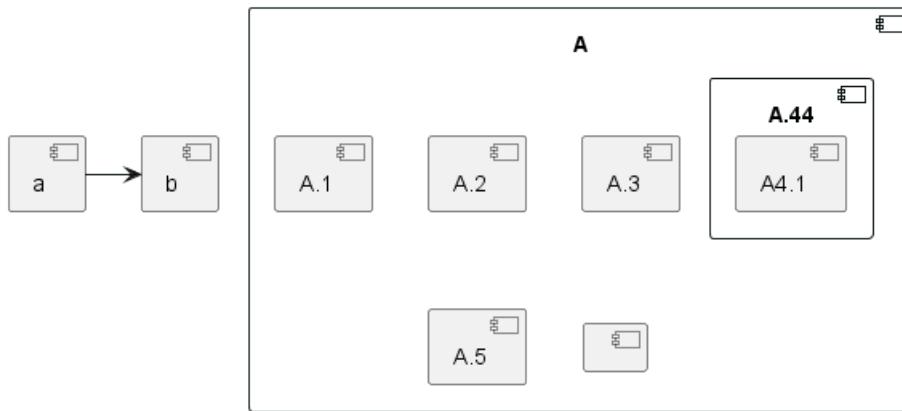
- By default (or with `skinparam componentStyle uml2`), you have an icon for component

```

@startuml
skinparam BackgroundColor transparent
skinparam componentStyle uml2
component A {
    component "A.1" {
}
    component A.44 {
        [A4.1]
}
    component "A.2"
    [A.3]
    component A.5 [
A.5]
    component A.6 [
]
}
[a]->[b]
@enduml

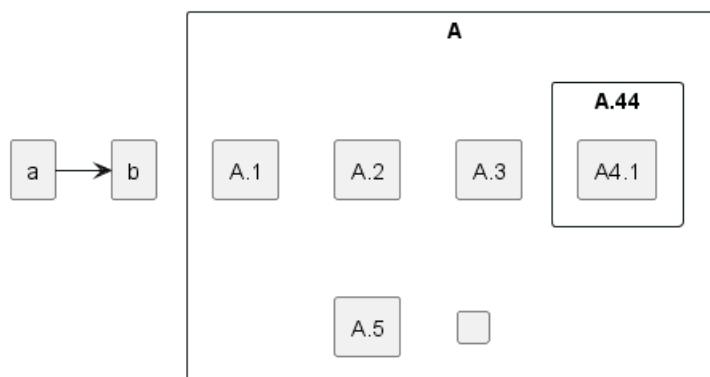
```





- If you want to suppress it, and to have only the rectangle, you can use `skinparam componentStyle rectangle`

```
@startuml
skinparam BackgroundColor transparent
skinparam componentStyle rectangle
component A {
    component "A.1" {
    }
    component A.44 {
        [A4.1]
    }
    component "A.2"
    [A.3]
    component A.5 [
    A.5]
    component A.6 [
    ]
}
[a]-->[b]
@enduml
```



[Ref. 10798]

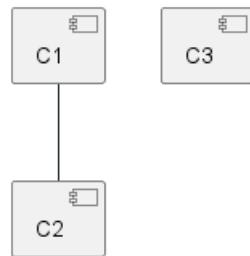
7.15 Hide or Remove unlinked component

By default, all components are displayed:

```
@startuml
component C1
component C2
component C3
C1 -- C2
```



```
@enduml
```

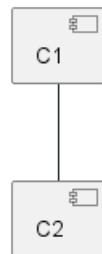


But you can:

- hide @unlinked components:

```
@startuml
component C1
component C2
component C3
C1 --> C2
```

```
hide @unlinked
@enduml
```



- or remove @unlinked components:

```
@startuml
component C1
component C2
component C3
C1 --> C2
```

```
remove @unlinked
@enduml
```



[Ref. QA-11052]

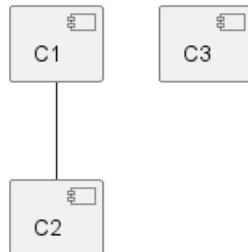
7.16 Hide, Remove or Restore tagged component or wildcard

You can put \$tags (using \$) on components, then remove, hide or restore components either individually or by tags.

By default, all components are displayed:



```
@startuml
component C1 $tag13
component C2
component C3 $tag13
C1 -- C2
@enduml
```



But you can:

- hide \$tag13 components:

```
@startuml
component C1 $tag13
component C2
component C3 $tag13
C1 -- C2

hide $tag13
@enduml
```



- or remove \$tag13 components:

```
@startuml
component C1 $tag13
component C2
component C3 $tag13
C1 -- C2

remove $tag13
@enduml
```



- or remove \$tag13 and restore \$tag1 components:

```
@startuml
component C1 $tag13 $tag1
component C2
component C3 $tag13
C1 -- C2

remove $tag13
```



```
restore $tag1
@enduml
```



- or remove * and restore \$tag1 components:

```
@startuml
component C1 $tag13 $tag1
component C2
component C3 $tag13
C1 -- C2

remove *
restore $tag1
@enduml
```

[Ref. QA-7337 and QA-11052]

7.17 Display JSON Data on Component diagram

7.17.1 Simple example

```
@startuml
allowmixing

component Component
()           Interface

json JSON {
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}
@enduml
```



JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

For another example, see on JSON page.



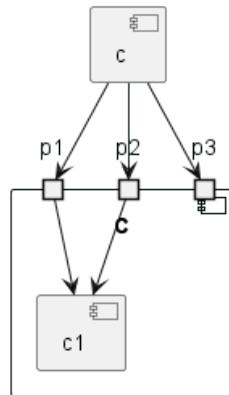
7.18 Port [port, portIn, portOut]

You can add **port** with **port**, **portin** and **portout** keywords.

7.18.1 Port

```
@startuml
[c]
component C {
    port p1
    port p2
    port p3
    component c1
}

c --> p1
c --> p2
c --> p3
p1 --> c1
p2 --> c1
@enduml
```

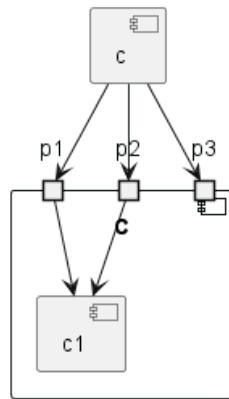


7.18.2 PortIn

```
@startuml
[c]
component C {
    portin p1
    portin p2
    portin p3
    component c1
}

c --> p1
c --> p2
c --> p3
p1 --> c1
p2 --> c1
@enduml
```

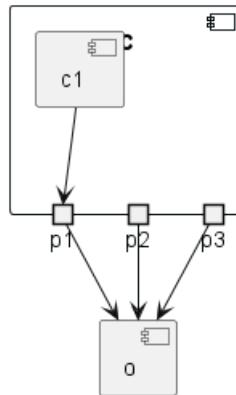




7.18.3 PortOut

```

@startuml
component C {
    portout p1
    portout p2
    portout p3
    component c1
}
[o]
p1 --> o
p2 --> o
p3 --> o
c1 --> p1
@enduml
  
```



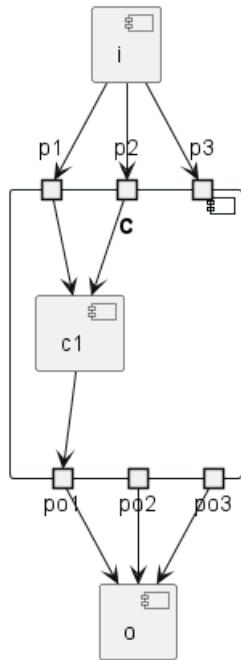
7.18.4 Mixing PortIn & PortOut

```

@startuml
[i]
component C {
    portin p1
    portin p2
    portin p3
    portout po1
    portout po2
    portout po3
    component c1
}
[o]
i --> p1
  
```



```
i --> p2  
i --> p3  
p1 --> c1  
p2 --> c1  
p01 --> o  
p02 --> o  
p03 --> o  
c1 --> p01  
@enduml
```



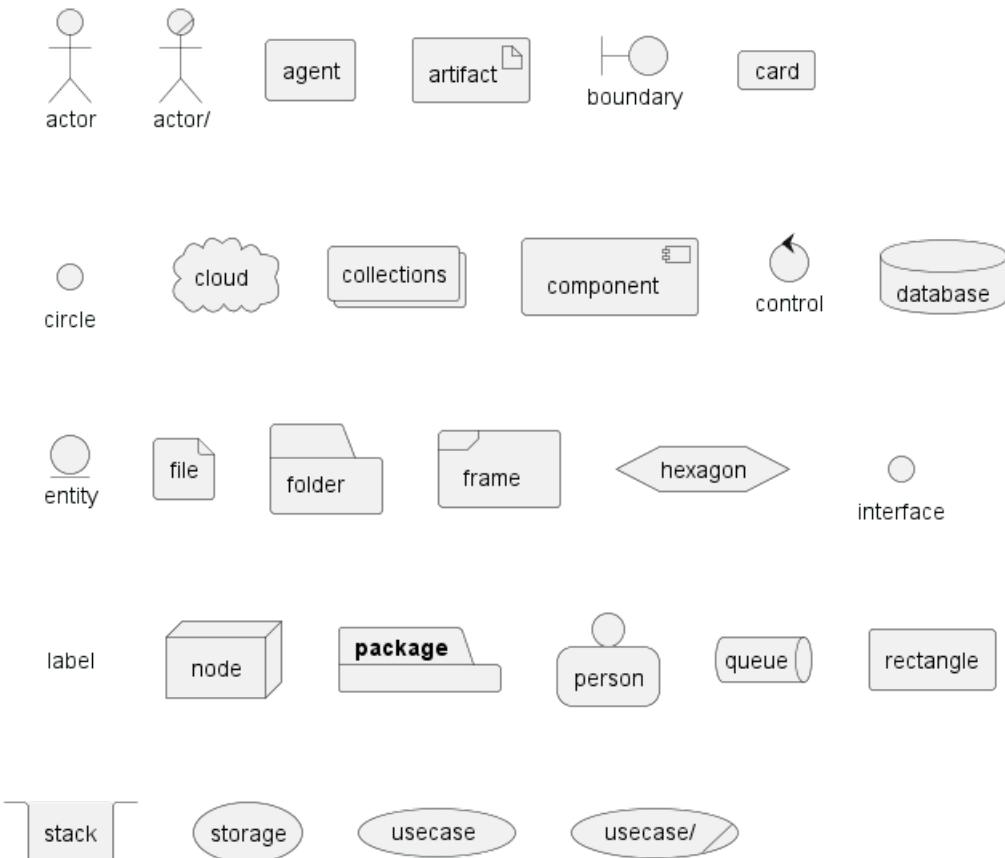
8 Diagrama de despliegue

With [PlantUML](<https://plantuml.com/en/deployment-diagram>), creating deployment diagrams becomes a breeze. The platform offers a simple and intuitive way to design these diagrams using plain text, ensuring rapid iterations and easy version control. Moreover, the [PlantUML forum](<https://forum.plantuml.net/>) provides a vibrant community where users can seek help, share ideas, and collaborate on diagramming challenges. One of the key advantages of PlantUML is its ability to integrate seamlessly with various tools and platforms, making it a preferred choice for professionals and enthusiasts alike.

8.1 Declaring element

```
@startuml  
actor actor  
actor/ "actor/"  
agent agent  
artifact artifact  
boundary boundary  
card card  
circle circle  
cloud cloud  
collections collections  
component component  
control control  
database database  
entity entity  
file file  
folder folder  
frame frame  
hexagon hexagon  
interface interface  
label label  
node node  
package package  
person person  
queue queue  
rectangle rectangle  
stack stack  
storage storage  
usecase usecase  
usecase/ "usecase/"  
@enduml
```





You can optionally put text using bracket [] for a long description.

```
@startuml
folder folder [
This is a <b>folder
-----
You can use separator
=====
of different kind
....
and style
]
```

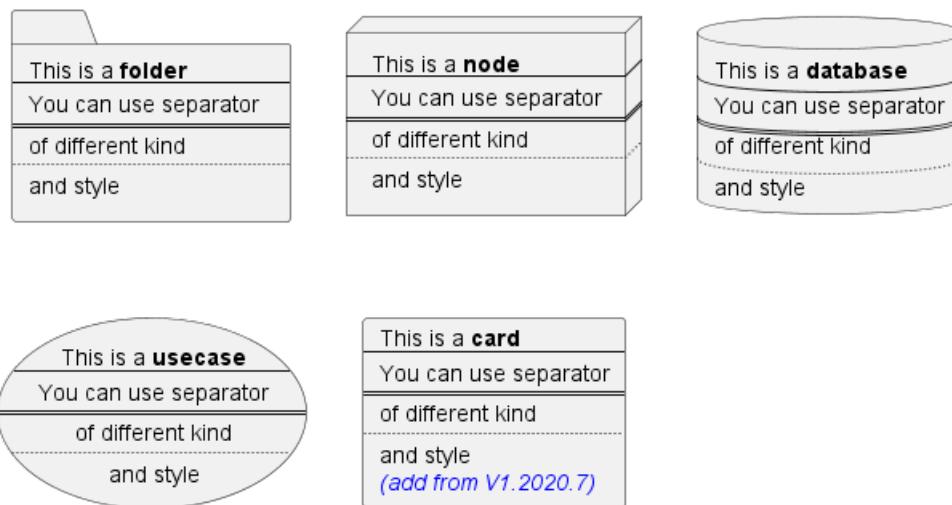
```
node node [
This is a <b>node
-----
You can use separator
=====
of different kind
....
and style
]
```

```
database database [
This is a <b>database
-----
You can use separator
=====
of different kind
....
and style
```



```
]
usecase usecase [
This is a <b>usecase
-----
You can use separator
=====
of different kind
....
and style
]
```

```
card card [
This is a <b>card
-----
You can use separator
=====
of different kind
....
and style
<i><color:blue>(add from V1.2020.7)</color></i>
]
@enduml
```



8.2 Declaring element (using short form)

We can declare element using some short forms.

Long form Keyword	Short form Keyword	Long form example	Short form example	Ref.
actor	: a :	actor actor1	:actor2:	Actors
component	[c]	component component1	[component2]	Components
interface	() i	interface interface1	() "interface2"	Interfaces
usecase	(u)	usecase usecase1	(usecase2)	Usecases

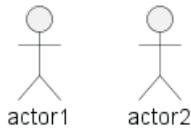
8.2.1 Actor

```
@startuml
```

```
actor actor1
:actor2:
```

```
@enduml
```





NB: There is an old syntax for actor with guillemet which is now deprecated and will be removed some days. Please do not use in your diagram.

8.2.2 Component

```
@startuml
```

```
component component1
[component2]
```

```
@enduml
```



8.2.3 Interface

```
@startuml
```

```
interface interface1
() "interface2"

label "//interface example//"
@enduml
```



interface example

8.2.4 Usecase

```
@startuml
```

```
usecase usecase1
(usecase2)
```

```
@enduml
```



8.3 Linking or arrow

You can create simple links between elements with or without labels:

```
@startuml
```

```
node node1
node node2
node node3
node node4
```

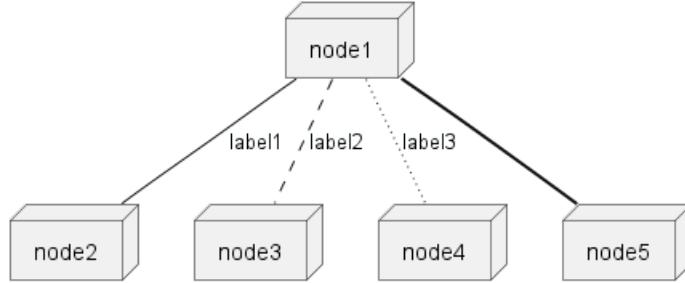


```

node node5
node1 -- node2 : label1
node1 .. node3 : label2
node1 ~~ node4 : label3
node1 == node5

@enduml

```



It is possible to use several types of links:

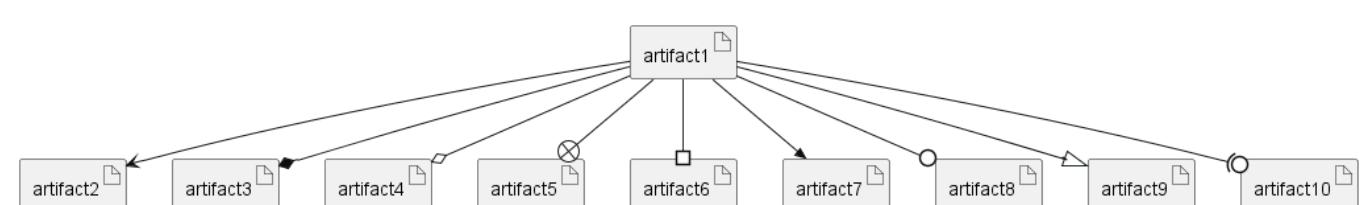
```

@startuml

artifact artifact1
artifact artifact2
artifact artifact3
artifact artifact4
artifact artifact5
artifact artifact6
artifact artifact7
artifact artifact8
artifact artifact9
artifact artifact10
artifact1 --> artifact2
artifact1 --* artifact3
artifact1 --o artifact4
artifact1 --+ artifact5
artifact1 --# artifact6
artifact1 -->> artifact7
artifact1 --o artifact8
artifact1 --^ artifact9
artifact1 --(0 artifact10

@enduml

```



You can also have the following types:

```
@startuml
```

```

cloud cloud1
cloud cloud2
cloud cloud3
cloud cloud4

```

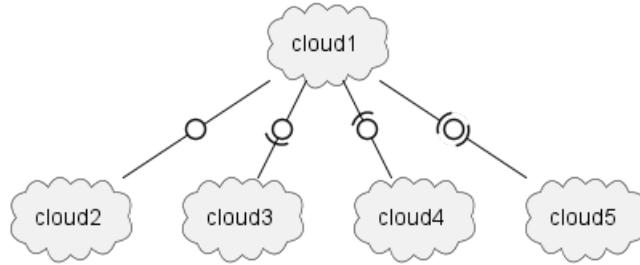


```

cloud cloud5
cloud1 -0- cloud2
cloud1 -0)- cloud3
cloud1 -(0- cloud4
cloud1 -(0)- cloud5

```

@enduml



or another example:

```

@startuml
actor foo1
actor foo2
foo1 <-0-> foo2
foo1 <-(0)-> foo2

```

```

(ac1) -le(0)-> left1
ac1 -ri(0)-> right1
ac1 .up(0).> up1
ac1 ~up(0)~> up2
ac1 -do(0)-> down1
ac1 -do(0)-> down2

```

actor1 -0)- actor2

```

component comp1
component comp2
comp1 *-0)--+ comp2
[comp3] <-->> [comp4]

```

```

boundary b1
control c1
b1 -(0)- c1

```

```

component comp1
interface interf1
comp1 #~~( interf1

```

```

:mode1actor: -0)- fooa1
:mode1actorl: -ri0)- foo1

```

```

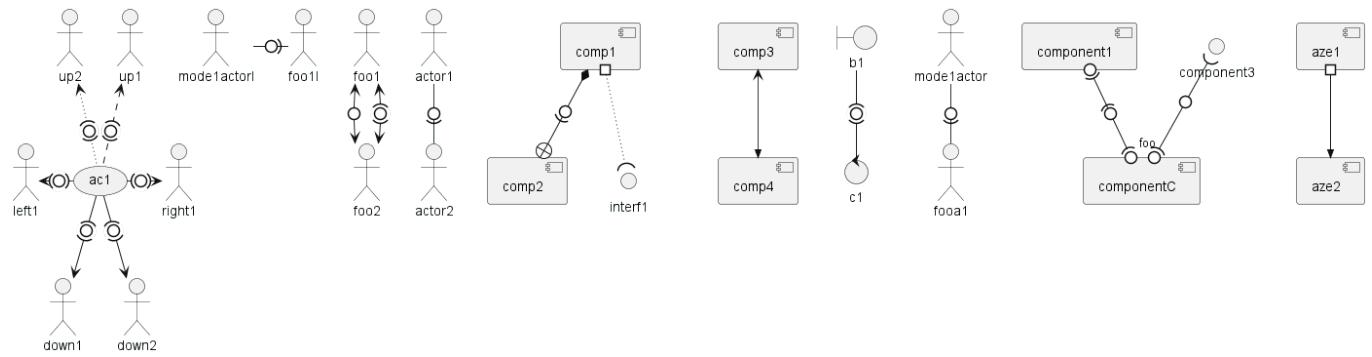
[component1] 0)-(0-(0 [componentC]
() component3 )-0-(0 "foo" [componentC]

```

[aze1] #-->> [aze2]

@enduml





[Ref. QA-547 and QA-1736]

See all type on [Appendix](#).

8.4 Bracketed arrow style

Similar as *Bracketed class relations (linking or arrow) style*

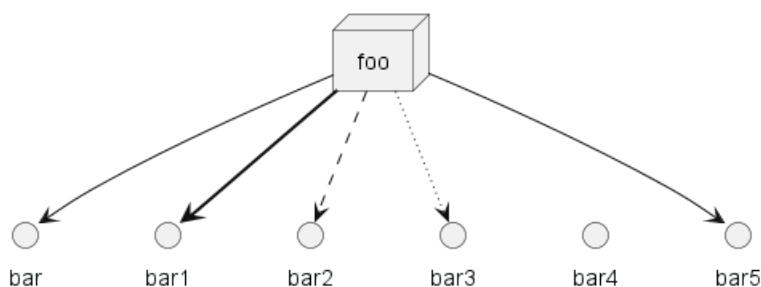
8.4.1 Line style

It's also possible to have explicitly **bold**, **dashed**, **dotted**, **hidden** or **plain** arrows:

- without label

```
@startuml
node foo
title Bracketed line style without label
foo --> bar
foo -[bold]-> bar1
foo -[dashed]-> bar2
foo -[dotted]-> bar3
foo -[hidden]-> bar4
foo -[plain]-> bar5
@enduml
```

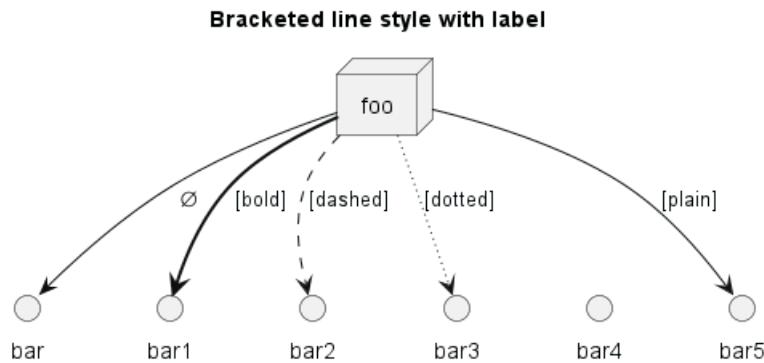
Bracketed line style without label



- with label

```
@startuml
title Bracketed line style with label
node foo
foo --> bar      :
foo -[bold]-> bar1   : [bold]
foo -[dashed]-> bar2  : [dashed]
foo -[dotted]-> bar3   : [dotted]
foo -[hidden]-> bar4   : [hidden]
foo -[plain]-> bar5   : [plain]
@enduml
```

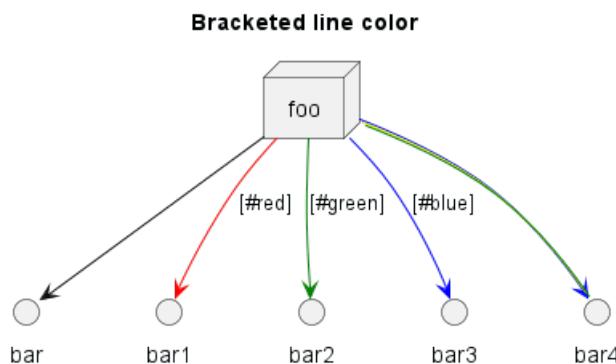




[Adapted from QA-4181]

8.4.2 Line color

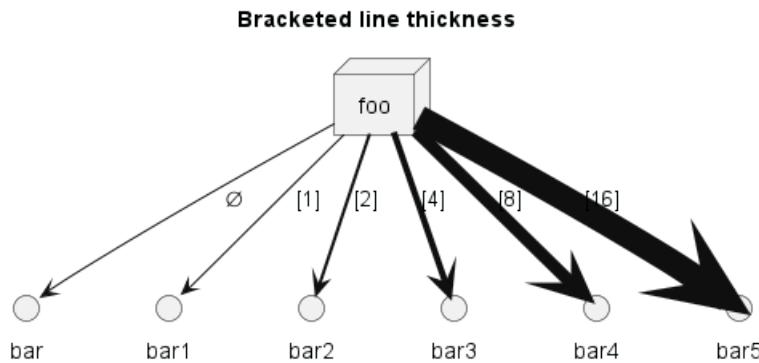
```
@startuml
title Bracketed line color
node foo
foo --> bar
foo -[#red]-> bar1      : [#red]
foo -[#green]-> bar2     : [#green]
foo -[#blue]-> bar3      : [#blue]
foo -[#blue;#yellow;#green]-> bar4
@enduml
```



8.4.3 Line thickness

```
@startuml
title Bracketed line thickness
node foo
foo --> bar      :
foo -[thickness=1]-> bar1   : [1]
foo -[thickness=2]-> bar2   : [2]
foo -[thickness=4]-> bar3   : [4]
foo -[thickness=8]-> bar4   : [8]
foo -[thickness=16]-> bar5  : [16]
@enduml
```



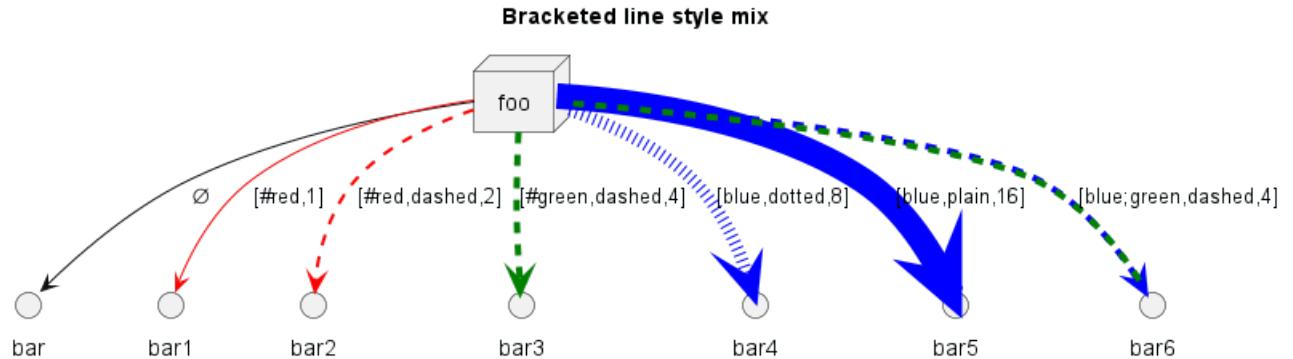


[Adapted from QA-4949]

8.4.4 Mix

```

@startuml
title Bracketed line style mix
node foo
foo --> bar
foo -[#red,thickness=1]-> bar1 : [#red,1]
foo -[#red,dashed,thickness=2]-> bar2 : [#red,dashed,2]
foo -[#green,dashed,thickness=4]-> bar3 : [#green,dashed,4]
foo -[#blue,dotted,thickness=8]-> bar4 : [blue,dotted,8]
foo -[#blue,plain,thickness=16]-> bar5 : [blue,plain,16]
foo -[#blue;#green,dashed,thickness=4]-> bar6 : [blue;green,dashed,4]
@enduml
  
```



8.5 Change arrow color and style (inline style)

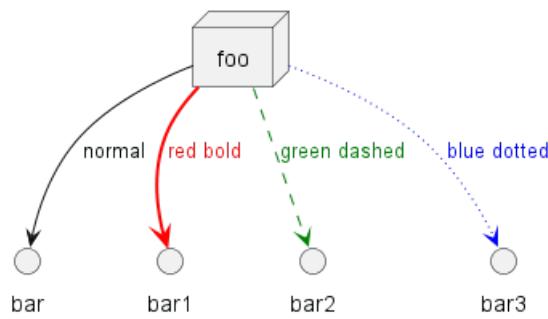
You can change the color or style of individual arrows using the inline following notation:

- #color;line.[bold|dashed|dotted];text:color

```

@startuml
node foo
foo --> bar : normal
foo --> bar1 #line:red;line.bold;text:red : red bold
foo --> bar2 #green;line.dashed;text:green : green dashed
foo --> bar3 #blue;line.dotted;text:blue : blue dotted
@enduml
  
```





[Ref. QA-3770 and QA-3816] [See similar feature on class diagram]

8.6 Change element color and style (inline style)

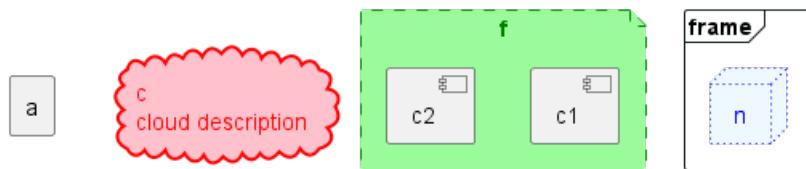
You can change the color or style of individual element using the following notation:

- # [color|back:color];line:color;line.[bold|dashed|dotted];text:color

```
@startuml
agent a
cloud c #pink;line:red;line.bold;text:red
file f #palegreen;line:green;line.dashed;text:green
node n #aliceblue;line:blue;line.dotted;text:blue
@enduml
```



```
@startuml
agent a
cloud c #pink;line:red;line.bold;text:red [
  c
  cloud description
]
file f #palegreen;line:green;line.dashed;text:green [
  [c1]
  [c2]
]
frame frame {
  node n #aliceblue;line:blue;line.dotted;text:blue
}
@enduml
```



[Ref. QA-6852]



8.7 Nestable elements

Here are the nestable elements:

```
@startuml
artifact artifact {
}
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
}
node node {
}
package package {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml
```



8.8 Packages and nested elements

8.8.1 Example with one level

```
@startuml
artifact      artifactVeryL00000000000000000000000g      as "artifact" {
file f1
}
card         cardVeryL00000000000000000000000g      as "card" {
file f2
}
cloud        cloudVeryL00000000000000000000000g      as "cloud" {
file f3
}
component    componentVeryL00000000000000000000000g   as "component" {
file f4
}
database     databaseVeryL00000000000000000000000g    as "database" {
file f5
}
```



```

}

file      fileVeryL0000000000000000000g      as "file" {
file f6
}

folder    folderVeryL0000000000000000000g     as "folder" {
file f7
}

frame     frameVeryL0000000000000000000g      as "frame" {
file f8
}

hexagon   hexagonVeryL0000000000000000000g     as "hexagon" {
file f9
}

node      nodeVeryL0000000000000000000g      as "node" {
file f10
}

package   packageVeryL0000000000000000000g     as "package" {
file f11
}

queue     queueVeryL0000000000000000000g      as "queue" {
file f12
}

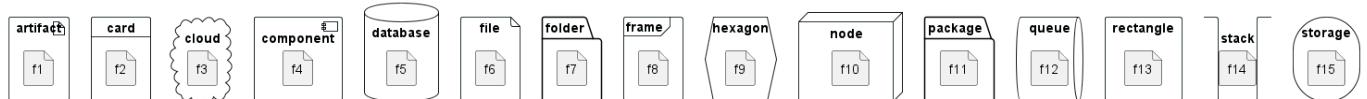
rectangle rectangleVeryL0000000000000000000g   as "rectangle" {
file f13
}

stack     stackVeryL0000000000000000000g      as "stack" {
file f14
}

storage   storageVeryL0000000000000000000g     as "storage" {
file f15
}

@enduml

```



8.8.2 Other example

```

@startuml
artifact Foo1 {
    folder Foo2
}

folder Foo3 {
    artifact Foo4
}

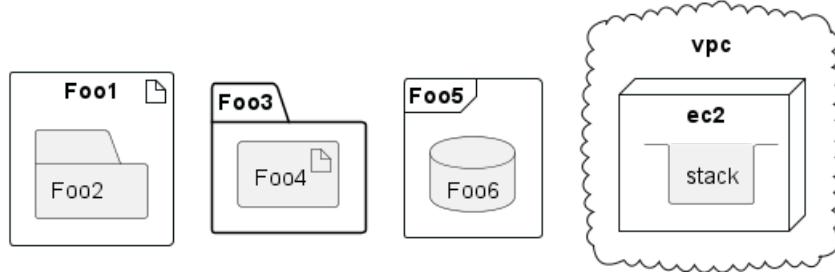
frame Foo5 {
    database Foo6
}

cloud vpc {
    node ec2 {
        stack stack
    }
}

```



```
@enduml
```



```
@startuml
```

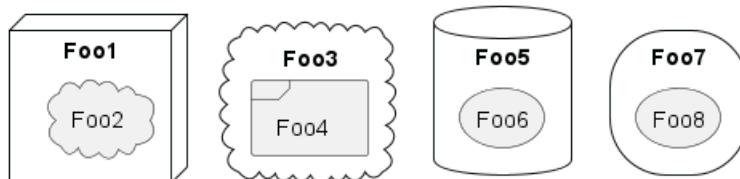
```
node Foo1 {  
    cloud Foo2  
}
```

```
cloud Foo3 {  
    frame Foo4  
}
```

```
database Foo5 {  
    storage Foo6  
}
```

```
storage Foo7 {  
    storage Foo8  
}
```

```
@enduml
```



8.8.3 Full nesting

Here is all the nested elements:

- by alphabetical order:

```
@startuml
```

```
artifact artifact {  
card card {  
cloud cloud {  
component component {  
database database {  
file file {  
folder folder {  
frame frame {  
hexagon hexagon {  
node node {  
package package {  
queue queue {  
rectangle rectangle {  
stack stack {  
storage storage {
```



```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

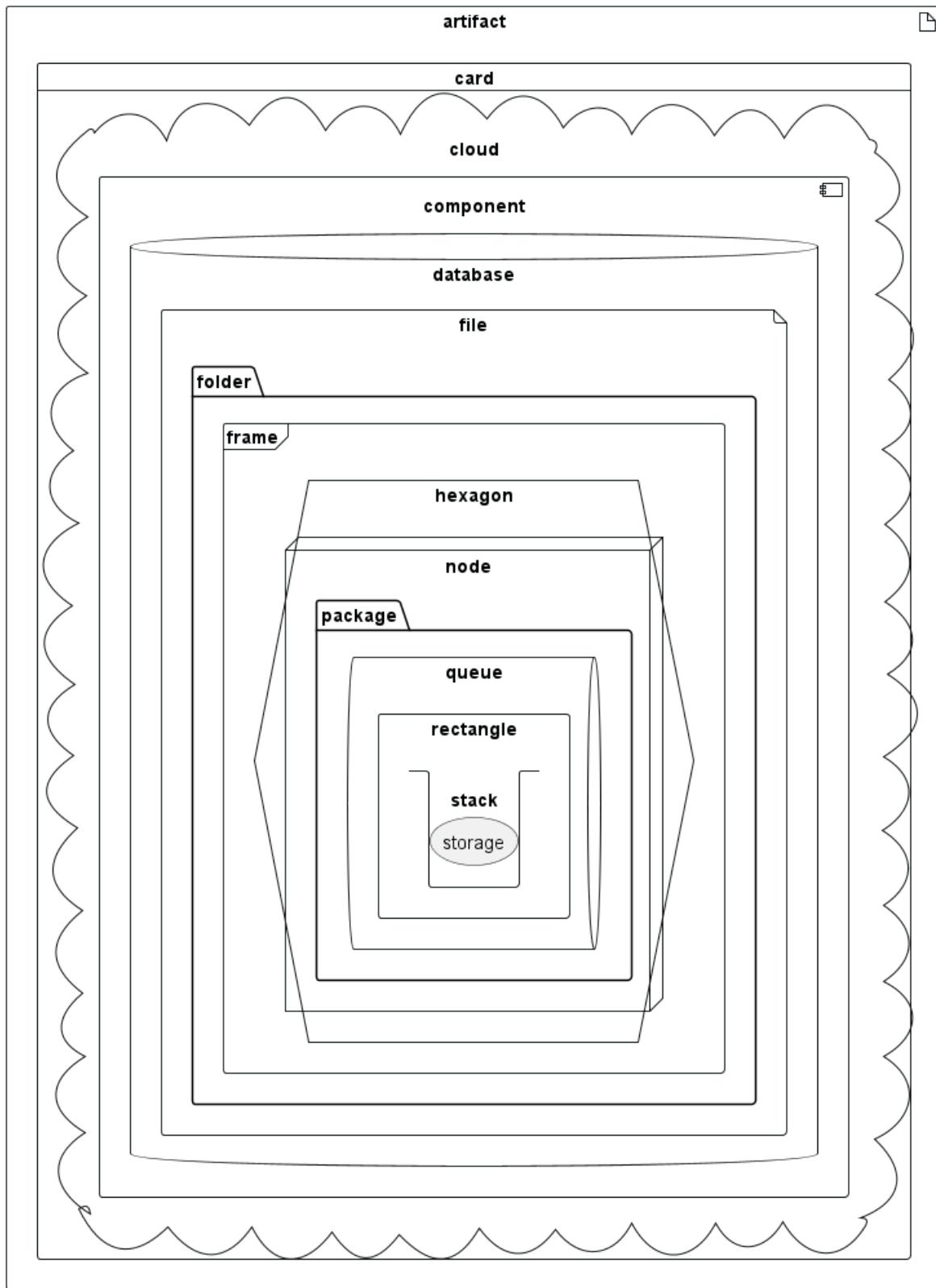
```
}
```

```
}
```

```
}
```

```
@enduml
```



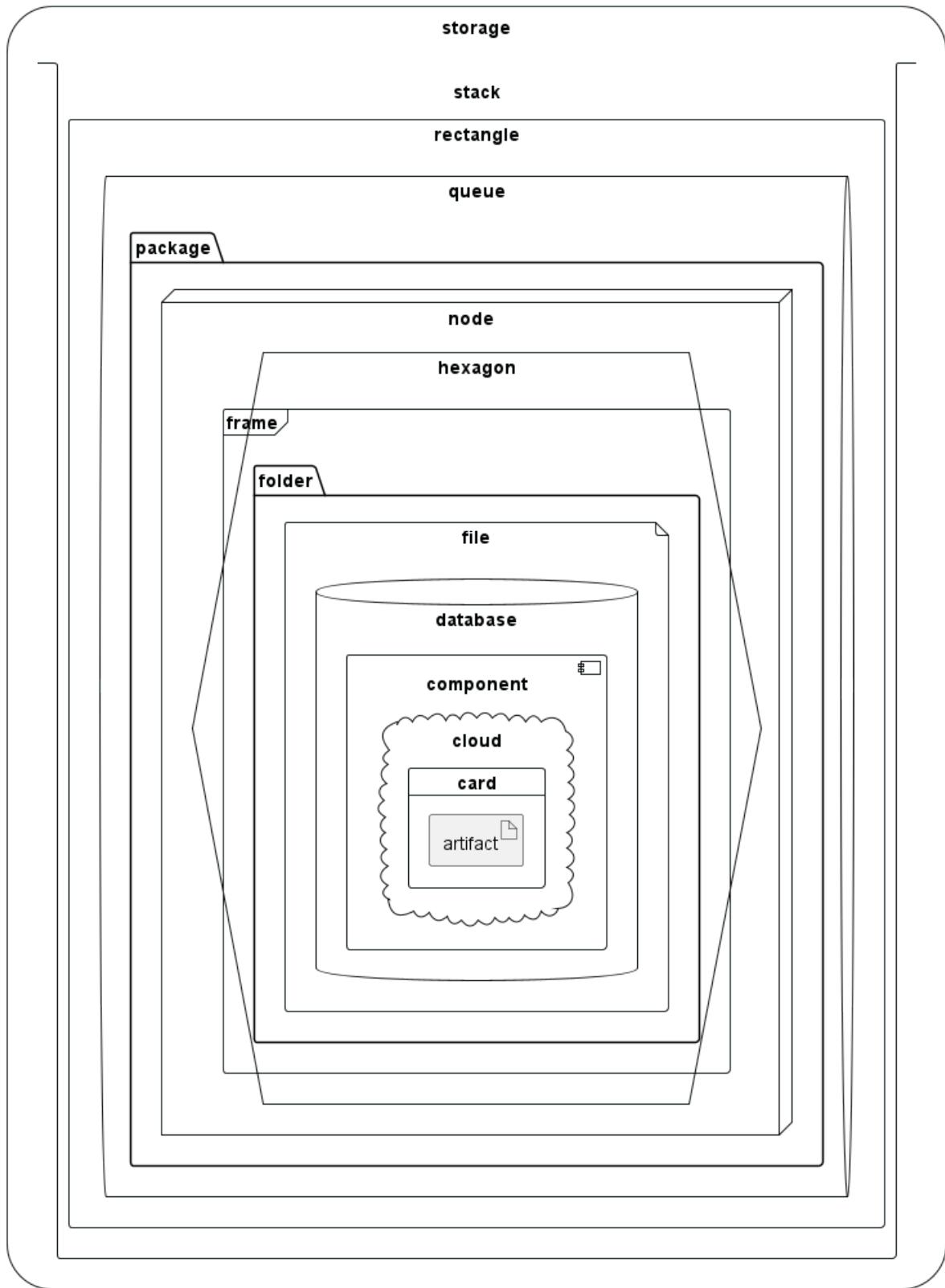


- or reverse alphabetical order

```
@startuml
storage storage {
stack stack {
rectangle rectangle {
queue queue {
package package {
```







8.9 Alias

8.9.1 Simple alias with as

```
@startuml  
node Node1 as n1  
node "Node 2" as n2
```

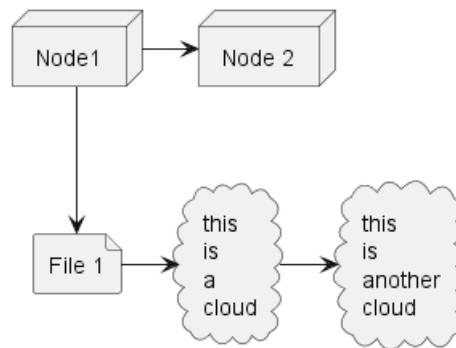


```

file f1 as "File 1"
cloud c1 as "this
is
a
cloud"
cloud c2 [this
is
another
cloud]

n1 -> n2
n1 --> f1
f1 -> c1
c1 -> c2
@enduml

```



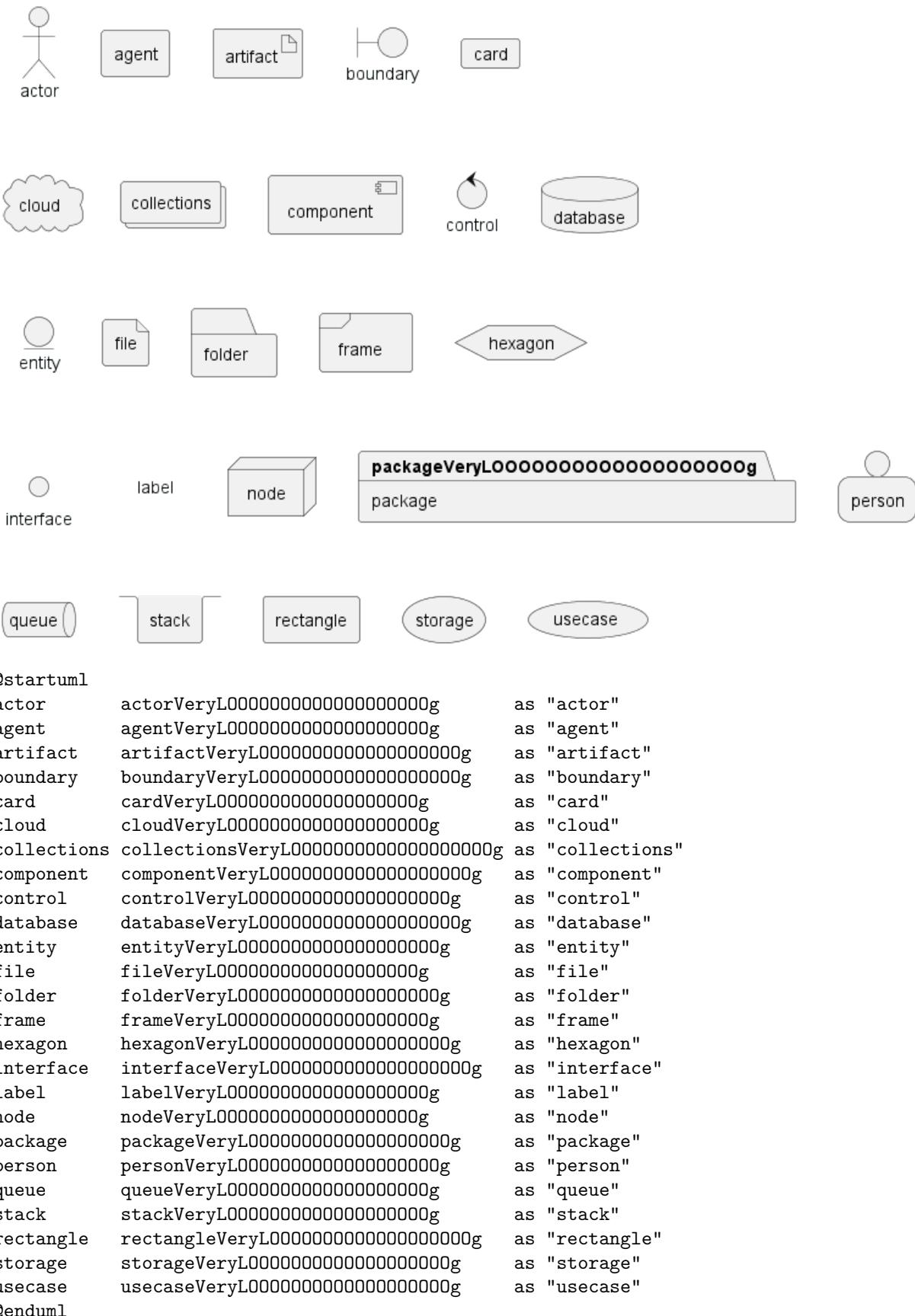
8.9.2 Examples of long alias

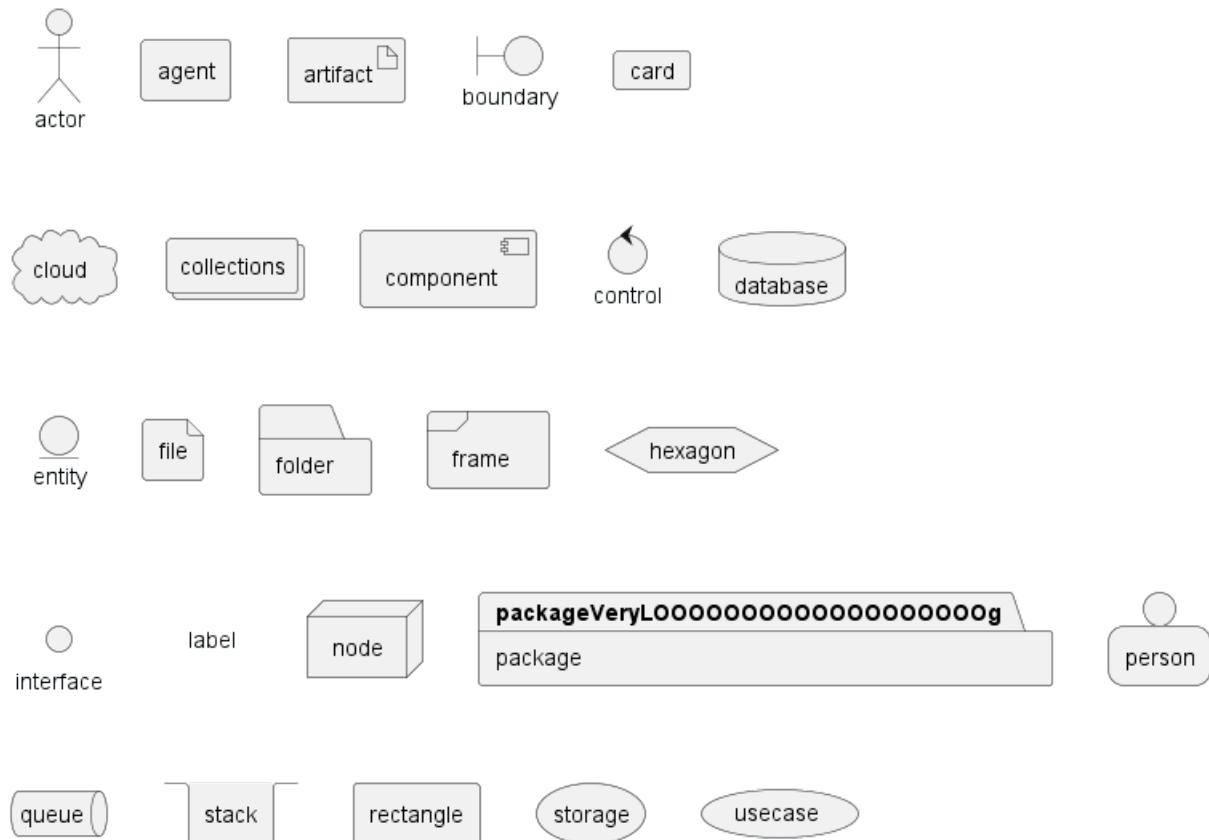
```

@startuml
actor      "actor"      as actorVeryL00000000000000000000g
agent      "agent"       as agentVeryL00000000000000000000g
artifact   "artifact"    as artifactVeryL00000000000000000000g
boundary   "boundary"   as boundaryVeryL00000000000000000000g
card       "card"        as cardVeryL00000000000000000000g
cloud      "cloud"       as cloudVeryL00000000000000000000g
collections "collections" as collectionsVeryL00000000000000000000g
component   "component"  as componentVeryL00000000000000000000g
control     "control"    as controlVeryL00000000000000000000g
database   "database"   as databaseVeryL00000000000000000000g
entity      "entity"     as entityVeryL00000000000000000000g
file        "file"        as fileVeryL00000000000000000000g
folder      "folder"     as folderVeryL00000000000000000000g
frame       "frame"      as frameVeryL00000000000000000000g
hexagon     "hexagon"    as hexagonVeryL00000000000000000000g
interface   "interface"  as interfaceVeryL00000000000000000000g
label       "label"      as labelVeryL00000000000000000000g
node        "node"        as nodeVeryL00000000000000000000g
package     "package"    as packageVeryL00000000000000000000g
person      "person"     as personVeryL00000000000000000000g
queue       "queue"      as queueVeryL00000000000000000000g
stack       "stack"      as stackVeryL00000000000000000000g
rectangle   "rectangle"  as rectangleVeryL00000000000000000000g
storage     "storage"   as storageVeryL00000000000000000000g
usecase     "usecase"   as usecaseVeryL00000000000000000000g
@enduml

```





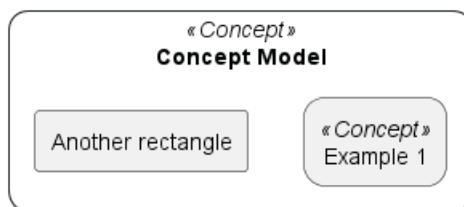


[Ref. QA-12082]

8.10 Round corner

```
@startuml
skinparam rectangle {
    roundCorner<<Concept>> 25
}

rectangle "Concept Model" <<Concept>> {
    rectangle "Example 1" <<Concept>> as ex1
    rectangle "Another rectangle"
}
@enduml
```



8.11 Specific SkinParameter

8.11.1 roundCorner

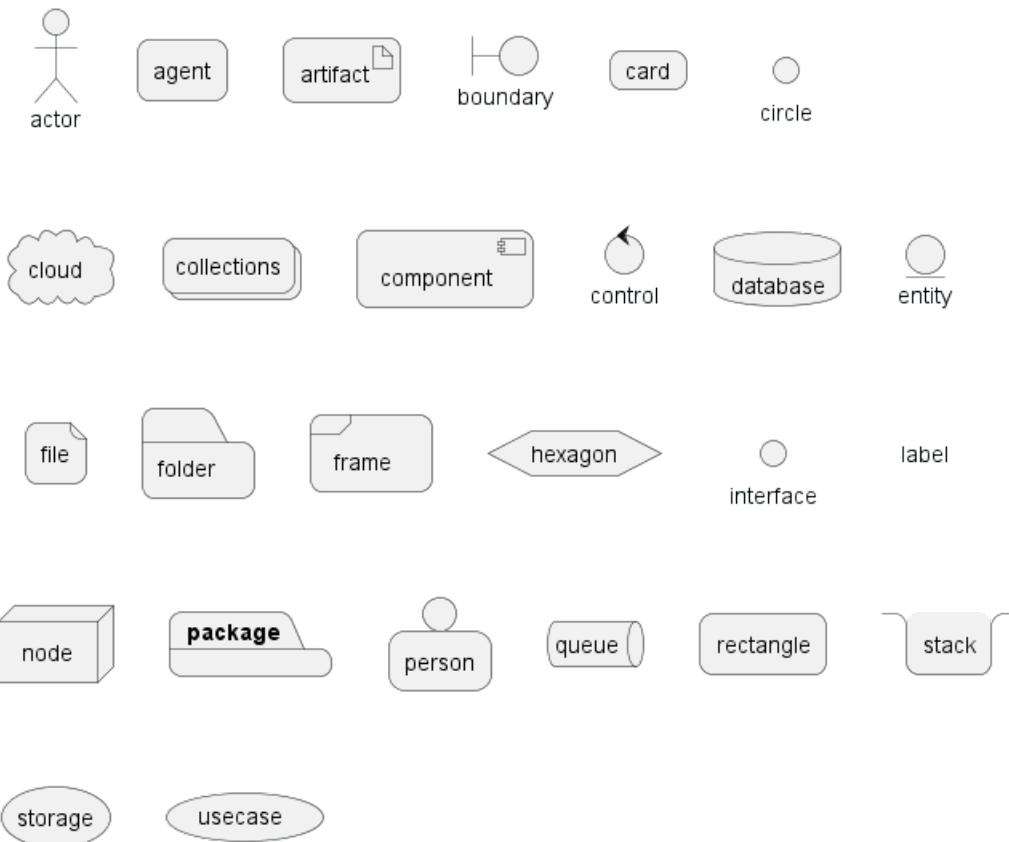
```
@startuml
skinparam roundCorner 15
actor actor
agent agent
artifact artifact
```



```

boundary boundary
card card
circle circle
cloud cloud
collections collections
component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label
node node
package package
person person
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
@enduml

```



8.12 Appendix: All type of arrow line

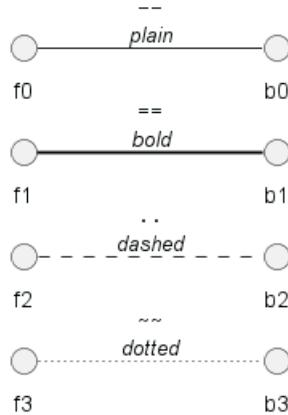
```

@startuml
left to right direction
skinparam nodesep 5

```



```
f3 ~~ b3 : ""~~""\n//dotted//  
f2 .. b2 : ""..""\n//dashed//  
f1 == b1 : ""==""\n//bold//  
f0 -- b0 : ""--""\n//plain//  
@enduml
```

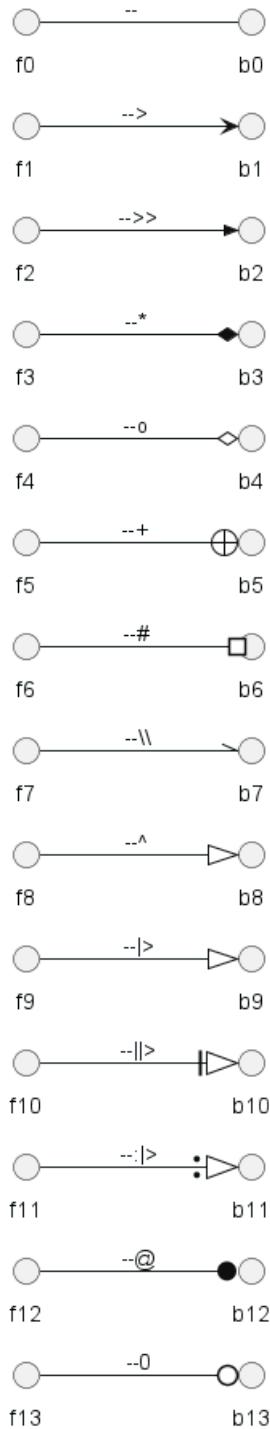


8.13 Appendix: All type of arrow head or '0' arrow

8.13.1 Type of arrow head

```
@startuml  
left to right direction  
skinparam nodesep 5  
  
f13 --0 b13 : ""--0""  
f12 --@ b12 : ""--@""  
f11 --:|> b11 : ""--:|>""  
f10 --||> b10 : ""--||>""  
f9 --|> b9 : ""--|>""  
f8 --^ b8 : ""--^""  
f7 --\\ b7 : ""--\\\""  
f6 --# b6 : ""--#""  
f5 --+ b5 : ""--+""  
f4 --o b4 : ""--o""  
f3 --* b3 : ""--*""  
f2 -->> b2 : ""-->>""  
f1 --> b1 : ""-->""  
f0 -- b0 : ""--""  
@enduml
```





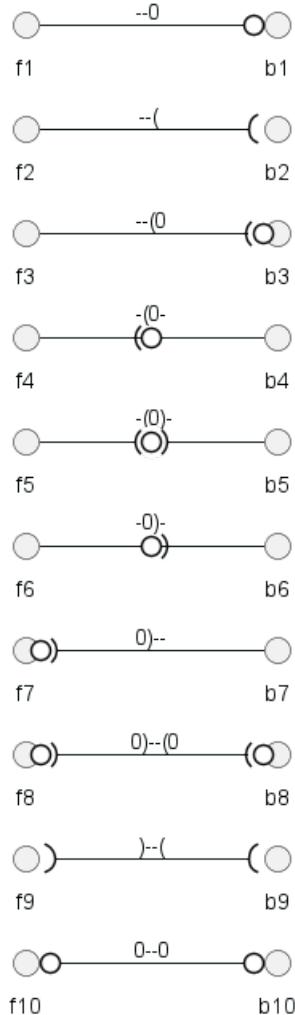
8.13.2 Type of '0' arrow or circle arrow

```
@startuml
left to right direction
skinparam nodesep 5

f10 0--0 b10 : "" 0--0 ""
f9 )--( b9 : "" )--( ""
f8 0)--(0 b8 : "" 0)--(0 ""
f7 0)-- b7 : "" 0)-- ""
f6 -0)- b6 : "" -0)- ""
f5 -(0)- b5 : "" -(0)-""
```



```
f4 -(0- b4 : "" -(0- ""
f3 --(0 b3 : "" --(0 ""
f2 --( b2 : "" --( "
f1 --0 b1 : "" --0 """
@enduml
```



8.14 Appendix: Test of inline style on all element

8.14.1 Simple element

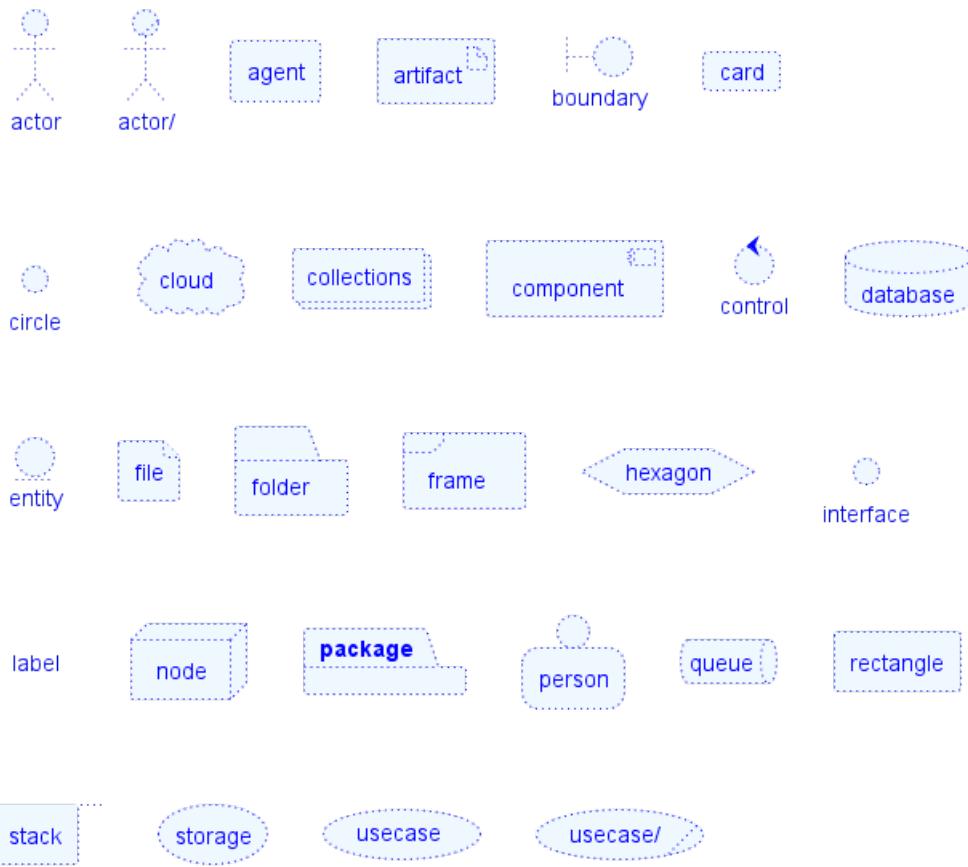
```
@startuml
actor actor          #aliceblue;line:blue;line.dotted;text:blue
actor/ "actor/"      #aliceblue;line:blue;line.dotted;text:blue
agent agent          #aliceblue;line:blue;line.dotted;text:blue
artifact artifact    #aliceblue;line:blue;line.dotted;text:blue
boundary boundary    #aliceblue;line:blue;line.dotted;text:blue
card card            #aliceblue;line:blue;line.dotted;text:blue
circle circle        #aliceblue;line:blue;line.dotted;text:blue
cloud cloud          #aliceblue;line:blue;line.dotted;text:blue
collections collections #aliceblue;line:blue;line.dotted;text:blue
component component  #aliceblue;line:blue;line.dotted;text:blue
control control      #aliceblue;line:blue;line.dotted;text:blue
database database    #aliceblue;line:blue;line.dotted;text:blue
entity entity         #aliceblue;line:blue;line.dotted;text:blue
file file            #aliceblue;line:blue;line.dotted;text:blue
folder folder         #aliceblue;line:blue;line.dotted;text:blue
```



```

frame frame          #aliceblue;line:blue;line.dotted;text:blue
hexagon hexagon     #aliceblue;line:blue;line.dotted;text:blue
interface interface #aliceblue;line:blue;line.dotted;text:blue
label label          #aliceblue;line:blue;line.dotted;text:blue
node node            #aliceblue;line:blue;line.dotted;text:blue
package package      #aliceblue;line:blue;line.dotted;text:blue
person person        #aliceblue;line:blue;line.dotted;text:blue
queue queue          #aliceblue;line:blue;line.dotted;text:blue
rectangle rectangle #aliceblue;line:blue;line.dotted;text:blue
stack stack          #aliceblue;line:blue;line.dotted;text:blue
storage storage      #aliceblue;line:blue;line.dotted;text:blue
usecase usecase      #aliceblue;line:blue;line.dotted;text:blue
usecase/ "usecase/" #aliceblue;line:blue;line.dotted;text:blue
@enduml

```



8.14.2 Nested element

8.14.3 Without sub-element

```

@startuml
artifact artifact #aliceblue;line:blue;line.dotted;text:blue {
}
card card #aliceblue;line:blue;line.dotted;text:blue {
}
cloud cloud #aliceblue;line:blue;line.dotted;text:blue {
}
component component #aliceblue;line:blue;line.dotted;text:blue {
}
database database #aliceblue;line:blue;line.dotted;text:blue {
}
file file #aliceblue;line:blue;line.dotted;text:blue {
}

```



```

}

folder folder #aliceblue;line:blue;line.dotted;text:blue {
}
frame frame #aliceblue;line:blue;line.dotted;text:blue {
}
hexagon hexagon #aliceblue;line:blue;line.dotted;text:blue {
}
node node #aliceblue;line:blue;line.dotted;text:blue {
}
package package #aliceblue;line:blue;line.dotted;text:blue {
}
queue queue #aliceblue;line:blue;line.dotted;text:blue {
}
rectangle rectangle #aliceblue;line:blue;line.dotted;text:blue {
}
stack stack #aliceblue;line:blue;line.dotted;text:blue {
}
storage storage #aliceblue;line:blue;line.dotted;text:blue {
}
@enduml

```



8.14.4 With sub-element

```

@startuml
artifact      artifactVeryL00000000000000000000g      as "artifact" #aliceblue;line:blue;line.dotted;text:
file f1
}
card         cardVeryL00000000000000000000g      as "card" #aliceblue;line:blue;line.dotted;text:blue
file f2
}
cloud        cloudVeryL00000000000000000000g     as "cloud" #aliceblue;line:blue;line.dotted;text:blue
file f3
}
component    componentVeryL00000000000000000000g   as "component" #aliceblue;line:blue;line.dotted;text:
file f4
}
database    databaseVeryL00000000000000000000g     as "database" #aliceblue;line:blue;line.dotted;text:
file f5
}
file         fileVeryL00000000000000000000g      as "file" #aliceblue;line:blue;line.dotted;text:blue
file f6
}
folder        folderVeryL00000000000000000000g    as "folder" #aliceblue;line:blue;line.dotted;text:blue
file f7
}
frame         frameVeryL00000000000000000000g     as "frame" #aliceblue;line:blue;line.dotted;text:blue
file f8
}
hexagon      hexagonVeryL00000000000000000000g    as "hexagon" #aliceblue;line:blue;line.dotted;text:blue
file f9
}
node          nodeVeryL00000000000000000000g      as "node" #aliceblue;line:blue;line.dotted;text:blue
file f10
}
package      packageVeryL00000000000000000000g    as "package" #aliceblue;line:blue;line.dotted;text:blue

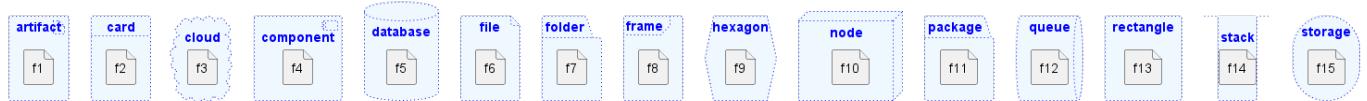
```



```

file f11
}
queue      queueVeryL0000000000000000000g      as "queue" #aliceblue;line:blue;line.dotted;text:bla
file f12
}
rectangle   rectangleVeryL0000000000000000000g   as "rectangle" #aliceblue;line:blue;line.dotted;text:bla
file f13
}
stack       stackVeryL0000000000000000000g     as "stack" #aliceblue;line:blue;line.dotted;text:bla
file f14
}
storage     storageVeryL0000000000000000000g    as "storage" #aliceblue;line:blue;line.dotted;text:bla
file f15
}
@enduml

```



8.15 Appendix: Test of style on all element

8.15.1 Simple element

8.15.2 Global style (on componentDiagram)

```

@startuml
<style>
componentDiagram {
    BackGroundColor palegreen
    LineThickness 1
    LineColor red
}
document {
    BackGroundColor white
}
</style>
actor actor
actor/ "actor/"
agent agent
artifact artifact
boundary boundary
card card
circle circle
cloud cloud
collections collections
component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label
node node
package package

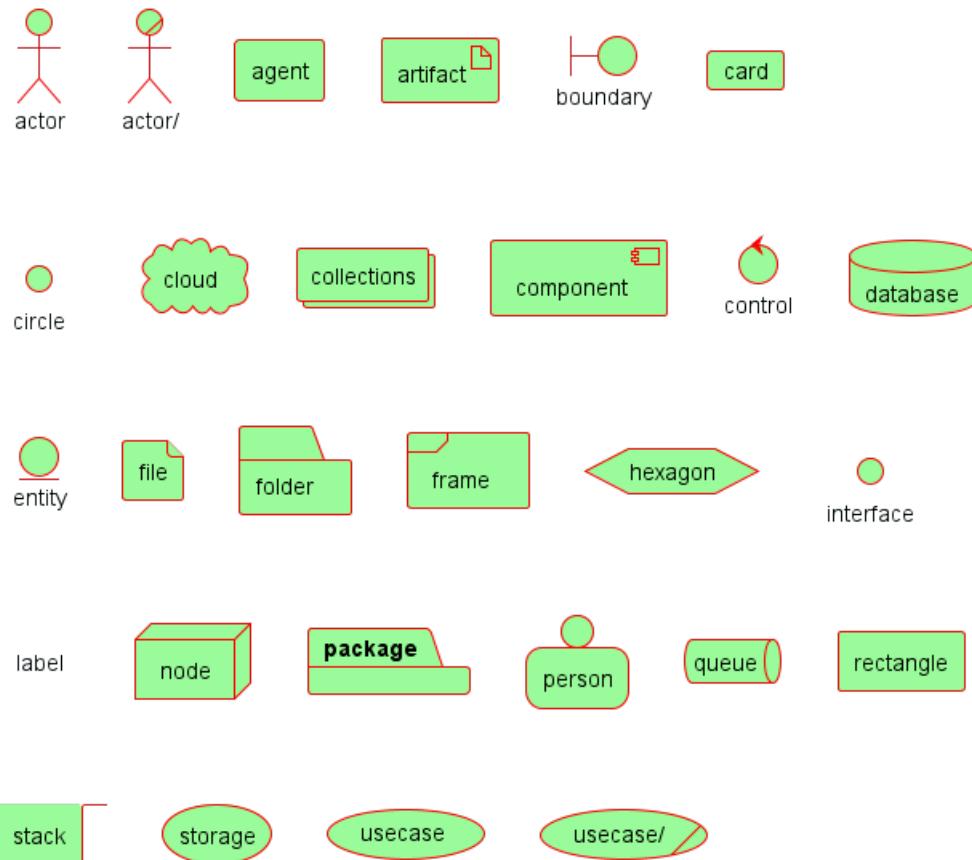
```



```

person person
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
usecase/ "usecase/"
@enduml

```



8.15.3 Style for each element

```

@startuml
<style>
actor {
    BackGroundColor #f80c12
    LineThickness 1
    LineColor black
}
agent {
    BackGroundColor #f80c12
    LineThickness 1
    LineColor black
}
artifact {
    BackGroundColor #ee1100
    LineThickness 1
    LineColor black
}
boundary {
    BackGroundColor #ee1100
    LineThickness 1

```



```
LineColor black
}
card {
    BackGroundColor #ff3311
    LineThickness 1
    LineColor black
}
circle {
    BackGroundColor #ff3311
    LineThickness 1
    LineColor black
}
cloud {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
collections {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
component {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
control {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
database {
    BackGroundColor #ff9933
    LineThickness 1
    LineColor black
}
entity {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
file {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
folder {
    BackGroundColor #ccbb33
    LineThickness 1
    LineColor black
}
frame {
    BackGroundColor #d0c310
    LineThickness 1
    LineColor black
}
hexagon {
```



```

BackGroundColor #aacc22
LineThickness 1
LineColor black
}
interface {
    BackGroundColor #69d025
    LineThickness 1
    LineColor black
}
label {
    BackGroundColor black
    LineThickness 1
    LineColor black
}
node {
    BackGroundColor #22ccaa
    LineThickness 1
    LineColor black
}
package {
    BackGroundColor #12bdb9
    LineThickness 1
    LineColor black
}
person {
    BackGroundColor #11aabb
    LineThickness 1
    LineColor black
}
queue {
    BackGroundColor #11aabb
    LineThickness 1
    LineColor black
}
rectangle {
    BackGroundColor #4444dd
    LineThickness 1
    LineColor black
}
stack {
    BackGroundColor #3311bb
    LineThickness 1
    LineColor black
}
storage {
    BackGroundColor #3b0cbd
    LineThickness 1
    LineColor black
}
usecase {
    BackGroundColor #442299
    LineThickness 1
    LineColor black
}
</style>
actor actor
actor/ "actor/"
agent agent

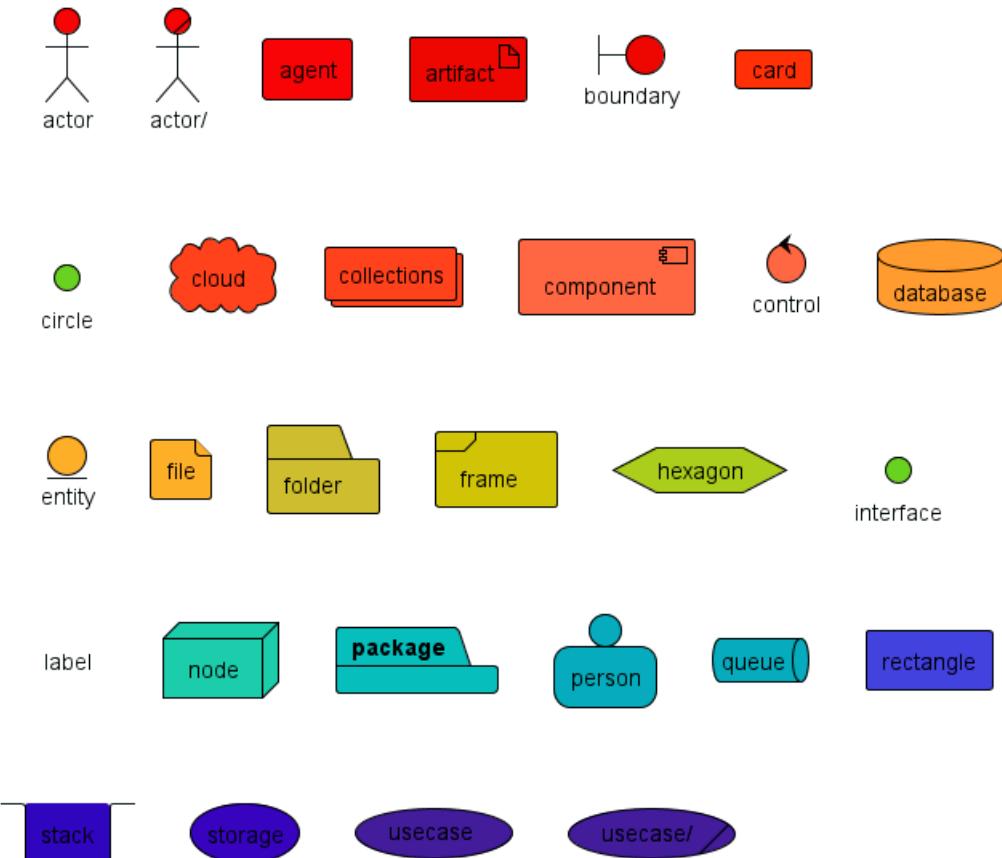
```



```

artifact artifact
boundary boundary
card card
circle circle
cloud cloud
collections collections
component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label
node node
package package
person person
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
usecase/ "usecase/"
@enduml

```



[Ref. QA-13261]



8.15.4 Nested element (without level)

8.15.5 Global style (on componentDiagram)

```
@startuml
<style>
componentDiagram {
    BackGroundColor palegreen
    LineThickness 2
    LineColor red
}
</style>
artifact artifact {
}
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
}
node node {
}
package package {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml
```



8.15.6 Style for each nested element

```
@startuml
<style>
artifact {
    BackGroundColor #ee1100
    LineThickness 1
    LineColor black
}
card {
    BackGroundColor #ff3311
    LineThickness 1
```



```
LineColor black
}
cloud {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
component {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
database {
    BackGroundColor #ff9933
    LineThickness 1
    LineColor black
}
file {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
folder {
    BackGroundColor #ccbb33
    LineThickness 1
    LineColor black
}
frame {
    BackGroundColor #d0c310
    LineThickness 1
    LineColor black
}
hexagon {
    BackGroundColor #aacc22
    LineThickness 1
    LineColor black
}
node {
    BackGroundColor #22ccaa
    LineThickness 1
    LineColor black
}
package {
    BackGroundColor #12bdb9
    LineThickness 1
    LineColor black
}
queue {
    BackGroundColor #11aabb
    LineThickness 1
    LineColor black
}
rectangle {
    BackGroundColor #4444dd
    LineThickness 1
    LineColor black
}
stack {
```



```

BackColor #3311bb
LineThickness 1
LineColor black
}
storage {
    BackGroundColor #3b0cbd
    LineThickness 1
    LineColor black
}

</style>
artifact artifact {
}
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
}
node node {
}
package package {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml

```



8.15.7 Nested element (with one level)

8.15.8 Global style (on componentDiagram)

```

@startuml
<style>
componentDiagram {
    BackGroundColor palegreen
    LineThickness 1
    LineColor red
}
document {
    BackGroundColor white

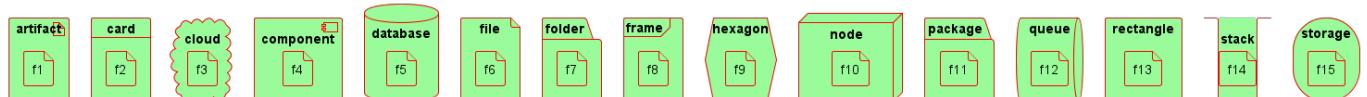
```



```

}
</style>
artifact e1 as "artifact" {
file f1
}
card e2 as "card" {
file f2
}
cloud e3 as "cloud" {
file f3
}
component e4 as "component" {
file f4
}
database e5 as "database" {
file f5
}
file e6 as "file" {
file f6
}
folder e7 as "folder" {
file f7
}
frame e8 as "frame" {
file f8
}
hexagon e9 as "hexagon" {
file f9
}
node e10 as "node" {
file f10
}
package e11 as "package" {
file f11
}
queue e12 as "queue" {
file f12
}
rectangle e13 as "rectangle" {
file f13
}
stack e14 as "stack" {
file f14
}
storage e15 as "storage" {
file f15
}
@enduml

```



8.15.9 Style for each nested element

```
@startuml
<style>
```



```
artifact {
    BackGroundColor #ee1100
    LineThickness 1
    LineColor black
}
card {
    BackGroundColor #ff3311
    LineThickness 1
    LineColor black
}
cloud {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
component {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
database {
    BackGroundColor #ff9933
    LineThickness 1
    LineColor black
}
file {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
folder {
    BackGroundColor #ccbb33
    LineThickness 1
    LineColor black
}
frame {
    BackGroundColor #d0c310
    LineThickness 1
    LineColor black
}
hexagon {
    BackGroundColor #aacc22
    LineThickness 1
    LineColor black
}
node {
    BackGroundColor #22ccaa
    LineThickness 1
    LineColor black
}
package {
    BackGroundColor #12bdb9
    LineThickness 1
    LineColor black
}
queue {
    BackGroundColor #11aabb
    LineThickness 1
```



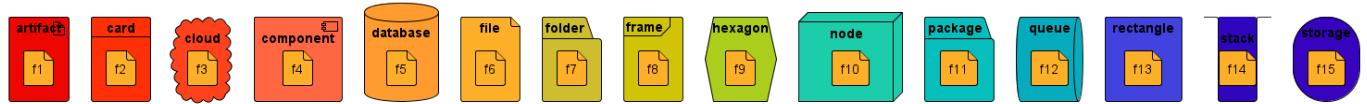
```
    LineColor black
}
rectangle {
    BackGroundColor #4444dd
    LineThickness 1
    LineColor black
}
stack {
    BackGroundColor #3311bb
    LineThickness 1
    LineColor black
}
storage {
    BackGroundColor #3b0cbd
    LineThickness 1
    LineColor black
}
</style>
artifact e1 as "artifact" {
file f1
}
card e2 as "card" {
file f2
}
cloud e3 as "cloud" {
file f3
}
component e4 as "component" {
file f4
}
database e5 as "database" {
file f5
}
file e6 as "file" {
file f6
}
folder e7 as "folder" {
file f7
}
frame e8 as "frame" {
file f8
}
hexagon e9 as "hexagon" {
file f9
}
node e10 as "node" {
file f10
}
package e11 as "package" {
file f11
}
queue e12 as "queue" {
file f12
}
rectangle e13 as "rectangle" {
file f13
}
stack e14 as "stack" {
```



```

file f14
}
storage e15 as "storage" {
file f15
}
}
@enduml

```



8.16 Appendix: Test of stereotype with style on all element

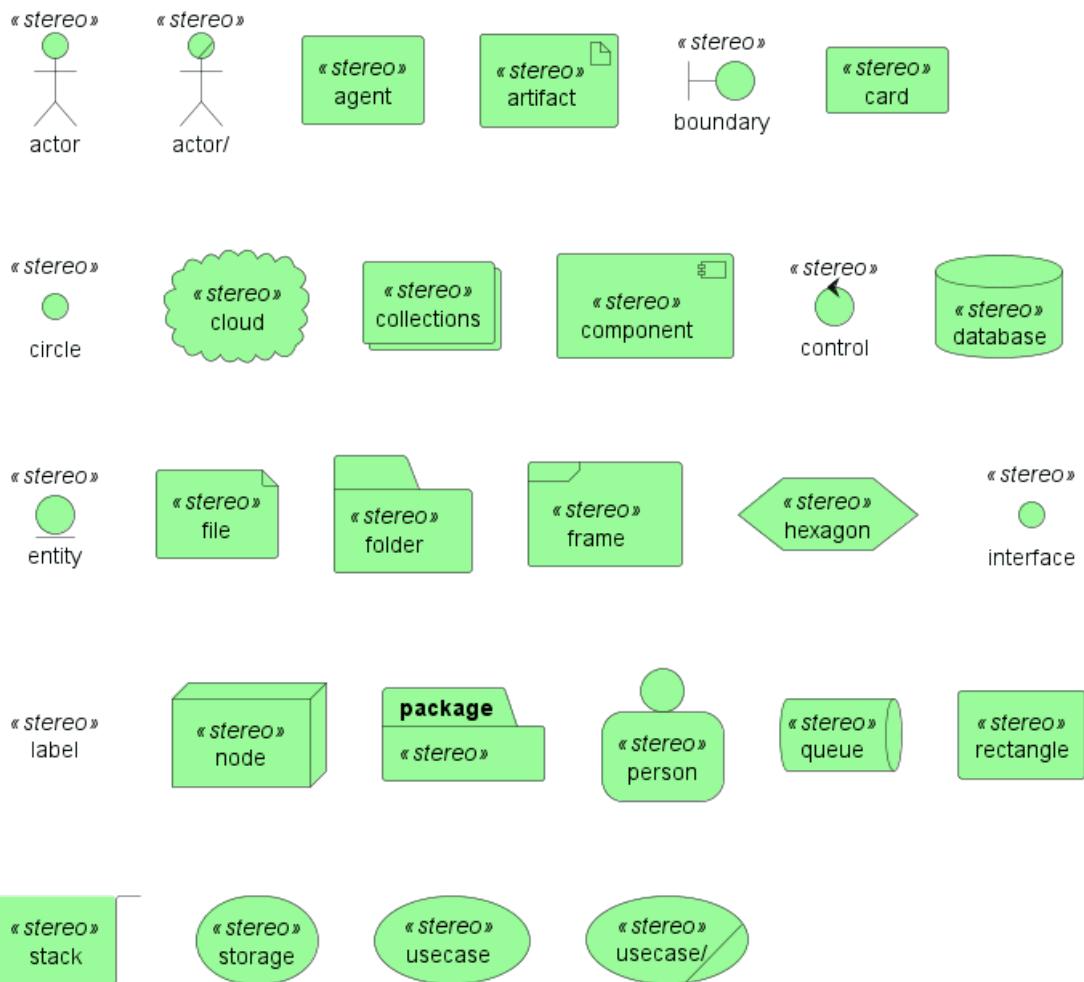
8.16.1 Simple element

```

@startuml
<style>
.stereo {
    BackgroundColor palegreen
}
</style>
actor actor << stereo >>
actor/ "actor/" << stereo >>
agent agent << stereo >>
artifact artifact << stereo >>
boundary boundary << stereo >>
card card << stereo >>
circle circle << stereo >>
cloud cloud << stereo >>
collections collections << stereo >>
component component << stereo >>
control control << stereo >>
database database << stereo >>
entity entity << stereo >>
file file << stereo >>
folder folder << stereo >>
frame frame << stereo >>
hexagon hexagon << stereo >>
interface interface << stereo >>
label label << stereo >>
node node << stereo >>
package package << stereo >>
person person << stereo >>
queue queue << stereo >>
rectangle rectangle << stereo >>
stack stack << stereo >>
storage storage << stereo >>
usecase usecase << stereo >>
usecase/ "usecase/" << stereo >>
@enduml

```





8.17 Display JSON Data on Deployment diagram

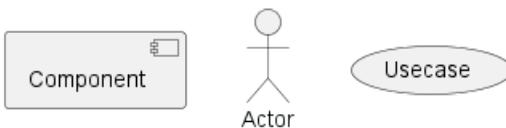
8.17.1 Simple example

```
@startuml
allowmixing
```

```
component Component
actor Actor
usecase Usecase
() Interface
node Node
cloud Cloud
```

```
json JSON {
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}
@enduml
```





JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

For another example, see on JSON page.

8.18 Mixing Deployment (Usecase, Component, Deployment) element within a Class or Object diagram

In order to add a Deployment element or a State element within a Class or Object diagram, you can use the `allowmixing` or `allow_mixing` directive.

8.18.1 Mixing all elements

```
@startuml
```

```
allowmixing
```

```

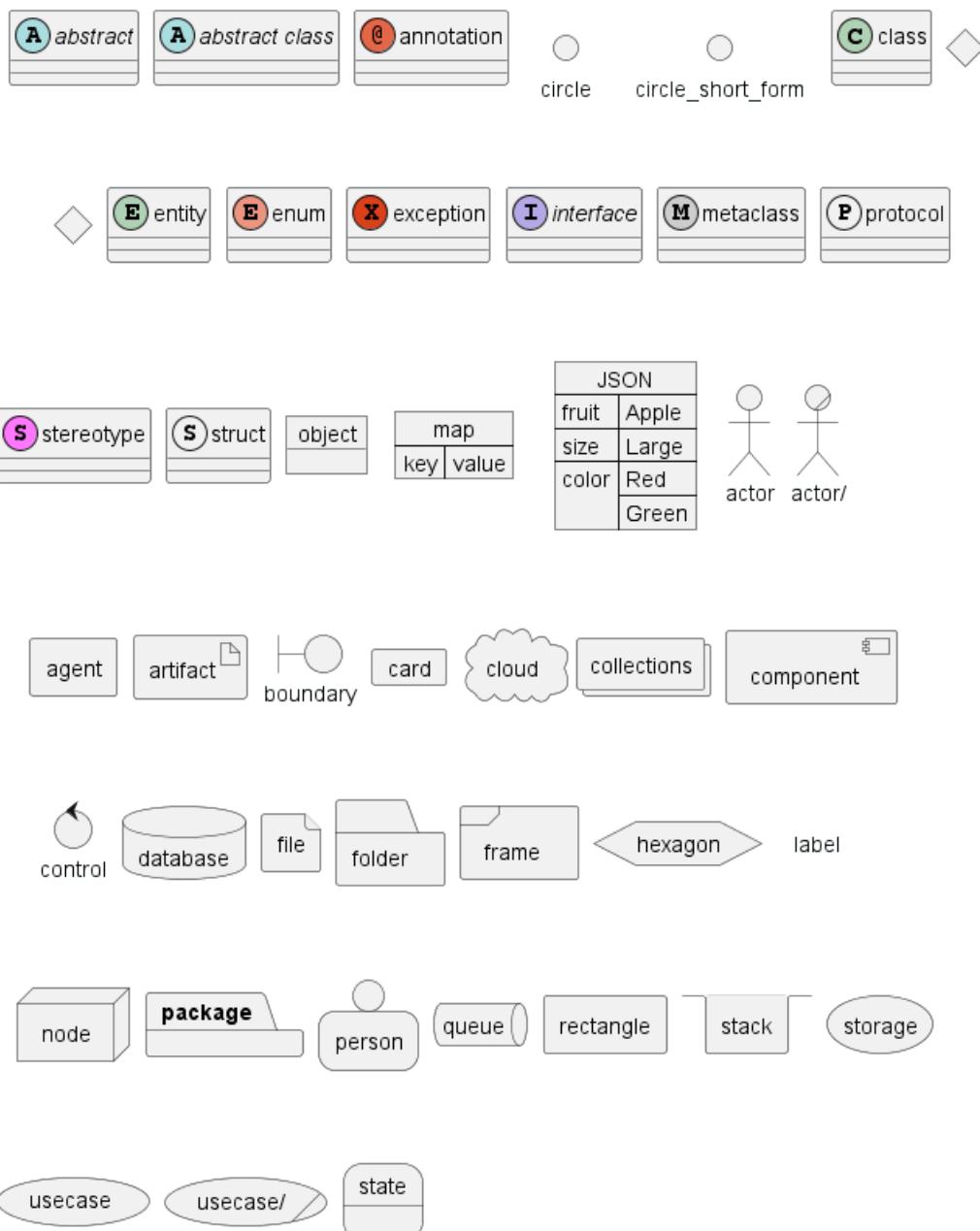
skinparam nodesep 10
abstract      abstract
abstract class "abstract class"
annotation    annotation
circle        circle
()            circle_short_form
class         class
diamond       diamond
<>           diamond_short_form
entity        entity
enum          enum
exception     exception
interface     interface
metaclass    metaclass
protocol      protocol
stereotype   stereotype
struct        struct
object        object
map map {
  key => value
}
json JSON {
  "fruit":"Apple",

```



```
"size":"Large",
"color": ["Red", "Green"]
}
actor actor
actor/ "actor/"
agent agent
artifact artifact
boundary boundary
card card
circle circle
cloud cloud
collections collections
component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label
node node
package package
person person
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
usecase/ "usecase/"
state state
@enduml
```





[Ref. QA-2335 and QA-5329]

8.19 Port [port, portIn, portOut]

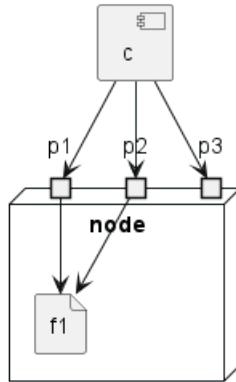
You can added **port** with **port**, **portin** and **portout** keywords.

8.19.1 Port

```
@startuml
[c]
node node {
    port p1
    port p2
    port p3
    file f1
}
c --> p1
```



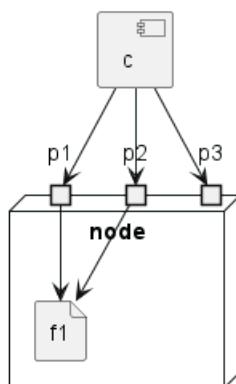
```
c --> p2
c --> p3
p1 --> f1
p2 --> f1
@enduml
```



8.19.2 PortIn

```
@startuml
[c]
node node {
    portin p1
    portin p2
    portin p3
    file f1
}

c --> p1
c --> p2
c --> p3
p1 --> f1
p2 --> f1
@enduml
```



8.19.3 PortOut

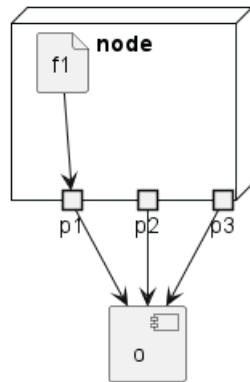
```
@startuml
node node {
    portout p1
    portout p2
    portout p3
    file f1
```



```

}
[o]
p1 --> o
p2 --> o
p3 --> o
f1 --> p1
@enduml

```



8.19.4 Mixing PortIn & PortOut

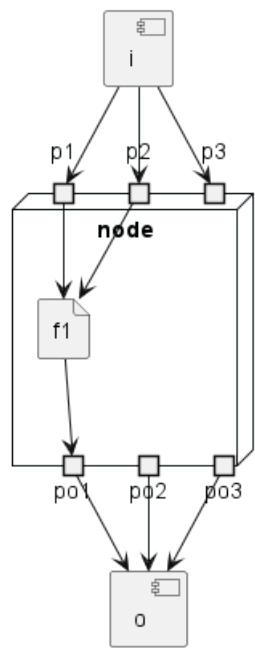
```

@startuml
[i]
node node {
    portin p1
    portin p2
    portin p3
    portout po1
    portout po2
    portout po3
    file f1
}
[o]

i --> p1
i --> p2
i --> p3
p1 --> f1
p2 --> f1
po1 --> o
po2 --> o
po3 --> o
f1 --> po1
@enduml

```





9 Diagrama de estado

Los diagramas de estados son usados para dar una descripción abstracta de el comportamiento de un sistema. Este comportamiento es representado como una serie de eventos que pueden ocurrir en uno o más estados posibles.

Using **PlantUML** to create state diagrams offers several advantages:

- **Text-Based Language:** Quickly define and visualize the states and transitions without the hassle of manual drawing.
- **Efficiency and Consistency:** Ensure streamlined diagram creation and easy version control.
- **Versatility:** Integrate with various documentation platforms and support multiple output formats.
- **Open-Source & Community Support:** Backed by a **strong community** that continuously contributes to its enhancements and offers invaluable resources.

9.1 Estado simple

Puedes usar [*] para el punto de inicio y finalización del diagrama de estados.

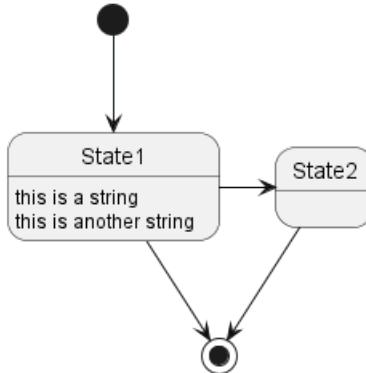
Utilice --> para las flechas.

```
@startuml
```

```
[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string

State1 -> State2
State2 --> [*]
```

```
@enduml
```



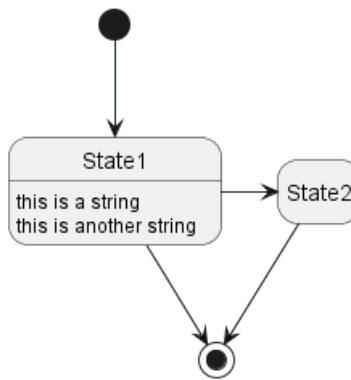
9.2 Cambiar la representación del estado

Puede utilizar `hide empty description` para representar el estado como una caja simple.

```
@startuml
hide empty description
[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string

State1 -> State2
State2 --> [*]
@enduml
```





9.3 Estados compuestos

Un estado también puede ser compuesto. Puedes definirlo usando la palabra reservada `state` y llaves.

```
@startuml
scale 350 width
[*] --> NotShooting

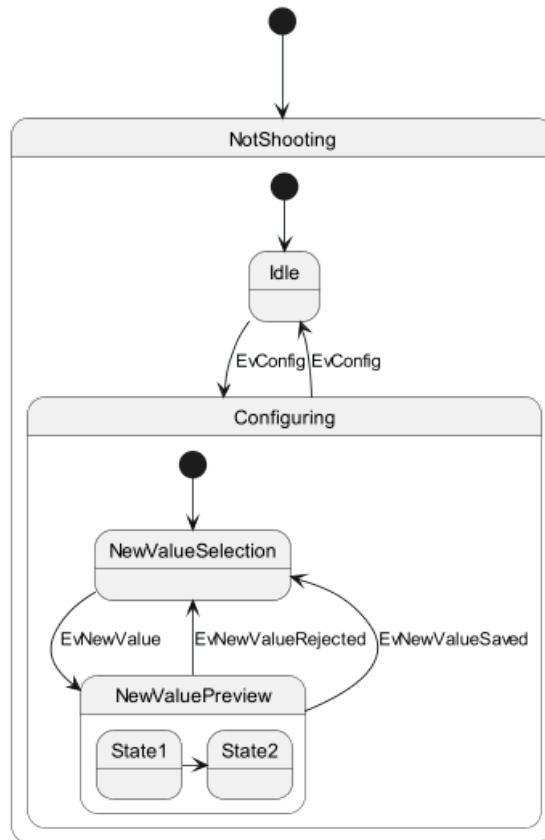
state NotShooting {
    [*] --> Idle
    Idle --> Configuring : EvConfig
    Configuring --> Idle : EvConfig
}

state Configuring {
    [*] --> NewValueSelection
    NewValueSelection --> NewValuePreview : EvnewValue
    NewValuePreview --> NewValueSelection : EvnewValueRejected
    NewValuePreview --> NewValueSelection : EvnewValueSaved

    state NewValuePreview {
        State1 -> State2
    }
}

}
@enduml
```





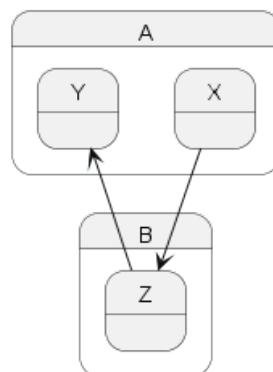
9.3.1 Sub-estado a sub-estado

```

@startuml
state A {
    state X {
        }
    state Y {
        }
}
state B {
    state Z {
        }
}

X --> Z
Z --> Y
@enduml

```



[Ref. QA-3300]

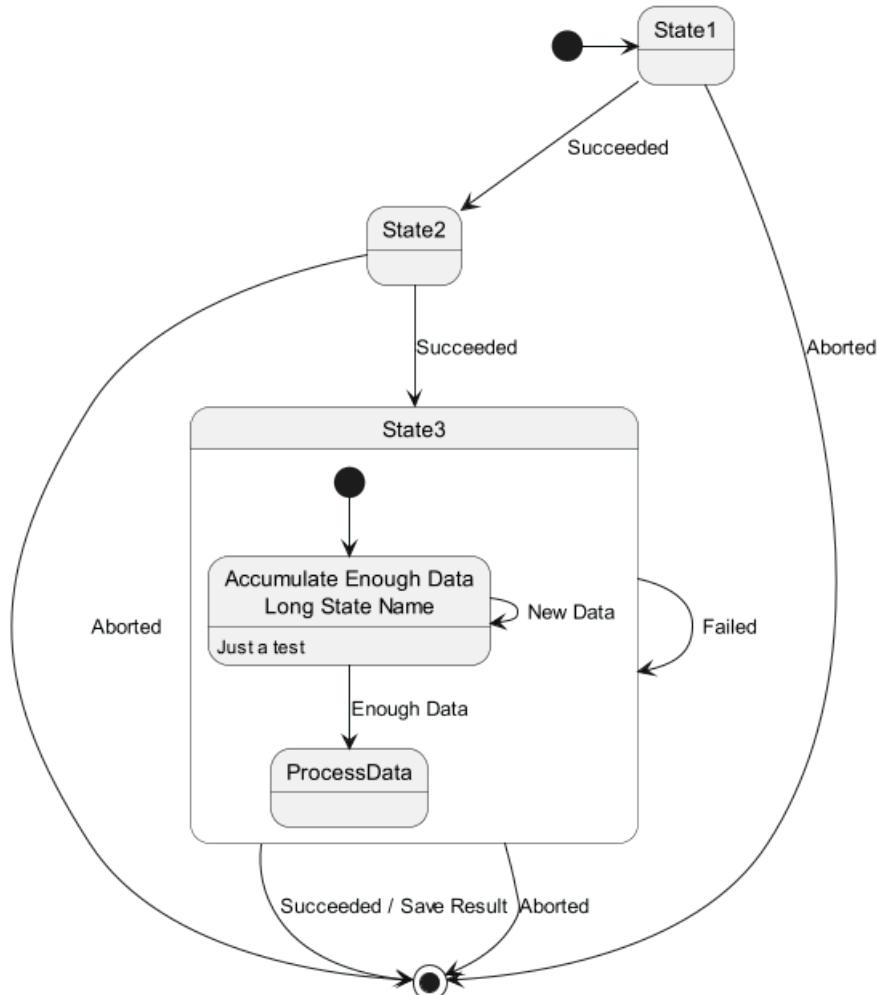
9.4 Nombres largos

También puedes usar la palabra reservada `state` para definir nombres largos en un estado.

```
@startuml
scale 600 width
```

```
[*] --> State1
State1 --> State2 : Succeeded
State1 --> [*] : Aborted
State2 --> State3 : Succeeded
State2 --> [*] : Aborted
state State3 {
    state "Accumulate Enough Data\nLong State Name" as long1
    long1 : Just a test
    [*] --> long1
    long1 --> long1 : New Data
    long1 --> ProcessData : Enough Data
}
State3 --> State3 : Failed
State3 --> [*] : Succeeded / Save Result
State3 --> [*] : Aborted
```

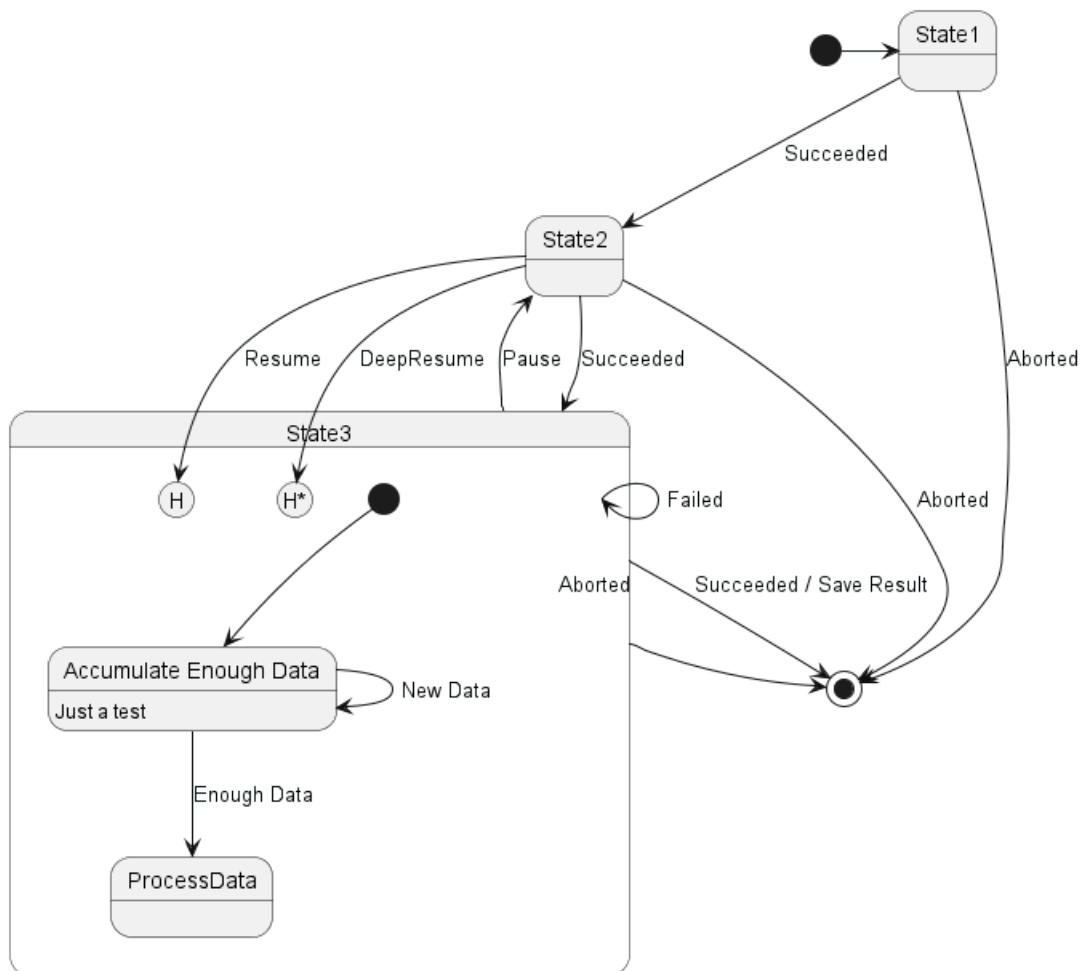
```
@enduml
```



9.5 Histórico [[H], [H*]]

Puedes usar [H] para el histórico y [H*] para el histórico profundo de un subestado.

```
@startuml
[*] --> State1
State1 --> State2 : Succeeded
State1 --> [*] : Aborted
State2 --> State3 : Succeeded
State2 --> [*] : Aborted
state State3 {
    state "Accumulate Enough Data" as long1
    long1 : Just a test
    [*] --> long1
    long1 --> long1 : New Data
    long1 --> ProcessData : Enough Data
    State2 --> [H] : Resume
}
State3 --> State2 : Pause
State2 --> State3[H*]: DeepResume
State3 --> State3 : Failed
State3 --> [*] : Succeeded / Save Result
State3 --> [*] : Aborted
@enduml
```



9.6 Bifurcación

También puedes bifurcar y unir usando <<fork>> y <<join>> respectivamente.



@startuml

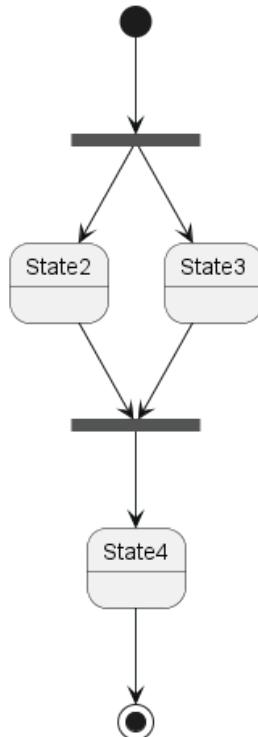
```

state fork_state <<fork>>
[*] --> fork_state
fork_state --> State2
fork_state --> State3

state join_state <<join>>
State2 --> join_state
State3 --> join_state
join_state --> State4
State4 --> [*]

```

@enduml



9.7 Estados concurrentes [-, ||]

Puedes definir estados concurrentes dentro de un estado compuesto usando los símbolos -- o || como separadores.

9.7.1 Separador horizontal --

```
@startuml
[*] --> Active
```

```

state Active {
    [*] -> NumLockOff
    NumLockOff --> NumLockOn : EvNumLockPressed
    NumLockOn --> NumLockOff : EvNumLockPressed
    --
    [*] -> CapsLockOff
    CapsLockOff --> CapsLockOn : EvCapsLockPressed
    CapsLockOn --> CapsLockOff : EvCapsLockPressed
    --
    [*] -> ScrollLockOff

```

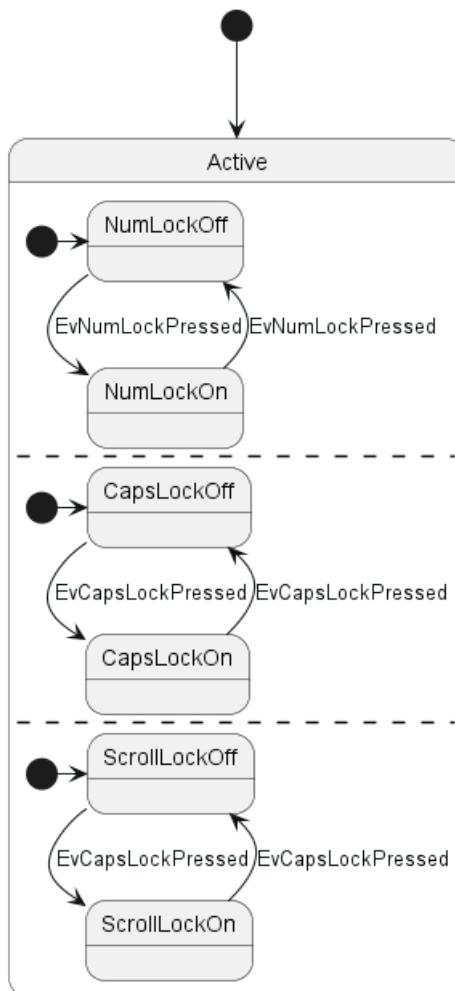


```

ScrollLockOff --> ScrollLockOn : EvCapsLockPressed
ScrollLockOn --> ScrollLockOff : EvCapsLockPressed
}

@enduml

```



9.7.2 Separador vertical ||

```

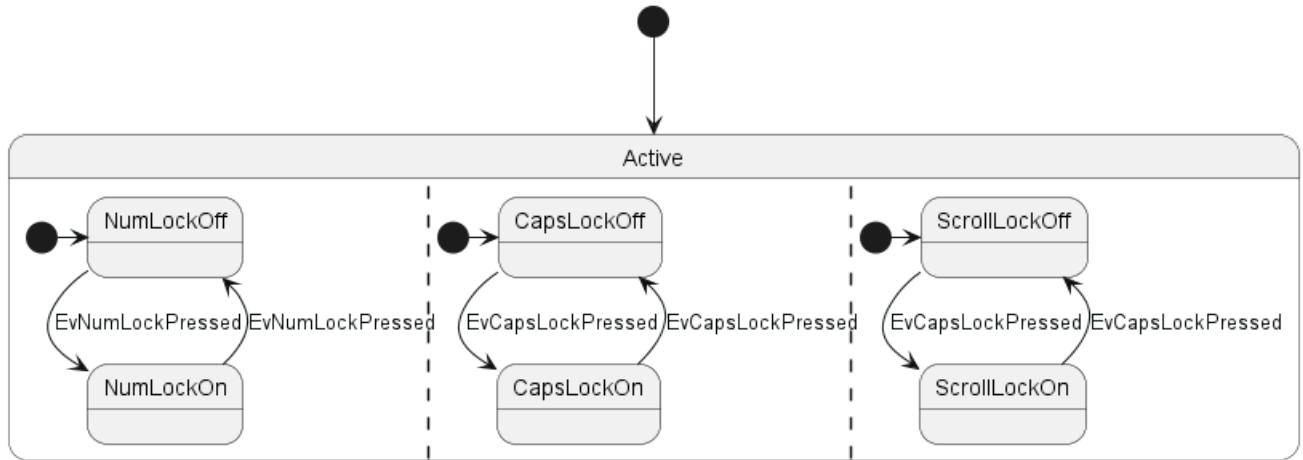
@startuml
[*] --> Active

state Active {
    [*] -> NumLockOff
    NumLockOff --> NumLockOn : EvNumLockPressed
    NumLockOn --> NumLockOff : EvNumLockPressed
    ||
    [*] -> CapsLockOff
    CapsLockOff --> CapsLockOn : EvCapsLockPressed
    CapsLockOn --> CapsLockOff : EvCapsLockPressed
    ||
    [*] -> ScrollLockOff
    ScrollLockOff --> ScrollLockOn : EvCapsLockPressed
    ScrollLockOn --> ScrollLockOff : EvCapsLockPressed
}

```

@enduml





[Ref. QA-3086]

9.8 Condicional [choice]

El estereotipo <<choice>> puede ser usado para uso de estado condicional.

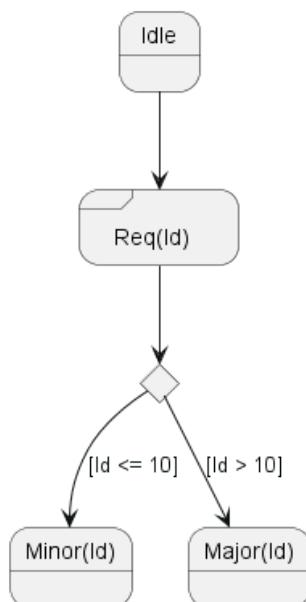
```

@startuml
state "Req(Id)" as ReqId <<sdlreceive>>
state "Minor(Id)" as MinorId
state "Major(Id)" as MajorId

state c <<choice>>

Idle --> ReqId
ReqId --> c
c --> MinorId : [Id <= 10]
c --> MajorId : [Id > 10]
@enduml

```



9.9 Ejemplo completo de estereotipos [start, choice, fork, join, end]

```

@startuml
state start1  <<start>>

```



```

state choice1 <<choice>>
state fork1    <<fork>>
state join2    <<join>>
state end3     <<end>>

[*]      --> choice1 : from start\nto choice
start1 --> choice1 : from start stereo\nto choice

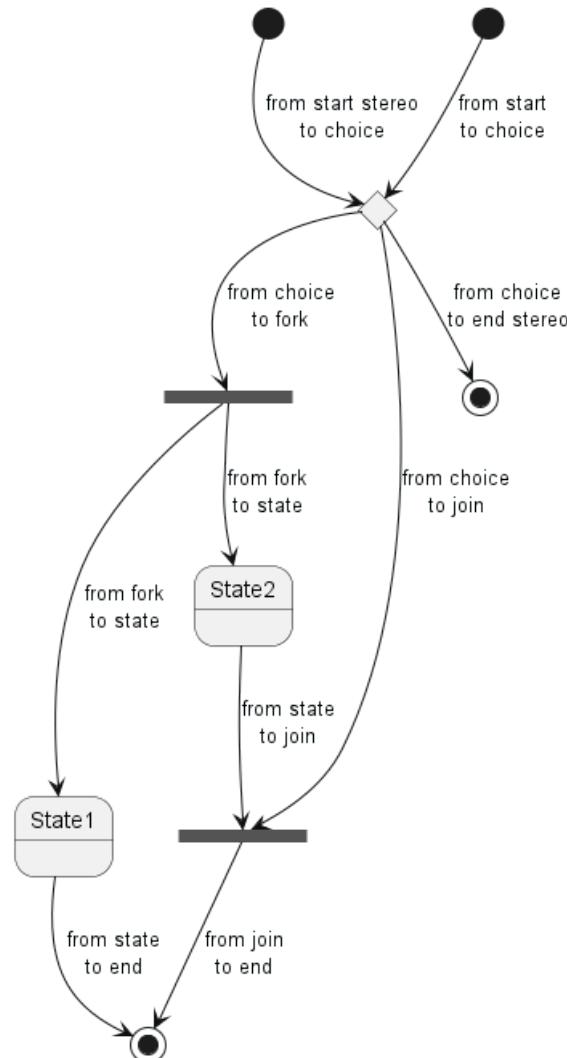
choice1 --> fork1    : from choice\nto fork
choice1 --> join2    : from choice\nto join
choice1 --> end3     : from choice\nto end stereo

fork1    ---> State1 : from fork\nto state
fork1    --> State2  : from fork\nto state

State2 --> join2    : from state\nto join
State1 --> [*]       : from state\nto end

join2    --> [*]       : from join\nto end
@enduml

```



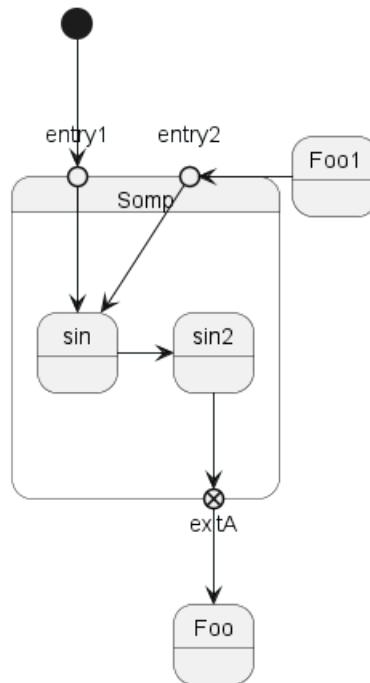
[Ref. QA-404, QA-1159 and GH-887]

9.10 Punto [entryPoint, exitPoint]

Puedes agregar **punto** con los esterotipos <<entryPoint>> y <<exitPoint>>:

```
@startuml
state Somp {
    state entry1 <<entryPoint>>
    state entry2 <<entryPoint>>
    state sin
    entry1 --> sin
    entry2 -> sin
    sin -> sin2
    sin2 --> exitA <<exitPoint>>
}

[*] --> entry1
exitA --> Foo
Foo1 -> entry2
@enduml
```



9.11 Pin [inputPin, outputPin]

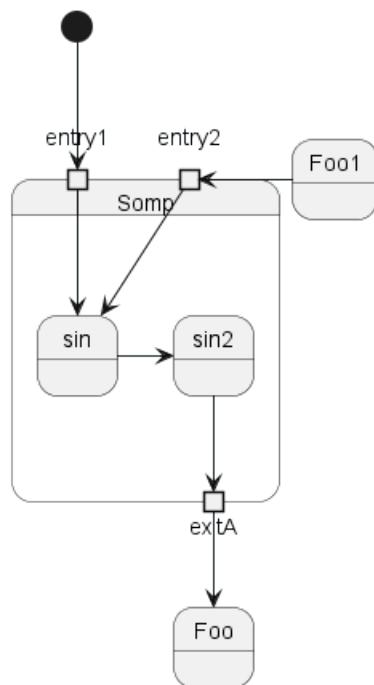
Puedes agregar **pin** con los estereotipos <<inputPin>> y <<outputPin>>:

```
@startuml
state Somp {
    state entry1 <<inputPin>>
    state entry2 <<inputPin>>
    state sin
    entry1 --> sin
    entry2 -> sin
    sin -> sin2
    sin2 --> exitA <<outputPin>>
}

[*] --> entry1
exitA --> Foo
Foo1 -> entry2
```



@enduml



[Ref. QA-4309]

9.12 Expansión [expansionInput, expansionOutput]

Puedes agregar **expansión** con los estereotipos <<expansionInput>> y <<expansionOutput>>:

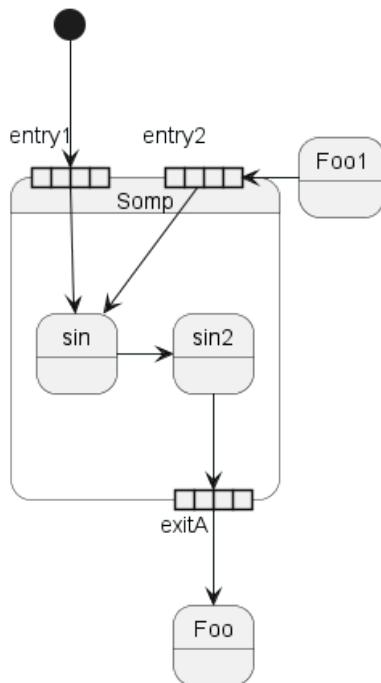
```

@startuml
state Somp {
    state entry1 <<expansionInput>>
    state entry2 <<expansionInput>>
    state sin
    entry1 --> sin
    entry2 -> sin
    sin -> sin2
    sin2 --> exitA <<expansionOutput>>
}

[*] --> entry1
exitA --> Foo
Foo1 -> entry2
@enduml

```





[Ref. QA-4309]

9.13 Dirección de flecha

Puedes usar `->` para flechas horizontales. Es posible forzar la dirección de las flechas usando la siguiente sintaxis:

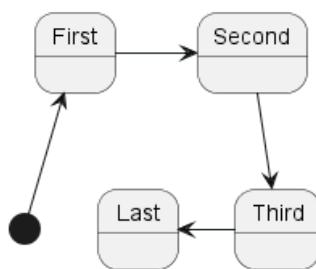
- `-down->` o `-->`
- `-right->` o `->` (flecha por defecto)
- `-left->`
- `-up->`

`@startuml`

```

[*] -up-> First
First -right-> Second
Second --> Third
Third -left-> Last
  
```

`@enduml`



Puedes acortar la definición de la flecha usando sólamente el primer carácter del nombre de la dirección (por ejemplo, `-d-` en lugar de `-down-`) o los dos primeros caracteres (`-do-`).

Por favor tenga en cuenta que no debería abusar de esta esta funcionalidad : *Graphviz* usualmente devuelve buenos resultados sin necesidad de ajustes.

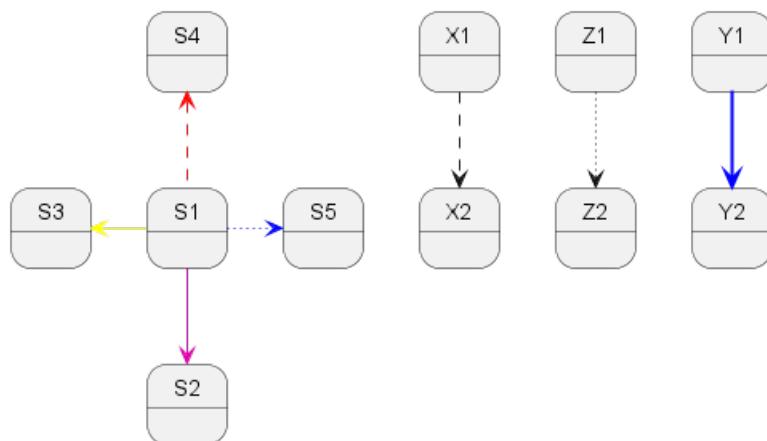


9.14 Cambiar el color y estilo de la línea

Puedes cambiar el color y/o el estilo de la línea.

```
@startuml
State S1
State S2
State S3
S1 -[#DD00AA]-> S2
S1 -left[#yellow]-> S3
S1 -up[#red,dashed]-> S4
S1 -right[dotted,#blue]-> S5
```

```
X1 -[dashed]-> X2
Z1 -[dotted]-> Z2
Y1 -[#blue,bold]-> Y2
@enduml
```



[Ref. Incubation: Change line color in state diagrams]

9.15 Notas

También puedes definir notas usando las palabras reservadas `note left of`, `note right of`, `note top of`, `note bottom of`.

También puedes definir notas de varias líneas.

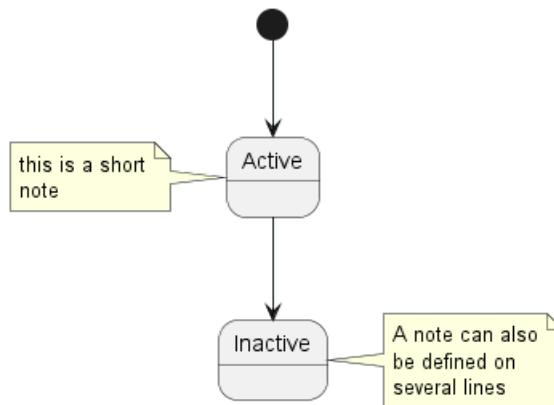
```
@startuml
[*] --> Active
Active --> Inactive

note left of Active : this is a short\nnote

note right of Inactive
  A note can also
  be defined on
  several lines
end note

@enduml
```





También puedes tener notas flotantes.

@startuml

```

state foo
note "This is a floating note" as N1
  
```

@enduml

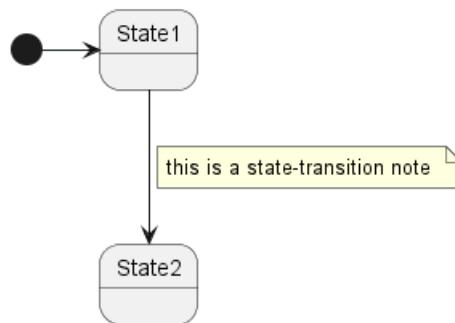


9.16 Notas sobre conexiones

Puedes poner notas sobre la transición de estados o conexiones, con la palabra reservada `note on link`.

```

@startuml
[*] --> State1
State1 --> State2
note on link
    this is a state-transition note
end note
@enduml
  
```



9.17 Más sobre notas

Puedes colocar notas en estados compuestos.

@startuml

```

[*] --> NotShooting

state "Not Shooting State" as NotShooting {
    state "Idle mode" as Idle
    state "Configuring mode" as Configuring
  
```



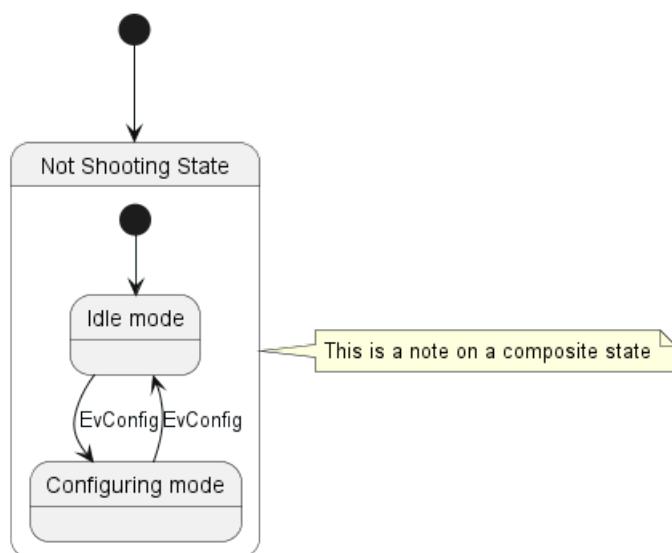
```

[*] --> Idle
Idle --> Configuring : EvConfig
Configuring --> Idle : EvConfig
}

note right of NotShooting : This is a note on a composite state

@enduml

```



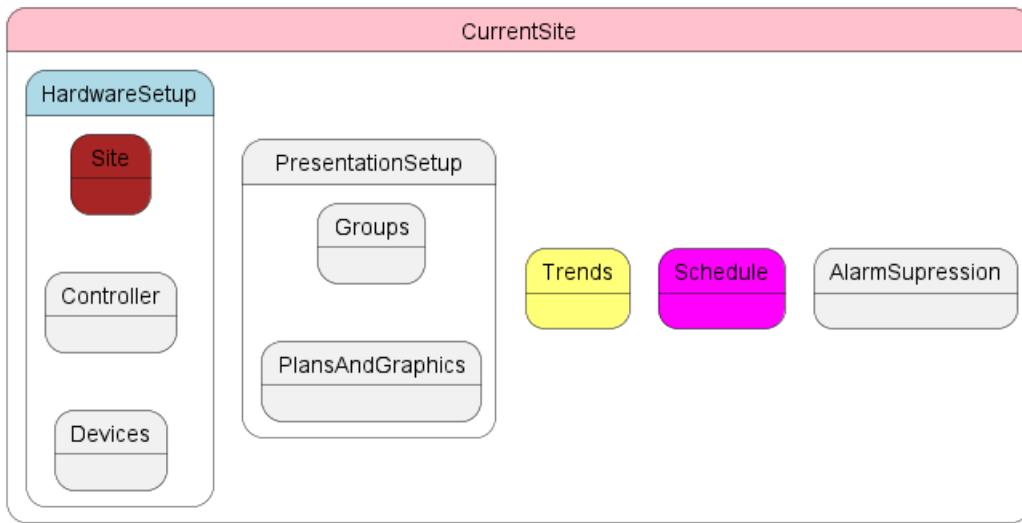
9.18 Color entre-línea

```

@startuml
state CurrentSite #pink {
    state HardwareSetup #lightblue {
        state Site #brown
        Site -[hidden]-> Controller
        Controller -[hidden]-> Devices
    }
    state PresentationSetup{
        Groups -[hidden]-> PlansAndGraphics
    }
    state Trends #FFFF77
    state Schedule #magenta
    state AlarmSupression
}
@enduml

```





[Ref. QA-1812]

9.19 Personalización (Skinparam)

Puedes usar el comando `skinparam` para cambiar los colores y las fuentes de los dibujos

Puedes usar este comando:

- En la definición del diagrama, como cualquier otro comando.
- En un archivo incluido.
- En un archivo de configuración, proporcionado en la consola de comandos o en el Ant task.

Puedes definir colores y fuentes específicas para estados estereotipados.

```

@startuml
skinparam backgroundColor LightYellow
skinparam state {
    StartColor MediumBlue
    EndColor Red
    BackgroundColor Peru
    BackgroundColor<<Warning>> Olive
    BorderColor Gray
    FontName Impact
}

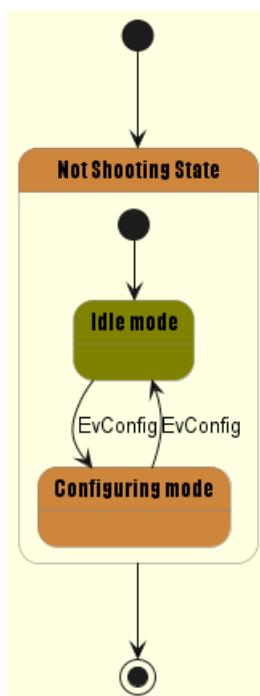
[*] --> NotShooting

state "Not Shooting State" as NotShooting {
    state "Idle mode" as Idle <<Warning>>
    state "Configuring mode" as Configuring
    [*] --> Idle
    Idle --> Configuring : EvConfig
    Configuring --> Idle : EvConfig
}

NotShooting --> [*]
@enduml

```





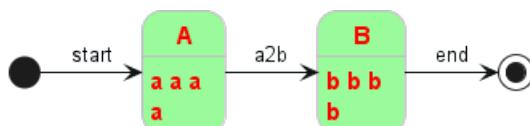
9.19.1 Prueba de todos los skinparam específicos de los diagramas de estado

```

@startuml
skinparam State {
    AttributeFontColor blue
    AttributeFontName serif
    AttributeFontSize 9
    AttributeFontStyle italic
    BackgroundColor palegreen
    BorderColor violet
    EndColor gold
    FontColor red
    FontName Sanserif
    FontSize 15
    FontStyle bold
    StartColor silver
}

state A : a a a\na
state B : b b b\nb

[*] -> A : start
A -> B : a2b
B -> [*] : end
@enduml
  
```



9.20 Cambiando estilo

Puedes cambiar el estilo.

```
@startuml
```



```

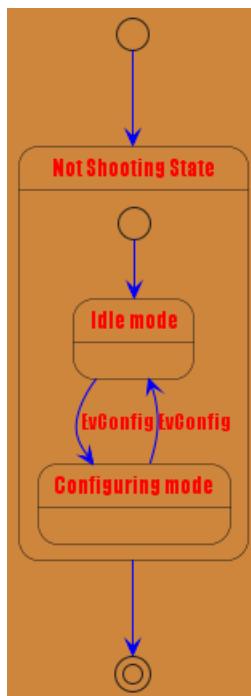
<style>
stateDiagram {
    backgroundColor Peru
    'LineColor Gray
    FontName Impact
    FontColor Red
    arrow {
        FontSize 13
        LineColor Blue
    }
}
</style>

[*] --> NotShooting

state "Not Shooting State" as NotShooting {
    state "Idle mode" as Idle <<Warning>>
    state "Configuring mode" as Configuring
    [*] --> Idle
    Idle --> Configuring : EvConfig
    Configuring --> Idle : EvConfig
}

NotShooting --> [*]
@enduml

```



[Ref. GH-880]

9.21 Change state color and style (inline style)

You can change the color or style of individual state using the following notation:

- `#color ##[style]color`

With background color first (`#color`), then line style and line color (`##[style]color`).



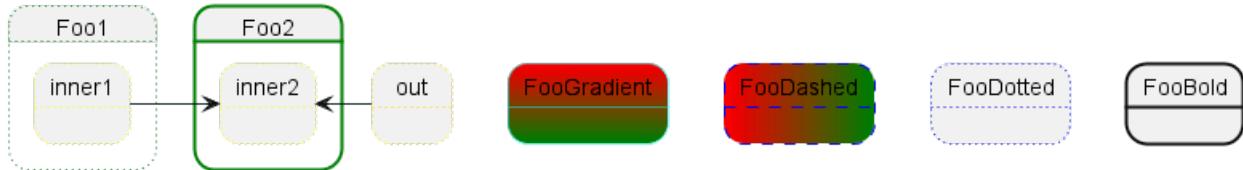
```

@startuml
state FooGradient #red-green ##00FFFF
state FooDashed #red|green ##[dashed]blue {
}
state FooDotted ##[dotted]blue {
}
state FooBold ##[bold] {
}
state Foo1 ##[dotted]green {
state inner1 ##[dotted]yellow
}

state out ##[dotted]gold

state Foo2 ##[bold]green {
state inner2 ##[dotted]yellow
}
inner1 -> inner2
out -> inner2
@enduml

```



[Ref. QA-1487]

- `#color;line:color;line.[bold|dashed|dotted];text:color`

TODO: FIXME `text:color` seems not to be taken into account **TODO:** FIXME

```

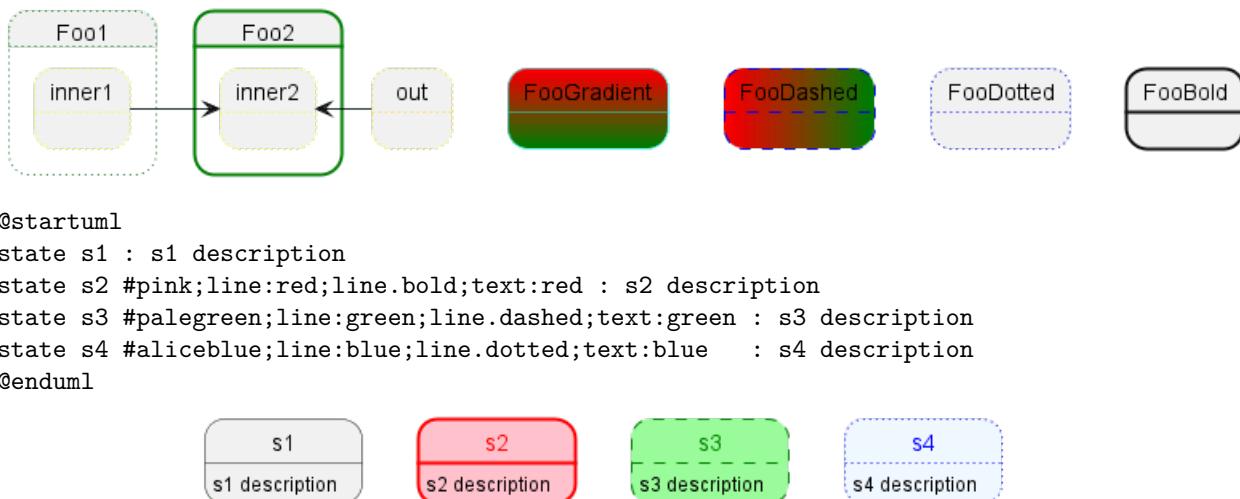
@startuml
@startuml
state FooGradient #red-green;line:00FFFF
state FooDashed #red|green;line.dashed;line:blue {
}
state FooDotted #line.dotted;line:blue {
}
state FooBold #line.bold {
}
state Foo1 #line.dotted;line:green {
state inner1 #line.dotted;line:yellow
}

state out #line.dotted;line:gold

state Foo2 #line.bold;line:green {
state inner2 #line.dotted;line:yellow
}
inner1 -> inner2
out -> inner2
@enduml
@enduml

```





9.22 Alias

With State you can use alias, like:

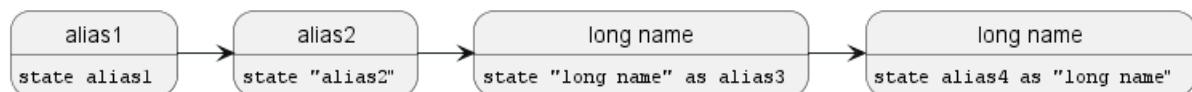
```

@startuml
state alias1
state "alias2"
state "long name" as alias3
state alias4 as "long name"

alias1 : ""state alias1"""
alias2 : ""state "alias2"""
alias3 : ""state "long name" as alias3"""
alias4 : ""state alias4 as "long name"""

alias1 -> alias2
alias2 -> alias3
alias3 -> alias4
@enduml

```



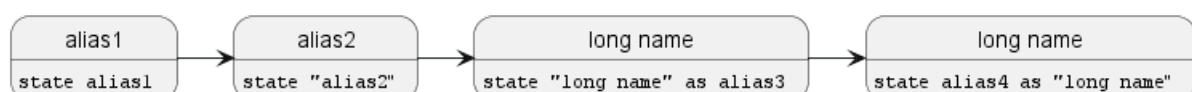
or:

```

@startuml
state alias1 : ""state alias1"""
state "alias2" : ""state "alias2"""
state "long name" as alias3 : ""state "long name" as alias3"""
state alias4 as "long name" : ""state alias4 as "long name"""

alias1 -> alias2
alias2 -> alias3
alias3 -> alias4
@enduml

```

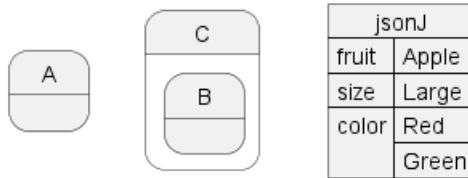


9.23 Display JSON Data on State diagram

9.23.1 Simple example

```
@startuml
state "A" as stateA
state "C" as stateC {
    state B
}

json jsonJ {
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}
@enduml
```



[Ref. QA-17275]

For another example, see on JSON page.



10 Timing Diagram

This is still under construction. You can propose new features if you need some.

10.1 Declaring element or participant

You declare participant using the following keywords, depending on how you want them to be drawn.

Keyword	Description
analog	An analog signal is continuous, and the values are linearly interpolated between the given setpoints
binary	A binary signal restricted to only 2 states
clock	A clocked signal that repeatedly transitions from high to low, with a period, and an optional pulse and offset
concise	A simplified concise signal designed to show the movement of data (great for messages)
robust	A robust complex line signal designed to show the transition from one state to another (can have many segments)

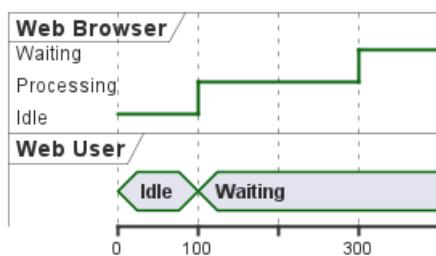
You define state change using the @ notation, and the is verb.

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU
```

```
@0
WU is Idle
WB is Idle
```

```
@100
WU is Waiting
WB is Processing
```

```
@300
WB is Waiting
@enduml
```



```
@startuml
clock "Clock_0" as C0 with period 50
clock "Clock_1" as C1 with period 50 pulse 15 offset 10
binary "Binary" as B
concise "Concise" as C
robust "Robust" as R
analog "Analog" as A
```

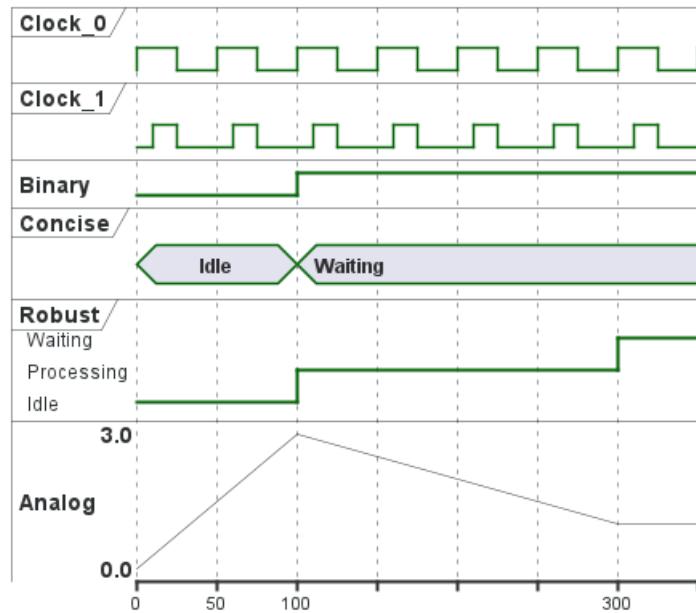
```
@0
C is Idle
R is Idle
A is 0
```

```
@100
B is high
C is Waiting
R is Processing
```



A is 3

```
@300
R is Waiting
A is 1
@enduml
```



[Ref. QA-14631, QA-14647 and QA-11288]

10.2 Binary and Clock

It's also possible to have binary and clock signal, using the following keywords:

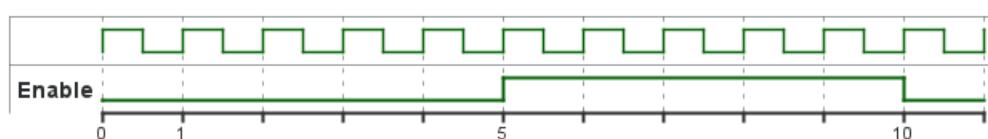
- `binary`
- `clock`

```
@startuml
clock clk with period 1
binary "Enable" as EN
```

```
@0
EN is low
```

```
@5
EN is high
```

```
@10
EN is low
@enduml
```



10.3 Adding message

You can add message using the following syntax.

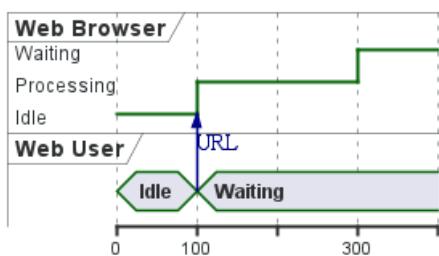


```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU
```

```
@0
WU is Idle
WB is Idle
```

```
@100
WU -> WB : URL
WU is Waiting
WB is Processing
```

```
@300
WB is Waiting
@enduml
```



10.4 Relative time

It is possible to use relative time with @.

```
@startuml
robust "DNS Resolver" as DNS
robust "Web Browser" as WB
concise "Web User" as WU
```

```
@0
WU is Idle
WB is Idle
DNS is Idle
```

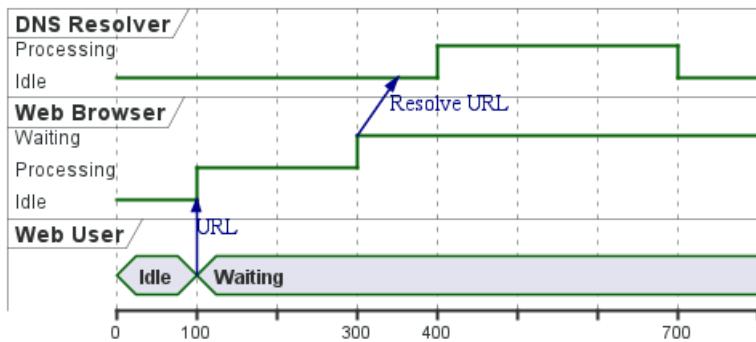
```
@+100
WU -> WB : URL
WU is Waiting
WB is Processing
```

```
@+200
WB is Waiting
WB -> DNS@+50 : Resolve URL
```

```
@+100
DNS is Processing
```

```
@+300
DNS is Idle
@enduml
```





10.5 Anchor Points

Instead of using absolute or relative time on an absolute time you can define a time as an anchor point by using the `as` keyword and starting the name with a `:`.

```
@XX as :<anchor point name>

@startuml
clock clk with period 1
binary "enable" as EN
concise "dataBus" as db

@0 as :start
@5 as :en_high
@10 as :en_low
@:en_high-2 as :en_highMinus2

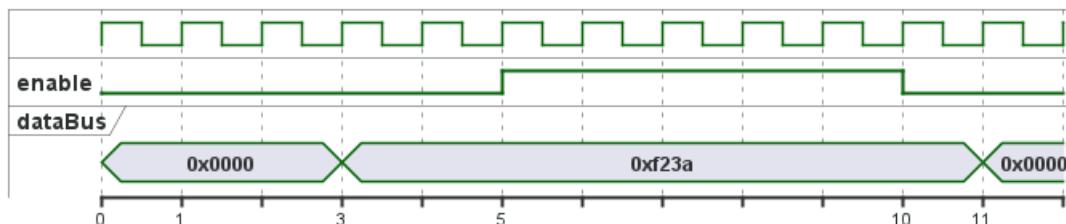
@:start
EN is low
db is "0x0000"

@:en_high
EN is high

@:en_low
EN is low

@:en_highMinus2
db is "0xf23a"

@:en_high+6
db is "0x0000"
@enduml
```



10.6 Participant oriented

Rather than declare the diagram in chronological order, you can define it by participant.

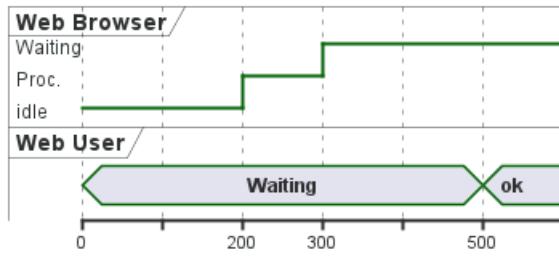
```
@startuml
```



```
robust "Web Browser" as WB
concise "Web User" as WU
```

```
@WB
0 is idle
+200 is Proc.
+100 is Waiting
```

```
@WU
0 is Waiting
+500 is ok
@enduml
```

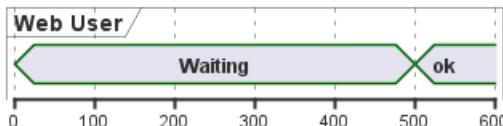


10.7 Setting scale

You can also set a specific scale.

```
@startuml
concise "Web User" as WU
scale 100 as 50 pixels
```

```
@WU
0 is Waiting
+500 is ok
@enduml
```



When using absolute Times/Dates, 1 "tick" is equivalent to 1 second.

```
@startuml
concise "Season" as S
'30 days is scaled to 50 pixels
scale 2592000 as 50 pixels
```

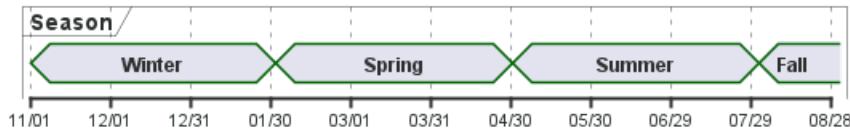
```
@2000/11/01
S is "Winter"
```

```
@2001/02/01
S is "Spring"
```

```
@2001/05/01
S is "Summer"
```

```
@2001/08/01
S is "Fall"
@enduml
```





10.8 Initial state

You can also define an initial state.

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU
```

WB is Initializing

WU is Absent

```
@WB
0 is idle
+200 is Processing
+100 is Waiting
```

```
@WU
0 is Waiting
+500 is ok
@enduml
```



10.9 Intricated state

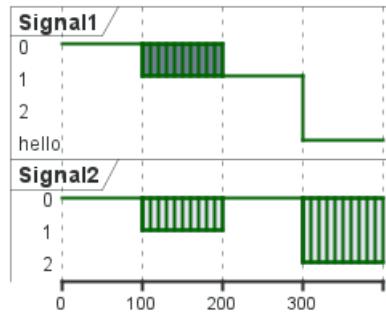
A signal could be in some undefined state.

10.9.1 Intricated or undefined robust state

```
@startuml
robust "Signal1" as S1
robust "Signal2" as S2
S1 has 0,1,2,hello
S2 has 0,1,2
@0
S1 is 0
S2 is 0
@100
S1 is {0,1} #SlateGrey
S2 is {0,1}
@200
S1 is 1
S2 is 0
@300
S1 is hello
```



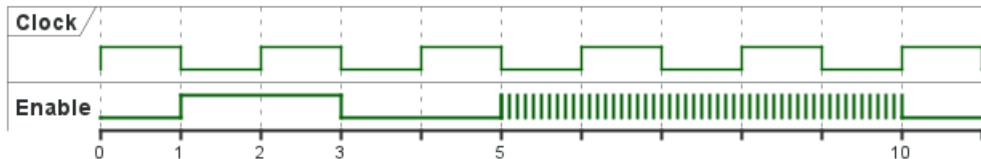
```
S2 is {0,2}
@enduml
```



10.9.2 Intricated or undefined binary state

```
@startuml
clock "Clock" as C with period 2
binary "Enable" as EN
```

```
@0
EN is low
@1
EN is high
@3
EN is low
@5
EN is {low,high}
@10
EN is low
@enduml
```



[Ref. QA-11936 and QA-15933]

10.10 Hidden state

It is also possible to hide some state.

```
@startuml
concise "Web User" as WU

@0
WU is {-}

@100
WU is A1

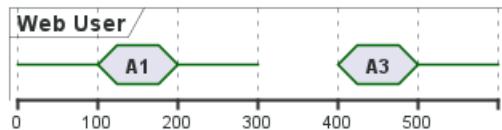
@200
WU is {-}

@300
WU is {hidden}
```



@400
WU is A3

@500
WU is {-}
@enduml



@startuml
scale 1 as 50 pixels

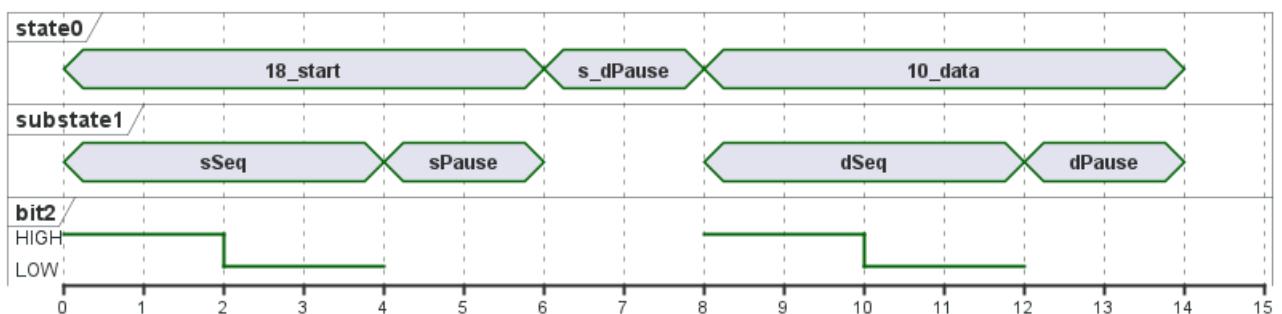
concise state0
concise substate1
robust bit2

bit2 has HIGH,LOW

@state0
0 is 18_start
6 is s_dPause
8 is 10_data
14 is {hidden}

@substate1
0 is sSeq
4 is sPause
6 is {hidden}
8 is dSeq
12 is dPause
14 is {hidden}

@bit2
0 is HIGH
2 is LOW
4 is {hidden}
8 is HIGH
10 is LOW
12 is {hidden}
@enduml



[Ref. QA-12222]



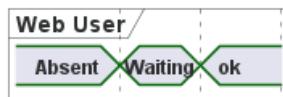
10.11 Hide time axis

It is possible to hide time axis.

```
@startuml
hide time-axis
concise "Web User" as WU
```

WU is Absent

```
@WU
0 is Waiting
+500 is ok
@enduml
```



10.12 Using Time and Date

It is possible to use time or date.

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU
```

@2019/07/02

WU is Idle

WB is Idle

@2019/07/04

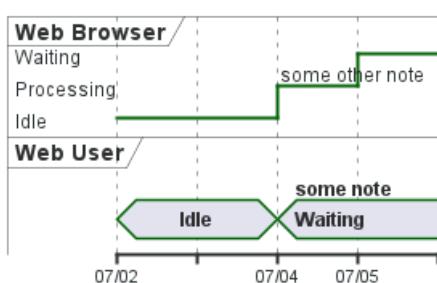
WU is Waiting : some note

WB is Processing : some other note

@2019/07/05

WB is Waiting

@enduml



@startuml

```
robust "Web Browser" as WB
concise "Web User" as WU
```

@1:15:00

WU is Idle

WB is Idle

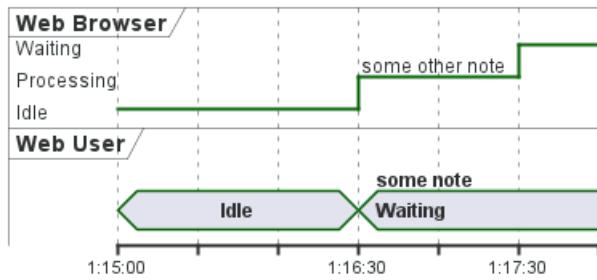
@1:16:30

WU is Waiting : some note

WB is Processing : some other note



```
@1:17:30
WB is Waiting
@enduml
```



10.13 Adding constraint

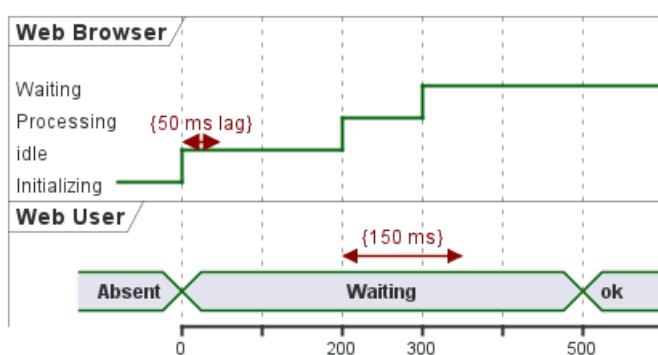
It is possible to display time constraints on the diagrams.

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

WB is Initializing
WU is Absent
```

```
@WB
0 is idle
+200 is Processing
+100 is Waiting
WB00 <-> @50 : {50 ms lag}
```

```
@WU
0 is Waiting
+500 is ok
@200 <-> @+150 : {150 ms}
@enduml
```



10.14 Highlighted period

You can highlight a part of diagram.

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

@0
```



WU is Idle
WB is Idle

@100
WU -> WB : URL
WU is Waiting #LightCyan;line:Aqua

WB is Proc.

@300
WU -> WB@350 : URL2
WB is Waiting

@+200
WU is ok

@+200
WB is Idle

```
highlight 200 to 450 #Gold;line:DimGrey : This is my caption  
highlight 600 to 700 : This is another\nhighlight  
@enduml
```



[Ref. QA-10868]

10.15 Using notes

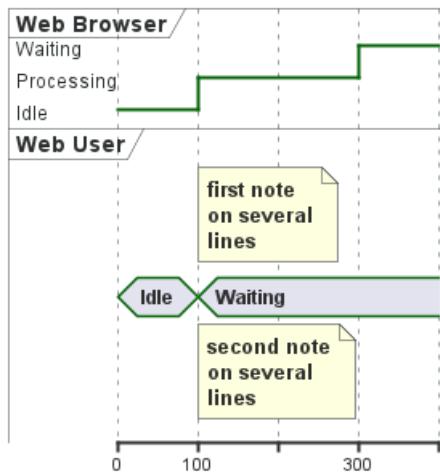
You can use the `note top` of and `note bottom` of keywords to define notes related to a single object or participant (*available only for concise or binary object*).

```
@startuml  
robust "Web Browser" as WB  
concise "Web User" as WU
```

@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing
note top of WU : first note\nnon several\nlines
note bottom of WU : second note\nnon several\nlines

@300
WB is Waiting
@enduml



[Ref. QA-6877, GH-1465]

10.16 Adding texts

You can optionally add a title, a header, a footer, a legend and a caption:

```
@startuml
Title This is my title
header: some header
footer: some footer
legend
Some legend
end legend
caption some caption

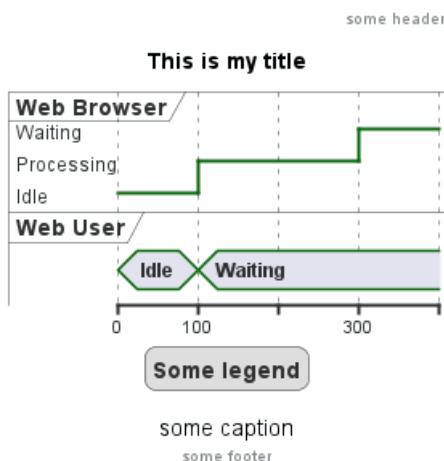
robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting
@enduml
```





10.17 Complete example

Thanks to Adam Rosien for this example.

```
@startuml
concise "Client" as Client
concise "Server" as Server
concise "Response freshness" as Cache

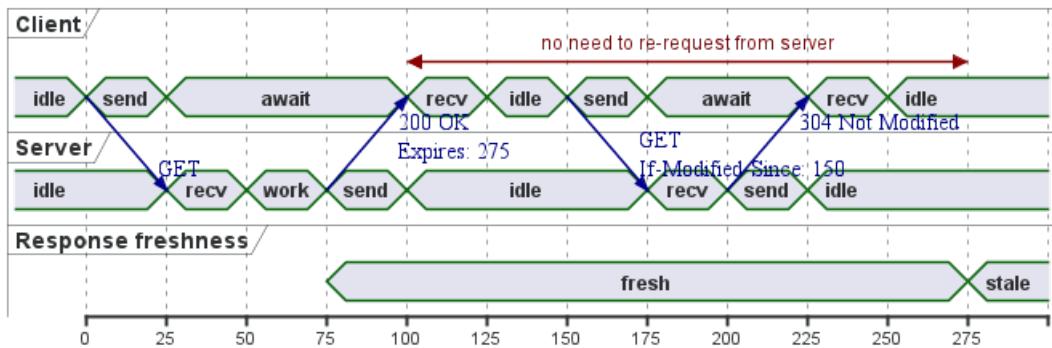
Server is idle
Client is idle

@Client
0 is send
Client -> Server@+25 : GET
+25 is await
+75 is recv
+25 is idle
+25 is send
Client -> Server@+25 : GET\nIf-Modified-Since: 150
+25 is await
+50 is recv
+25 is idle
@100 <-> @275 : no need to re-request from server

@Server
25 is recv
+25 is work
+25 is send
Server -> Client@+25 : 200 OK\nExpires: 275
+25 is idle
+75 is recv
+25 is send
Server -> Client@+25 : 304 Not Modified
+25 is idle

@Cache
75 is fresh
+200 is stale
@enduml
```





10.18 Digital Example

```
@startuml
scale 5 as 150 pixels

clock clk with period 1
binary "enable" as en
binary "R/W" as rw
binary "data Valid" as dv
concise "dataBus" as db
concise "address bus" as addr
```

```
@6 as :write_beg
@10 as :write_end
```

```
@15 as :read_beg
@19 as :read_end
```

```
@0
en is low
db is "0x0"
addr is "0x03f"
rw is low
dv is 0
```

```
@:write_beg-3
en is high
@:write_beg-2
db is "0xDEADBEEF"
@:write_beg-1
dv is 1
@:write_beg
rw is high
```

```
@:write_end
rw is low
dv is low
@:write_end+1
rw is low
db is "0x0"
addr is "0x23"
```

```
@12
dv is high
```



```

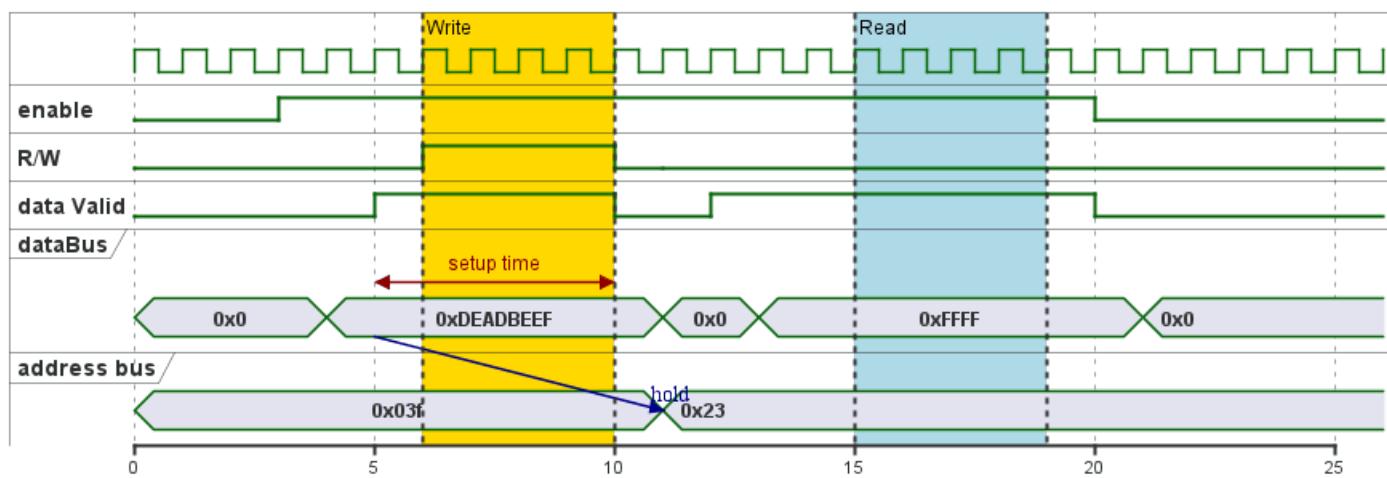
@13
db is "0xFFFF"

@20
en is low
dv is low
@21
db is "0x0"

highlight :write_beg to :write_end #Gold:Write
highlight :read_beg to :read_end #lightBlue:Read

db@:write_beg-1 <-> @:write_end : setup time
db@:write_beg-1 -> addr@:write_end+1 : hold
@enduml

```



10.19 Adding color

You can add color.

```

@startuml
concise "LR" as LR
concise "ST" as ST

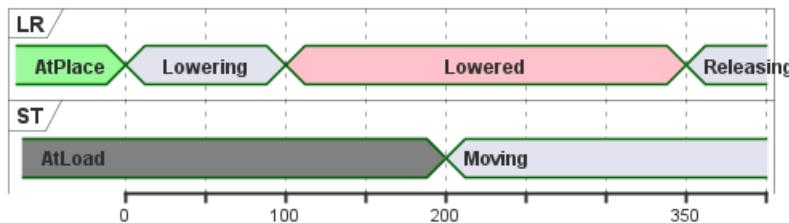
LR is AtPlace #palegreen
ST is AtLoad #gray

@LR
0 is Lowering
100 is Lowered #pink
350 is Releasing

@ST
200 is Moving
@enduml

```





[Ref. QA-5776]

10.20 Using (global) style

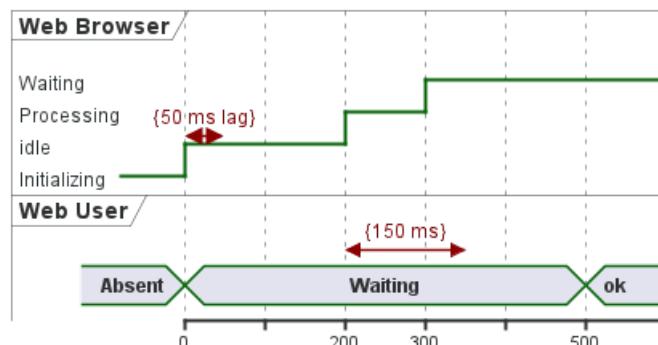
10.20.1 Without style (by default)

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU
```

WB is Initializing
WU is Absent

```
@WB
0 is idle
+200 is Processing
+100 is Waiting
WB@0 <-> @50 : {50 ms lag}
```

```
@WU
0 is Waiting
+500 is ok
@200 <-> @+150 : {150 ms}
@enduml
```



10.20.2 With style

You can use style to change rendering of elements.

```
@startuml
<style>
timingDiagram {
    document {
        BackGroundColor SandyBrown
    }
    constraintArrow {
        LineStyle 2-1
        LineThickness 3
        LineColor Blue
    }
}
```



```

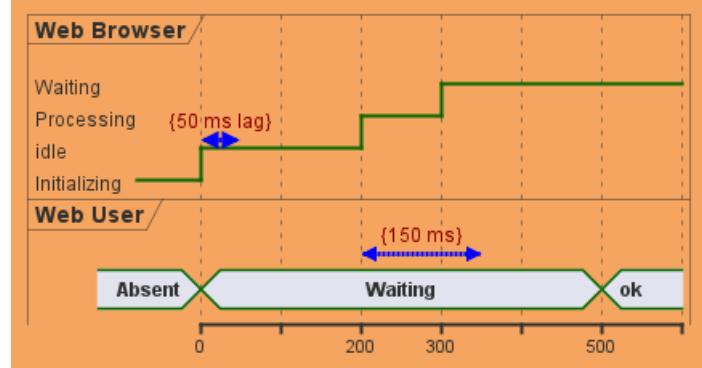
}
}
</style>
robust "Web Browser" as WB
concise "Web User" as WU

WB is Initializing
WU is Absent

@WB
0 is idle
+200 is Processing
+100 is Waiting
WB00 <-> @50 : {50 ms lag}

@WU
0 is Waiting
+500 is ok
@200 <-> @+150 : {150 ms}
@enduml

```



[Ref. QA-14340]

10.21 Applying Colors to specific lines

You can use the `<style>` tags and stereotyping to give a name to line attributes.

```

@startuml
<style>
timingDiagram {
    .red {
        LineColor red
    }
    .blue {
        LineColor blue
        LineThickness 5
    }
}
</style>

clock clk with period 1
binary "Input Signal 1" as IS1
binary "Input Signal 2" as IS2 <<blue>>
binary "Output Signal 1" as OS1 <<red>>

@0
IS1 is low

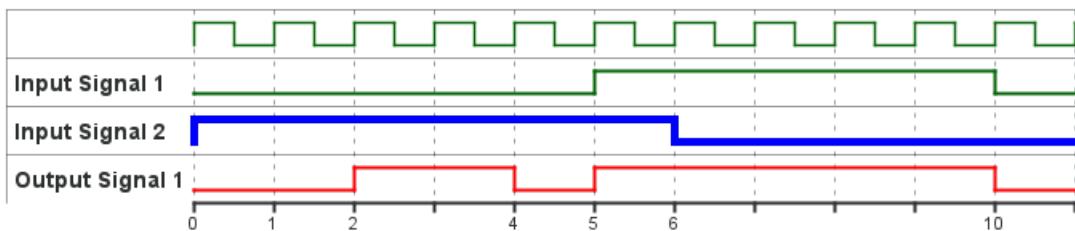
```



```

IS2 is high
OS1 is low
@2
OS1 is high
@4
OS1 is low
@5
IS1 is high
OS1 is high
@6
IS2 is low
@10
IS1 is low
OS1 is low
@enduml

```



[Ref. QA-15870]

10.22 Compact mode

You can use `compact` command to compact the timing layout.

10.22.1 By default

```

@startuml
robust "Web Browser" as WB
concise "Web User" as WU
robust "Web Browser2" as WB2

@0
WU is Waiting
WB is Idle
WB2 is Idle

@200
WB is Proc.

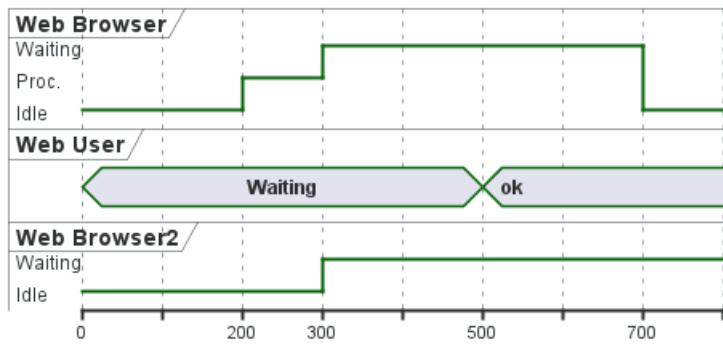
@300
WB is Waiting
WB2 is Waiting

@500
WU is ok

@700
WB is Idle
@enduml

```





10.22.2 Global mode with mode compact

```
@startuml
mode compact
robust "Web Browser" as WB
concise "Web User" as WU
robust "Web Browser2" as WB2
```

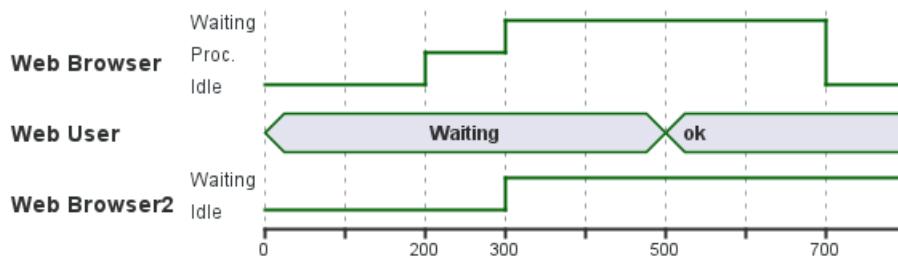
```
@0
WU is Waiting
WB is Idle
WB2 is Idle
```

```
@200
WB is Proc.
```

```
@300
WB is Waiting
WB2 is Waiting
```

```
@500
WU is ok
```

```
@700
WB is Idle
@enduml
```



10.22.3 Local mode with only compact on element

```
@startuml
compact robust "Web Browser" as WB
compact concise "Web User" as WU
robust "Web Browser2" as WB2
```

```
@0
WU is Waiting
WB is Idle
```



WB2 is Idle

@200

WB is Proc.

@300

WB is Waiting

WB2 is Waiting

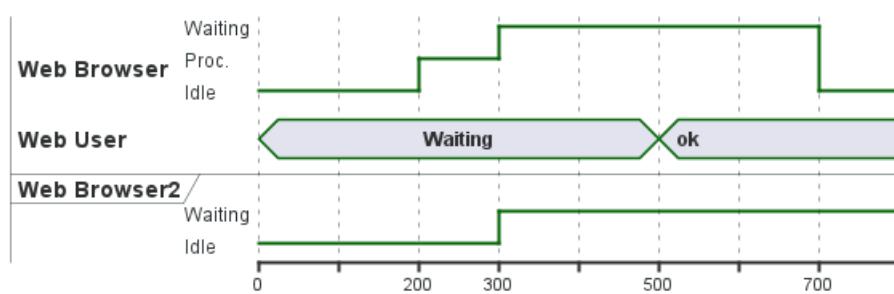
@500

WU is ok

@700

WB is Idle

@enduml



[Ref. QA-11130]



11 Display JSON Data

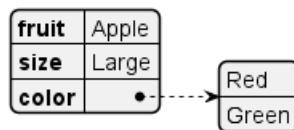
JSON format is widely used in software.

You can use PlantUML to visualize your data.

To activate this feature, the diagram must:

- begin with `@startjson` keyword
- end with `@endjson` keyword.

```
@startjson
{
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}
@endjson
```

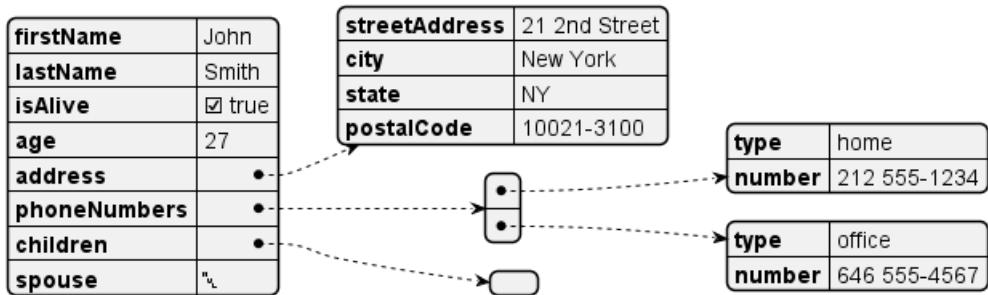


11.1 Complex example

You can use complex JSON structure.

```
@startjson
{
    "firstName": "John",
    "lastName": "Smith",
    "isAlive": true,
    "age": 27,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021-3100"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "office",
            "number": "646 555-4567"
        }
    ],
    "children": [],
    "spouse": null
}
@endjson
```



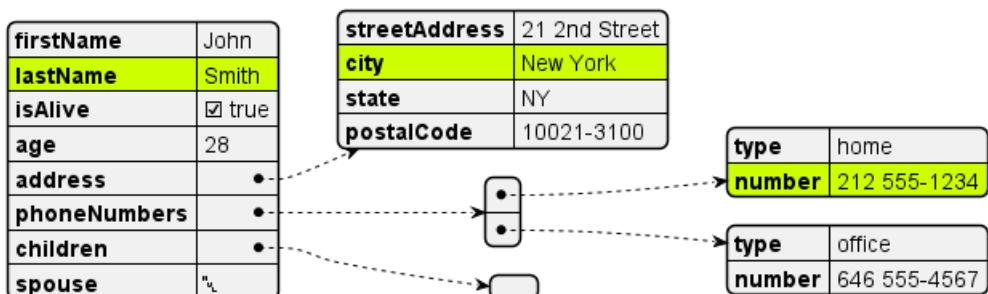


11.2 Highlight parts

```

@startjson
#highlight "lastName"
#highlight "address" / "city"
#highlight "phoneNumbers" / "0" / "number"
{
    "firstName": "John",
    "lastName": "Smith",
    "isAlive": true,
    "age": 28,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021-3100"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "office",
            "number": "646 555-4567"
        }
    ],
    "children": [],
    "spouse": null
}
@endjson

```



11.3 Using different styles for highlight

It is possible to have different styles for different highlights.

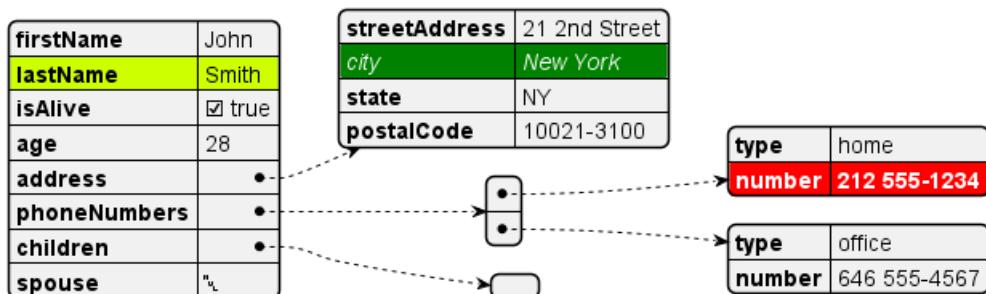
```
@startjson
```



```

<style>
.h1 {
    BackGroundColor green
    FontColor white
    FontStyle italic
}
.h2 {
    BackGroundColor red
    FontColor white
    FontStyle bold
}
</style>
#highlight "lastName"
#highlight "address" / "city" <<h1>>
#highlight "phoneNumbers" / "0" / "number" <<h2>>
{
    "firstName": "John",
    "lastName": "Smith",
    "isAlive": true,
    "age": 28,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021-3100"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "office",
            "number": "646 555-4567"
        }
    ],
    "children": [],
    "spouse": null
}
@endjson

```



[Ref. QA-15756, GH-1393]

11.4 JSON basic element

11.4.1 Synthesis of all JSON basic element

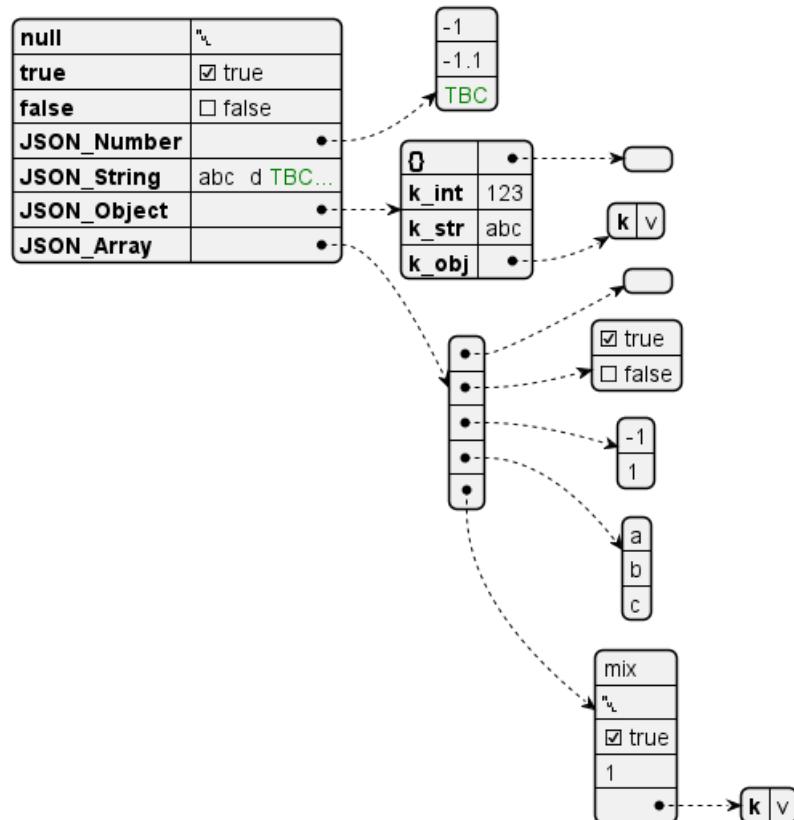
```
@startjson
{
```



```

>null": null,
>true": true,
>false": false,
"JSON_Number": [-1, -1.1, "<color:green>TBC"],
"JSON_String": "a\nb\rc\td <color:green>TBC...",
"JSON_Object": {
    "{}": {},
    "k_int": 123,
    "k_str": "abc",
    "k_obj": {"k": "v"}
},
"JSON_Array" : [
    [],
    [true, false],
    [-1, 1],
    ["a", "b", "c"],
    ["mix", null, true, 1, {"k": "v"}]
]
}
@endjson

```



11.5 JSON array or table

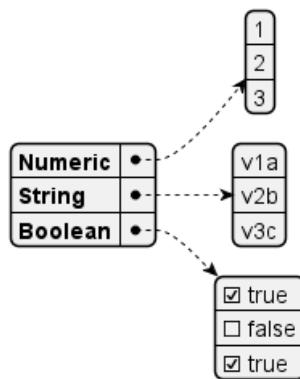
11.5.1 Array type

```

@startjson
{
"Numeric": [1, 2, 3],
"String": ["v1a", "v2b", "v3c"],
"Boolean": [true, false, true]
}
@endjson

```





11.5.2 Minimal array or table

11.5.3 Number array

```
@startjson
[1, 2, 3]
@endjson
```



11.5.4 String array

```
@startjson
["1a", "2b", "3c"]
@endjson
```



11.5.5 Boolean array

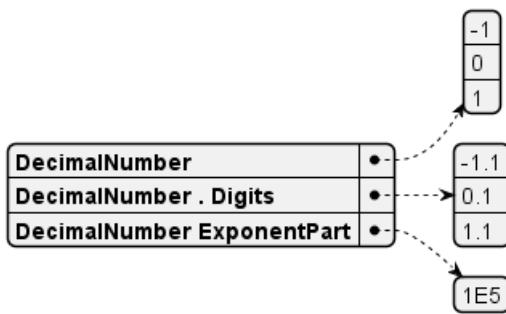
```
@startjson
[true, false, true]
@endjson
```



11.6 JSON numbers

```
@startjson
{
    "DecimalNumber": [-1, 0, 1],
    "DecimalNumber . Digits": [-1.1, 0.1, 1.1],
    "DecimalNumber ExponentPart": [1E5]
}
@endjson
```





11.7 JSON strings

11.7.1 JSON Unicode

On JSON you can use Unicode directly or by using escaped form like .

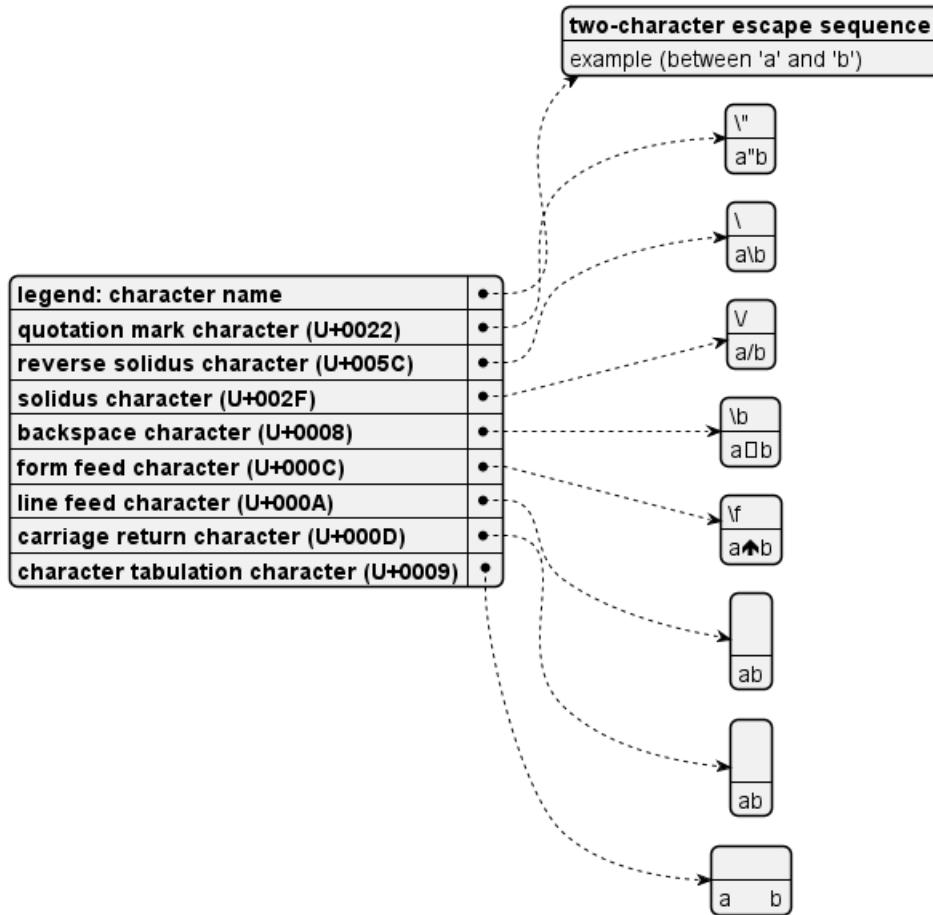
```
@startjson
{
    "<color:blue><b>code": "<color:blue><b>value",
    "a\\u005Cb": "a\u005Cb",
    "\\uD83D\\uDE10": "\uD83D\uDE10",
    " ":
}
@endjson
```

code	value
a\u005Cb	a\b
\uD83D\uDE10	(?)
(?)	(?)

11.7.2 JSON two-character escape sequence

```
@startjson
{
    "**legend**: character name":
    "quotation mark character (U+0022)": ["\"\"", "a\"b"],
    "reverse solidus character (U+005C)": ["\\\\\\\", "a\\\"b"],
    "solidus character (U+002F)": ["\\\\\\/", "a\\/b"],
    "backspace character (U+0008)": ["\\b", "a\\bb"],
    "form feed character (U+000C)": ["\\f", "a\\fb"],
    "line feed character (U+000A)": ["\\n", "a\\nb"],
    "carriage return character (U+000D)": ["\\r", "a\\rb"],
    "character tabulation character (U+0009)": ["\\t", "a\\tb"]
}
@endjson
```





TODO: FIXME FIXME or not , on the same item as management in PlantUML See Report Bug on QA-13066 **TODO:** FIXME

```
@startjson
[
  "\\\\" ,
  "\\n" ,
  "\\r" ,
  "\\t"
]
@endjson
```



11.8 Minimal JSON examples

```
@startjson
"Hello world!"
@endjson
```

Hello world!



```
@startjson
42
@endjson
```

```
@startjson
true
@endjson
```

(Examples come from STD 90 - Examples)

11.9 Empty table or list

```
@startjson
{
  "empty_tab": [],
  "empty_list": {}
}
@endjson
```

empty_tab	•	→	○
empty_list	•	→	○

[Ref. QA-14397]

11.10 Using (global) style

11.10.1 Without style (*by default*)

```
@startjson
#highlight "1" / "hr"
[
  {
    "name": "Mark McGwire",
    "hr": 65,
    "avg": 0.278
  },
  {
    "name": "Sammy Sosa",
    "hr": 63,
    "avg": 0.288
  }
]
@endjson
```

•	→	•	
•	→	•	

name	Mark McGwire
hr	65
avg	0.278

name	Sammy Sosa
hr	63
avg	0.288

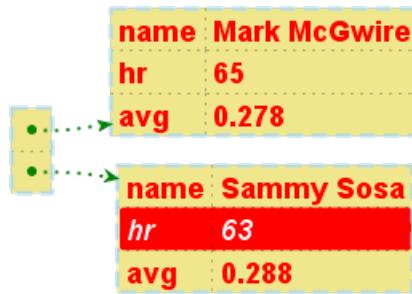


11.10.2 With style

You can use style to change rendering of elements.

```
@startjson
<style>
jsonDiagram {
    node {
        BackGroundColor Khaki
        LineColor lightblue
        FontName Helvetica
        FontColor red
        FontSize 18
        FontStyle bold
        RoundCorner 0
        LineThickness 2
        LineStyle 10-5
        separator {
            LineThickness 0.5
            LineColor black
            LineStyle 1-5
        }
    }
    arrow {
        BackGroundColor lightblue
        LineColor green
        LineThickness 2
        LineStyle 2-5
    }
    highlight {
        BackGroundColor red
        FontColor white
        FontStyle italic
    }
}
</style>
#highlight "1" / "hr"
[
{
    "name": "Mark McGwire",
    "hr": 65,
    "avg": 0.278
},
{
    "name": "Sammy Sosa",
    "hr": 63,
    "avg": 0.288
}
]
@endjson
```





[Adapted from QA-13123 and QA-13288]

11.11 Display JSON Data on Class or Object diagram

11.11.1 Simple example

```
@startuml
class Class
object Object
json JSON {
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}
@enduml
```



JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

11.11.2 Complex example: with all JSON basic element

```
@startuml
json "<b>JSON basic element</b>" as J {
    "null": null,
    "true": true,
    "false": false,
    "JSON_Number": [-1, -1.1, "<color:green>TBC"],
    "JSON_String": "a\nb\rc\td <color:green>TBC...",
    "JSON_Object": {
        "{}": {},
        "k_int": 123,
        "k_str": "abc",
        "k_obj": {"k": "v"}
    },
    "JSON_Array" : [
        [],
        [true, false],
        [-1, 1],
        ...
    ]
}
```



```

["a", "b", "c"],
["mix", null, true, 1, {"k": "v"}]
]
}
@enduml

```

JSON basic element	
null	null
true	true
false	false
JSON_Number	-1 -1.1 TBC
JSON_String	abc d TBC...
JSON_Object	{ } k_int 123 k_str abc k_obj k v
JSON_Array	true false -1 1 a b c mix null true 1 k v

11.12 Display JSON Data on Deployment (Usecase, Component, Deployment) diagram

11.12.1 Simple example

```

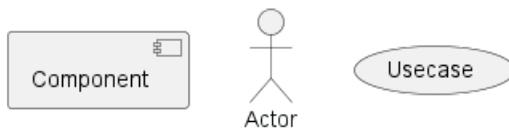
@startuml
allowmixing

component Component
actor Actor
usecase Usecase
() Interface
node Node
cloud Cloud

json JSON {
    "fruit":"Apple",
    "size":"Large",
    "color": ["Red", "Green"]
}
@enduml

```





JSON	
fruit	Apple
size	Large
color	Red
	Green

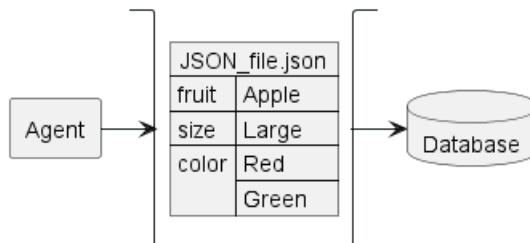
[Ref. QA-15481]

Complex example: with arrow

```
@startuml
allowmixing

agent Agent
stack {
    json "JSON_file.json" as J {
        "fruit":"Apple",
        "size":"Large",
        "color": ["Red", "Green"]
    }
}
database Database

Agent -> J
J -> Database
@enduml
```



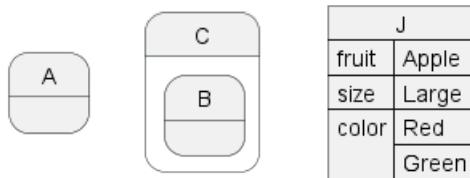
11.13 Display JSON Data on State diagram

11.13.1 Simple example

```
@startuml
state "A" as stateA
state "C" as stateC {
    state B
}
```



```
json J {  
    "fruit": "Apple",  
    "size": "Large",  
    "color": ["Red", "Green"]  
}  
@enduml
```



[Ref. QA-17275]



12 Display YAML Data

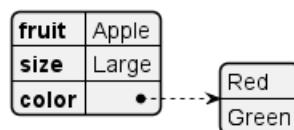
YAML format is widely used in software.

You can use PlantUML to visualize your data.

To activate this feature, the diagram must:

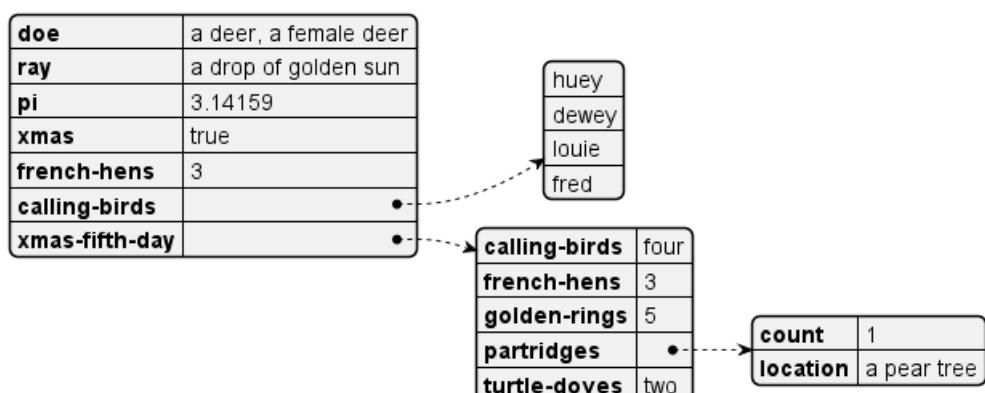
- begin with `@startyaml` keyword
- end with `@endyaml` keyword.

```
@startyaml
fruit: Apple
size: Large
color:
  - Red
  - Green
@endyaml
```



12.1 Complex example

```
@startyaml
doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
xmas-fifth-day:
  french-hens: 3
  calling-birds:
    - huey
    - dewey
    - louie
    - fred
  xmas-fifth-day:
    calling-birds: four
    french-hens: 3
    golden-rings: 5
    partridges:
      count: 1
      location: "a pear tree"
    turtle-doves: two
@endyaml
```



12.2 Specific key (with symbols or unicode)

```
@startyaml
$fruit: Apple
$size: Large
&color: Red
: Heart
%: Per mille
@endyaml
```

@fruit	Apple
\$size	Large
&color	Red
♥	Heart
%	Per mille

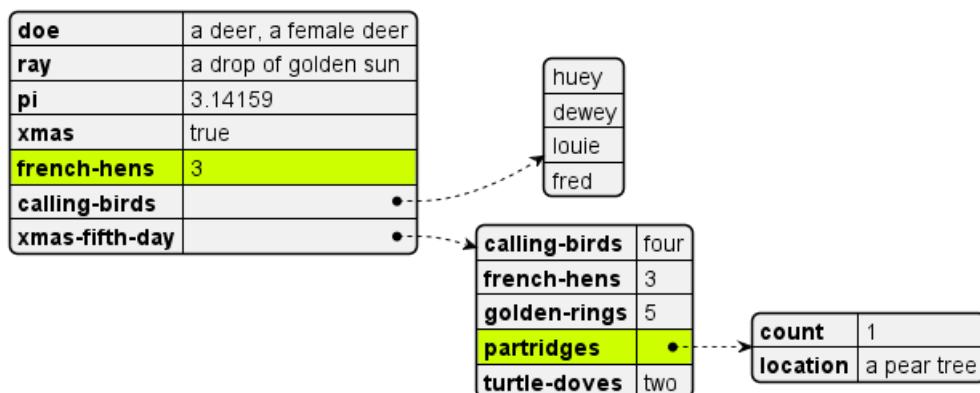
[Ref. QA-13376]

12.3 Highlight parts

12.3.1 Normal style

```
@startyaml
#highlight "french-hens"
#highlight "xmas-fifth-day" / "partridges"

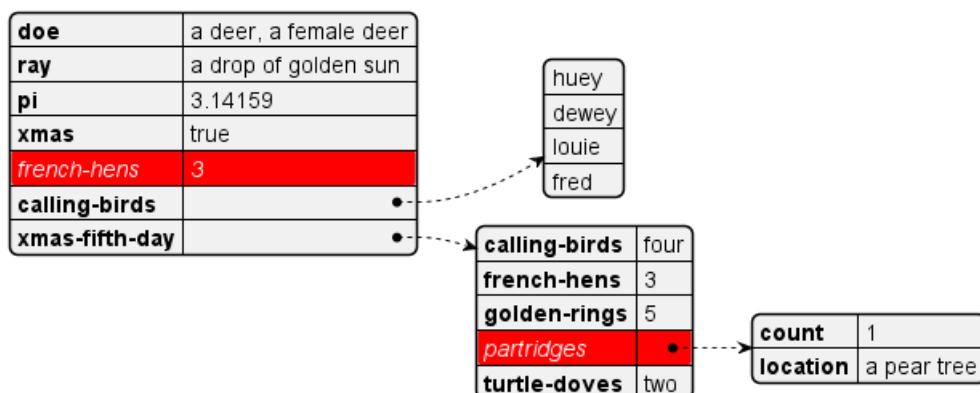
doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```



12.3.2 Customised style

```
@startyaml
<style>
yamlDiagram {
    highlight {
        BackGroundColor red
        FontColor white
        FontStyle italic
    }
}
</style>
#highlight "french-hens"
#highlight "xmas-fifth-day" / "partridges"

doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```



[Ref. QA-13288]

12.4 Using different styles for highlight

It is possible to have different styles for different highlights.

```
@startyaml
<style>
.h1 {
    BackGroundColor green
    FontColor white
}
```

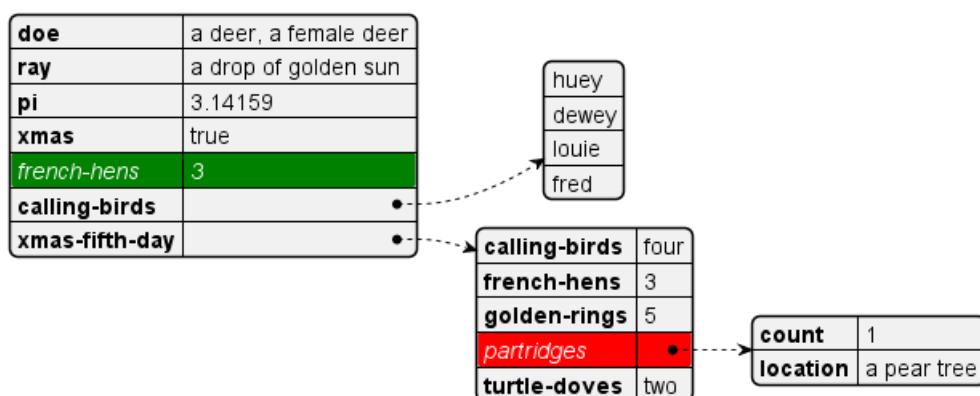


```

    FontStyle italic
}
.h2 {
    BackGroundColor red
    FontColor white
    FontStyle italic
}
</style>
#highlight "french-hens" <<h1>>
#highlight "xmas-fifth-day" / "partridges" <<h2>>

doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml

```



[Ref. QA-15756, GH-1393]

12.5 Using (global) style

12.5.1 Without style (*by default*)

@startyaml

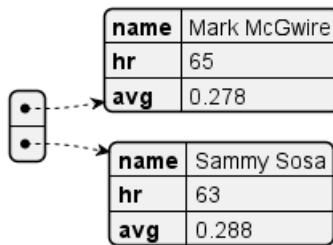
```

-
  name: Mark McGwire
  hr: 65
  avg: 0.278
-
  name: Sammy Sosa
  hr: 63

```



```
avg: 0.288
@endyaml
```

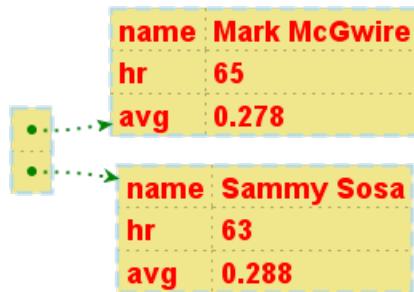


12.5.2 With style

You can use style to change rendering of elements.

```
@startyaml
<style>
yamlDiagram {
    node {
        BackGroundColor lightblue
        LineColor lightblue
        FontName Helvetica
        FontColor red
        FontSize 18
        FontStyle bold
        BackGroundColor Khaki
        RoundCorner 0
        LineThickness 2
        LineStyle 10-5
        separator {
            LineThickness 0.5
            LineColor black
            LineStyle 1-5
        }
    }
    arrow {
        BackGroundColor lightblue
        LineColor green
        LineThickness 2
        LineStyle 2-5
    }
}
</style>
-
  name: Mark McGwire
  hr: 65
  avg: 0.278
-
  name: Sammy Sosa
  hr: 63
  avg: 0.288
@endyaml
```





[Ref. QA-13123]



13 Network diagram (nwdiag)

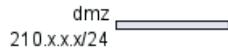
nwdiag has been created by Takeshi Komiya and allows to quickly draw network diagrams. So we thank him for his creation!

Since the syntax is clear and simple, this has been integrated within PlantUML. We reuse here the examples that Takeshi has documented.

13.1 Simple diagram

13.1.1 Define a network

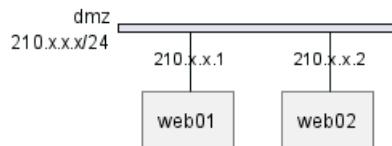
```
@startuml
nwdiag {
    network dmz {
        address = "210.x.x.x/24"
    }
}
@enduml
```



13.1.2 Define some elements or servers on a network

```
@startuml
nwdiag {
    network dmz {
        address = "210.x.x.x/24"

        web01 [address = "210.x.x.1"];
        web02 [address = "210.x.x.2"];
    }
}
@enduml
```



13.1.3 Full example

```
@startuml
nwdiag {
    network dmz {
        address = "210.x.x.x/24"

        web01 [address = "210.x.x.1"];
        web02 [address = "210.x.x.2"];
    }
    network internal {
        address = "172.x.x.x/24";

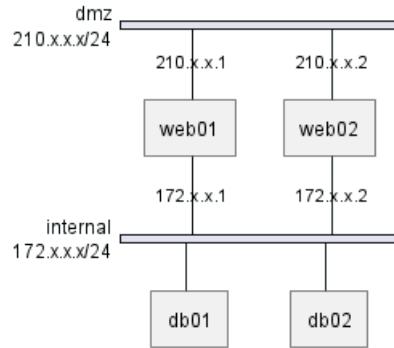
        web01 [address = "172.x.x.1"];
        web02 [address = "172.x.x.2"];
    }
}
```



```

db01;
db02;
}
}
@enduml

```



13.2 Define multiple addresses

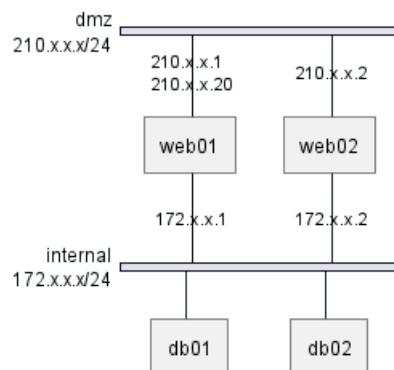
```

@startuml
nwdiag {
    network dmz {
        address = "210.x.x.x/24"

        // set multiple addresses (using comma)
        web01 [address = "210.x.x.1, 210.x.x.20"];
        web02 [address = "210.x.x.2"];
    }
    network internal {
        address = "172.x.x.x/24";

        web01 [address = "172.x.x.1"];
        web02 [address = "172.x.x.2"];
        db01;
        db02;
    }
}
@enduml

```



13.3 Grouping nodes

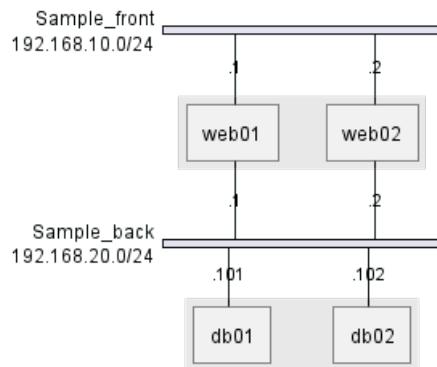
13.3.1 Define group inside network definitions

```
@startuml
nwdiag {
    network Sample_front {
        address = "192.168.10.0/24";

        // define group
        group web {
            web01 [address = ".1"];
            web02 [address = ".2"];
        }
    }

    network Sample_back {
        address = "192.168.20.0/24";
        web01 [address = ".1"];
        web02 [address = ".2"];
        db01 [address = ".101"];
        db02 [address = ".102"];

        // define network using defined nodes
        group db {
            db01;
            db02;
        }
    }
}
@enduml
```



13.3.2 Define group outside of network definitions

```
@startuml
nwdiag {
    // define group outside of network definitions
    group {
        color = "#FFAAAA";

        web01;
        web02;
        db01;
    }

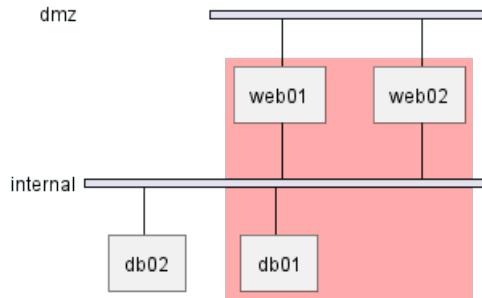
    network dmz {
```



```

web01;
web02;
}
network internal {
    web01;
    web02;
    db01;
    db02;
}
}
@enduml

```



13.3.3 Define several groups on same network

13.3.4 Example with 2 group

```

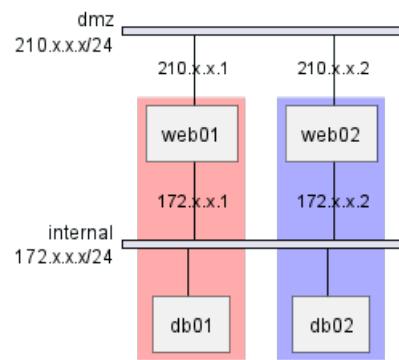
@startuml
nwdiag {
    group {
        color = "#FFaaaa";
        web01;
        db01;
    }
    group {
        color = "#aaaaFF";
        web02;
        db02;
    }
    network dmz {
        address = "210.x.x.x/24"

        web01 [address = "210.x.x.1"];
        web02 [address = "210.x.x.2"];
    }
    network internal {
        address = "172.x.x.x/24";

        web01 [address = "172.x.x.1"];
        web02 [address = "172.x.x.2"];
        db01 ;
        db02 ;
    }
}
@enduml

```





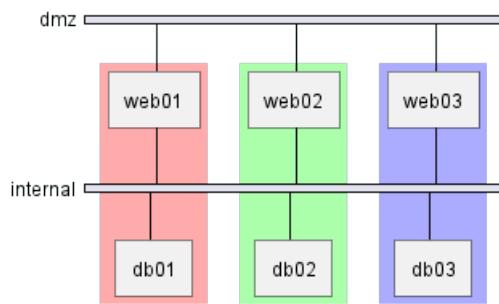
[Ref. QA-12663]

13.3.5 Example with 3 groups

```
@startuml
nwdiag {
    group {
        color = "#FFaaaa";
        web01;
        db01;
    }
    group {
        color = "#aaFFaa";
        web02;
        db02;
    }
    group {
        color = "#aaaaFF";
        web03;
        db03;
    }

    network dmz {
        web01;
        web02;
        web03;
    }
    network internal {
        web01;
        db01 ;
        web02;
        db02 ;
        web03;
        db03;
    }
}
@enduml
```





[Ref. QA-13138]

13.4 Extended Syntax (for network or group)

13.4.1 Network

For network or network's component, you can add or change:

- addresses (*separated by comma ,*);
- color;
- description;
- shape.

```

@startuml
nwdiag {
    network Sample_front {
        address = "192.168.10.0/24"
        color = "red"

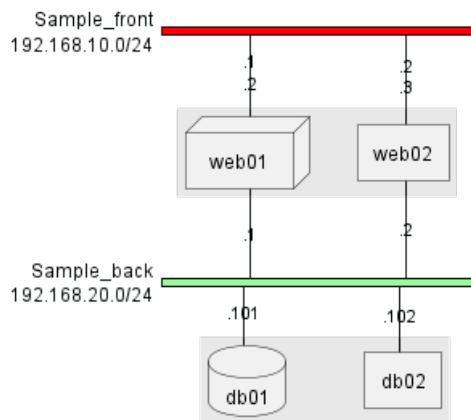
        // define group
        group web {
            web01 [address = ".1, .2", shape = "node"]
            web02 [address = ".2, .3"]
        }
    }

    network Sample_back {
        address = "192.168.20.0/24"
        color = "palegreen"
        web01 [address = ".1"]
        web02 [address = ".2"]
        db01 [address = ".101", shape = database ]
        db02 [address = ".102"]

        // define network using defined nodes
        group db {
            db01;
            db02;
        }
    }
}
@enduml

```





13.4.2 Group

For a group, you can add or change:

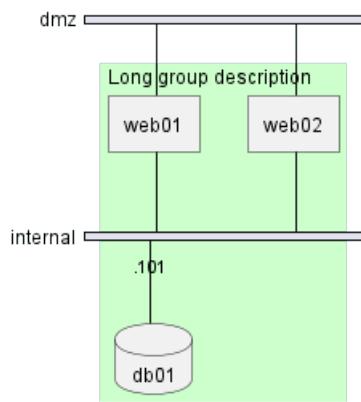
- color;
- description.

```
@startuml
nwdiag {
    group {
        color = "#CCFFCC";
        description = "Long group description";

        web01;
        web02;
        db01;
    }

    network dmz {
        web01;
        web02;
    }
    network internal {
        web01;
        web02;
        db01 [address = ".101", shape = database];
    }
}
@enduml
```





[Ref. QA-12056]

13.5 Using Sprites

You can use all sprites (icons) from the Standard Library or any other library.

Use the notation <\$sprite> to use a sprite, to make a new line, or any other Creole syntax.

```

@startuml
!include <office/Servers/application_server>
!include <office/Servers/database_server>

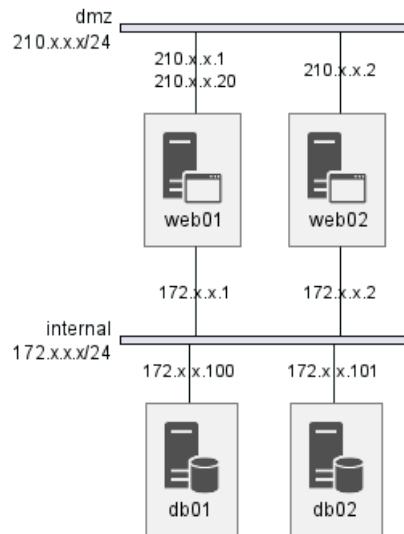
nwdiag {
    network dmz {
        address = "210.x.x.x/24"

        // set multiple addresses (using comma)
        web01 [address = "210.x.x.1, 210.x.x.20", description = "<$application_server>\n web01"]
        web02 [address = "210.x.x.2", description = "<$application_server>\n web02"];
    }
    network internal {
        address = "172.x.x.x/24";

        web01 [address = "172.x.x.1"];
        web02 [address = "172.x.x.2"];
        db01 [address = "172.x.x.100", description = "<$database_server>\n db01"];
        db02 [address = "172.x.x.101", description = "<$database_server>\n db02"];
    }
}
@enduml

```





[Ref. QA-11862]

13.6 Using OpenIconic

You can also use the icons from OpenIconic in network or node descriptions.

Use the notation `<&icon>` to make an icon, `<&icon*n>` to multiply the size by a factor n, and `\n` to make a newline:

```
@startuml

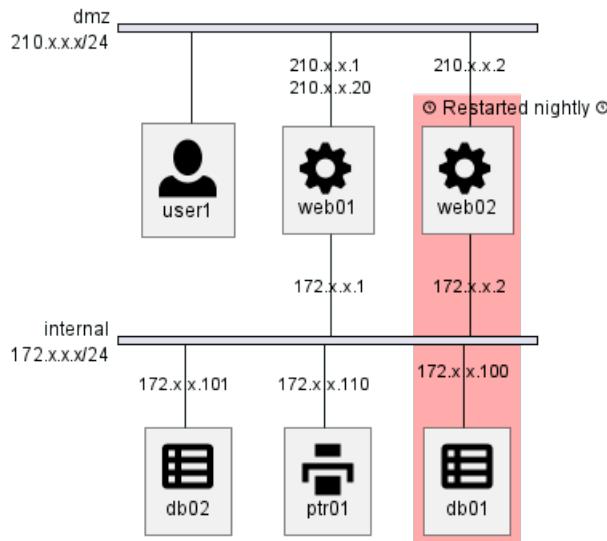
nwdiag {
    group nightly {
        color = "#FFAAAA";
        description = "<&clock> Restarted nightly <&clock>";
        web02;
        db01;
    }
    network dmz {
        address = "210.x.x.x/24"

        user [description = "<&person*4.5>\n user1"];
        // set multiple addresses (using comma)
        web01 [address = "210.x.x.1, 210.x.x.20", description = "<&cog*4>\nweb01"];
        web02 [address = "210.x.x.2", description = "<&cog*4>\nweb02"];

    }
    network internal {
        address = "172.x.x.x/24";

        web01 [address = "172.x.x.1"];
        web02 [address = "172.x.x.2"];
        db01 [address = "172.x.x.100", description = "<&spreadsheet*4>\n db01"];
        db02 [address = "172.x.x.101", description = "<&spreadsheet*4>\n db02"];
        ptr [address = "172.x.x.110", description = "<&print*4>\n ptr01"];
    }
}
@enduml
```





13.7 Same nodes on more than two networks

You can use same nodes on different networks (more than two networks); *nwdiag* use in this case '*jump line*' over networks.

```

@startuml
nwdiag {
    // define group at outside network definitions
    group {
        color = "#7777FF";

        web01;
        web02;
        db01;
    }

    network dmz {
        color = "pink"

        web01;
        web02;
    }

    network internal {
        web01;
        web02;
        db01 [shape = database ];
    }

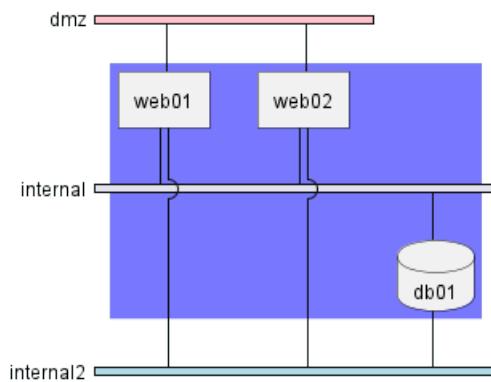
    network internal2 {
        color = "LightBlue";

        web01;
        web02;
        db01;
    }
}

```

}

```
@enduml
```



13.8 Peer networks

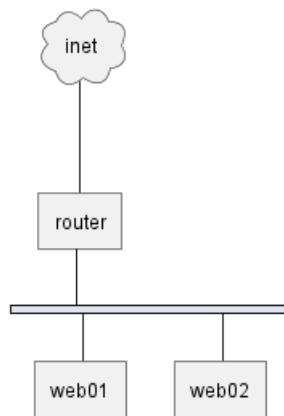
Peer networks are simple connections between two nodes, for which we don't use a horizontal "busbar" network

```

@startuml
nwdiag {
    inet [shape = cloud];
    inet -- router;

    network {
        router;
        web01;
        web02;
    }
}
@enduml

```



13.9 Peer networks and group

13.9.1 Without group

```

@startuml
nwdiag {
    internet [ shape = cloud];
    
```



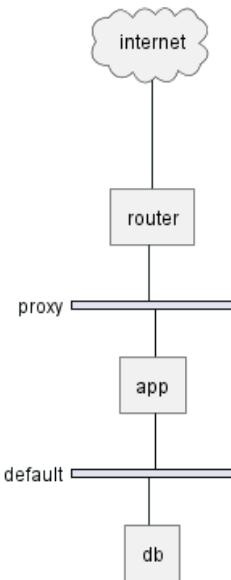
```

internet -- router;

network proxy {
    router;
    app;
}
network default {
    app;
    db;
}
}

@enduml

```



13.9.2 Group on first

```

@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    group {
        color = "pink";
        app;
        db;
    }

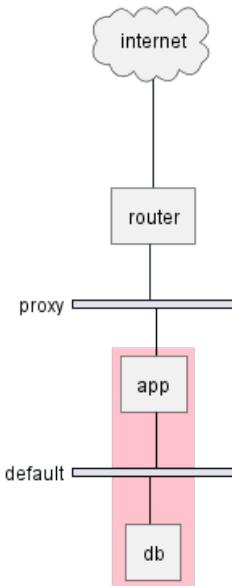
    network proxy {
        router;
        app;
    }

    network default {
        app;
        db;
    }
}

```



@enduml



13.9.3 Group on second

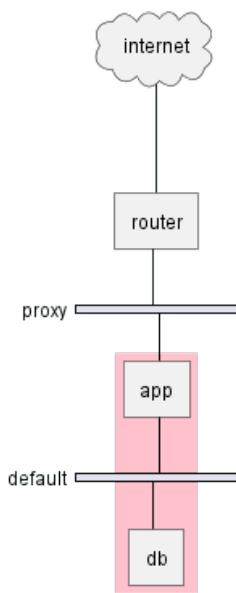
```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    network proxy {
        router;
        app;
    }

    group {
        color = "pink";
        app;
        db;
    }

    network default {
        app;
        db;
    }
}
@enduml
```



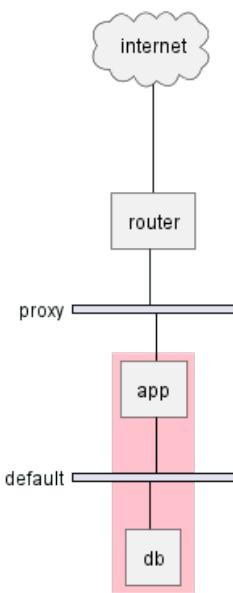


13.9.4 Group on third

```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    network proxy {
        router;
        app;
    }
    network default {
        app;
        db;
    }
    group {
        color = "pink";
        app;
        db;
    }
}
@enduml
```





[Ref. Issue#408 and QA-12655]

13.10 Add title, caption, header, footer or legend on network diagram

```
@startuml
```

```
header some header
```

```
footer some footer
```

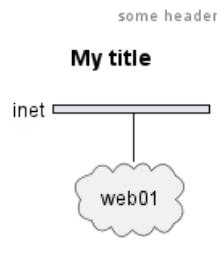
```
title My title
```

```
nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}
```

```
legend
The legend
end legend
```

```
caption This is caption
@enduml
```



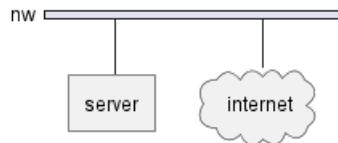


[Ref. QA-11303 and Common commands]

13.11 With or without shadow

13.11.1 With shadow (by default)

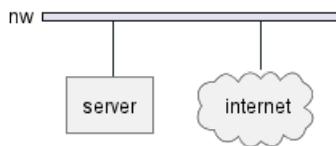
```
@startuml
nwdiag {
    network nw {
        server;
        internet;
    }
    internet [shape = cloud];
}
@enduml
```



13.11.2 Without shadow

```
@startuml
<style>
root {
    shadowing 0
}
</style>
nwdiag {
    network nw {
        server;
        internet;
    }
    internet [shape = cloud];
}
@enduml
```





[Ref. QA-14516]

13.12 Change width of the networks

You can change the width of the networks, especially in order to have the same full width for only some or all networks.

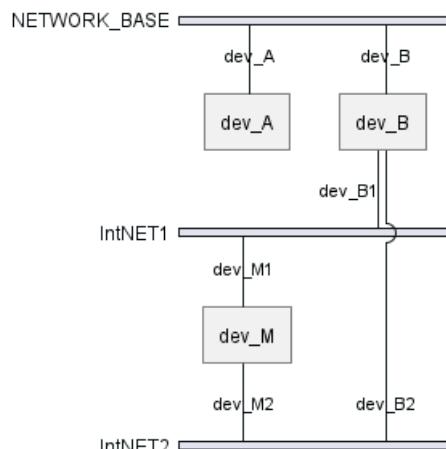
Here are some examples, with all the possibilities:

- without

```

@startuml
nwdiag {
    network NETWORK_BASE {
        dev_A [address = "dev_A" ]
        dev_B [address = "dev_B" ]
    }
    network IntNET1 {
        dev_B [address = "dev_B1" ]
        dev_M [address = "dev_M1" ]
    }
    network IntNET2 {
        dev_B [address = "dev_B2" ]
        dev_M [address = "dev_M2" ]
    }
}
@enduml

```



- only the first

```

@startuml
nwdiag {
    network NETWORK_BASE {
        width = full
        dev_A [address = "dev_A" ]
        dev_B [address = "dev_B" ]
    }
}

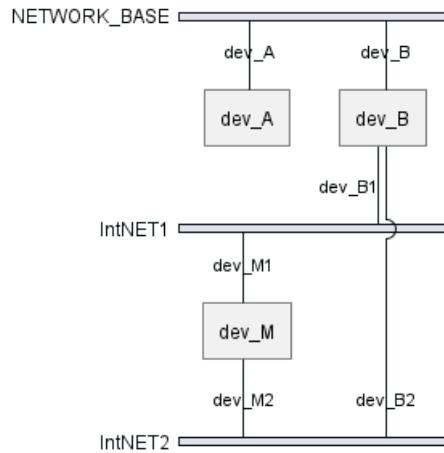
```



```

}
network IntNET1 {
    dev_B [address = "dev_B1" ]
    dev_M [address = "dev_M1" ]
}
network IntNET2 {
    dev_B [address = "dev_B2" ]
    dev_M [address = "dev_M2" ]
}
}
@enduml

```



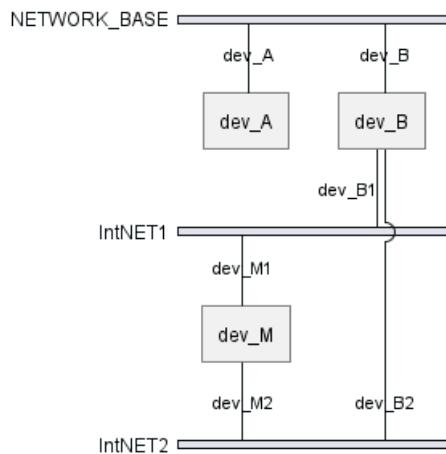
- the first and the second

```

@startuml
nwdiag {
    network NETWORK_BASE {
        width = full
        dev_A [address = "dev_A" ]
        dev_B [address = "dev_B" ]
    }
    network IntNET1 {
        width = full
        dev_B [address = "dev_B1" ]
        dev_M [address = "dev_M1" ]
    }
    network IntNET2 {
        dev_B [address = "dev_B2" ]
        dev_M [address = "dev_M2" ]
    }
}
@enduml

```



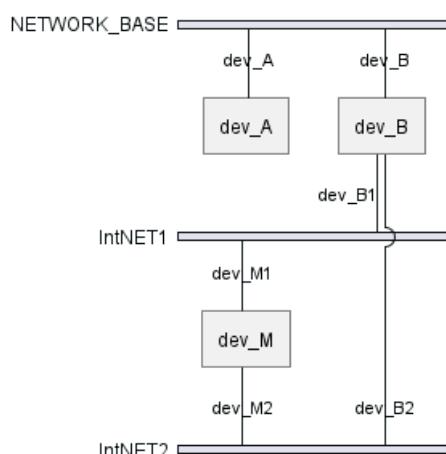


- all the network (with same full width)

```

@startuml
nwdiag {
    network NETWORK_BASE {
        width = full
        dev_A [address = "dev_A" ]
        dev_B [address = "dev_B" ]
    }
    network IntNET1 {
        width = full
        dev_B [address = "dev_B1" ]
        dev_M [address = "dev_M1" ]
    }
    network IntNET2 {
        width = full
        dev_B [address = "dev_B2" ]
        dev_M [address = "dev_M2" ]
    }
}
@enduml

```

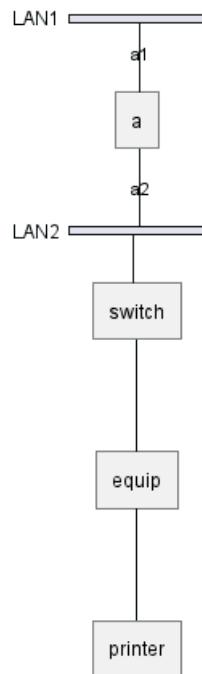


13.13 Other internal networks

You can define other internal networks (TCP/IP, USB, SERIAL,...).

- Without address or type

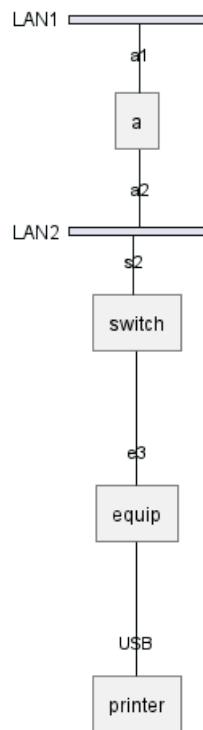
```
@startuml
nwdiag {
    network LAN1 {
        a [address = "a1"];
    }
    network LAN2 {
        a [address = "a2"];
        switch;
    }
    switch -- equip;
    equip -- printer;
}
@enduml
```



- With address or type

```
@startuml
nwdiag {
    network LAN1 {
        a [address = "a1"];
    }
    network LAN2 {
        a [address = "a2"];
        switch [address = "s2"];
    }
    switch -- equip;
    equip [address = "e3"];
    equip -- printer;
    printer [address = "USB"];
}
@enduml
```





[Ref. QA-12824]

13.14 Using (global) style

13.14.1 Without style (by default)

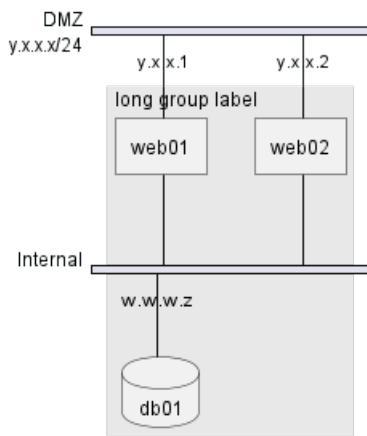
```

@startuml
nwdiag {
    network DMZ {
        address = "y.x.x.x/24"
        web01 [address = "y.x.x.1"];
        web02 [address = "y.x.x.2"];
    }

    network Internal {
        web01;
        web02;
        db01 [address = "w.w.w.z", shape = database];
    }

    group {
        description = "long group label";
        web01;
        web02;
        db01;
    }
}
@enduml
  
```





13.14.2 With style

You can use style to change rendering of elements.

```

@startuml
<style>
nwdiagDiagram {
    network {
        BackGroundColor green
        LineColor red
        LineThickness 1.0
        FontSize 18
        FontColor navy
    }
    server {
        BackGroundColor pink
        LineColor yellow
        LineThickness 1.0
        ' FontXXX only for description or label
        FontSize 18
        FontColor #blue
    }
    arrow {
        ' FontXXX only for address
        FontSize 17
        FontColor #red
        FontName Monospaced
        LineColor black
    }
    group {
        BackGroundColor cadetblue
        LineColor black
        LineThickness 2.0
        FontSize 11
        FontStyle bold
        Margin 5
        Padding 5
    }
}
</style>
nwdiag {

```



```

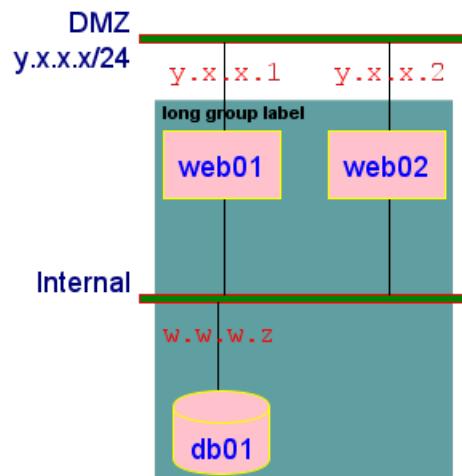
network DMZ {
    address = "y.x.x.x/24"
    web01 [address = "y.x.x.1"];
    web02 [address = "y.x.x.2"];
}

network Internal {
    web01;
    web02;
    db01 [address = "w.w.w.z", shape = database];
}

group {
    description = "long group label";
    web01;
    web02;
    db01;
}
}

@enduml

```



[Ref. QA-14479]

13.15 Appendix: Test of all shapes on Network diagram (nwdiag)

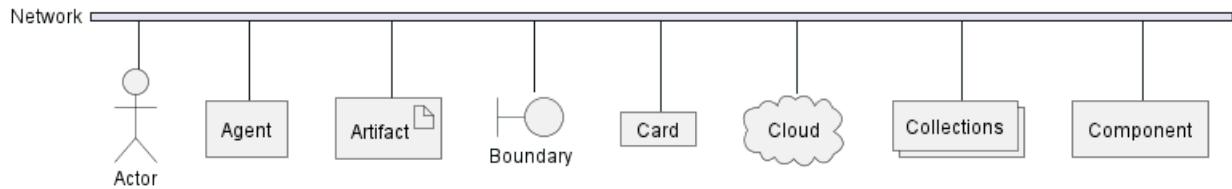
```

@startuml
nwdiag {
    network Network {
        Actor      [shape = actor]
        Agent      [shape = agent]
        Artifact   [shape = artifact]
        Boundary   [shape = boundary]
        Card       [shape = card]
        Cloud      [shape = cloud]
        Collections [shape = collections]
        Component   [shape = component]
    }
}
@enduml

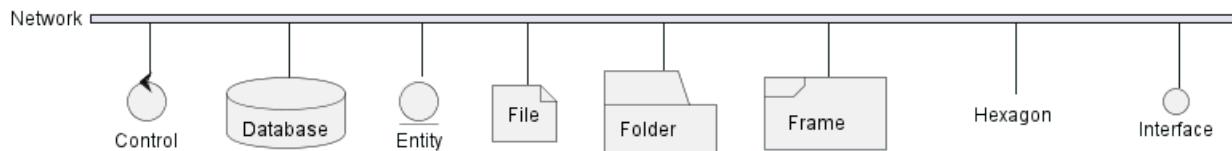
```



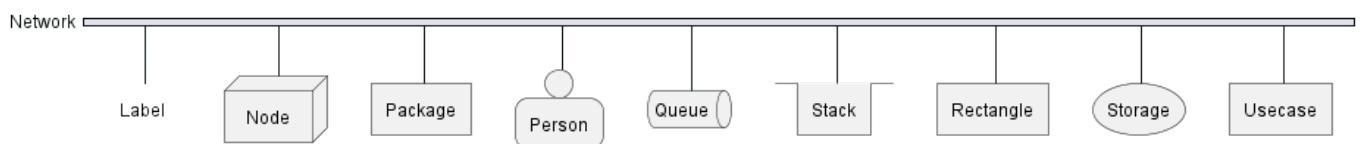
13.15 Appendix: Test of all shapes on Network diagram (nwdiag) NETWORK DIAGRAM (NWDIAG)



```
@startuml
nwdiag {
    network Network {
        Control      [shape = control]
        Database     [shape = database]
        Entity       [shape = entity]
        File         [shape = file]
        Folder       [shape = folder]
        Frame        [shape = frame]
        Hexagon      [shape = hexagon]
        Interface    [shape = interface]
    }
}
@enduml
```



```
@startuml
nwdiag {
    network Network {
        Label        [shape = label]
        Node         [shape = node]
        Package      [shape = package]
        Person       [shape = person]
        Queue        [shape = queue]
        Stack        [shape = stack]
        Rectangle    [shape = rectangle]
        Storage      [shape = storage]
        Usecase      [shape = usecase]
    }
}
@enduml
```

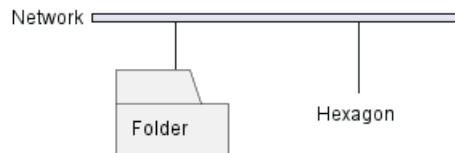


TODO: FIXME olli level 0 Overlap of label for folder olli level 0 Hexagon shape is missing olli ol

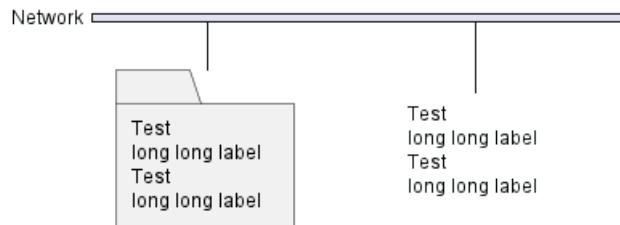


13.15 Appendix: Test of all shapes on Network diagram (nwdiag) NETWORK DIAGRAM (NWDIAG)

```
@startuml  
nwdiag {  
network Network {  
Folder [shape = folder]  
Hexagon [shape = hexagon]  
}  
}  
@enduml
```



```
@startuml  
nwdiag {  
network Network {  
Folder [shape = folder, description = "Test, long long label\\nTest, long long label"]  
Hexagon [shape = hexagon, description = "Test, long long label\\nTest, long long label"]  
}  
}  
@enduml
```



TODO: FIXME



14 Salt (Wireframe)

Salt is a subproject included in PlantUML that may help you to design graphical interface or *Website Wireframe or Page Schematic or Screen Blueprint*.

The goal of this tool is to discuss about simple and sample windows.

You can use either `@startsalt` keyword, or `@startuml` followed by a line with `salt` keyword.

14.1 Basic widgets

A window must start and end with brackets. You can then define:

- Button using [and].
- Radio button using (and).
- Checkbox using [and].
- User text area using ".
- Dropdown using ^.

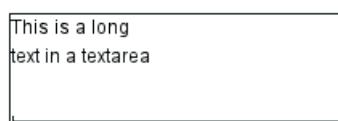
```
@startsalt
{
    Just plain text
    [This is my button]
    () Unchecked radio
    (X) Checked radio
    [] Unchecked box
    [X] Checked box
    "Enter text here"
    ^This is a dropdown^
}
@endsalt
```



14.2 Text area

Here is an attempt to create a text area:

```
@startsalt
{+
    This is a long
    text in a textarea
    .
    "
}
@endsalt
```



Note:



- the dot (.) to fill up vertical space;
- the last line of space (" ") to make the area wider.

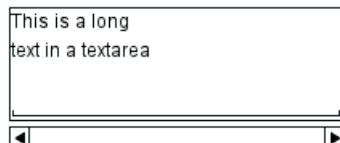
[Ref. QA-14765]

Then you can add scroll bar:

```
@startsalt
{SI
This is a long
text in a textarea
.
"
}
@endsalt
```



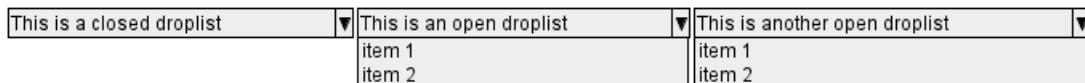
```
@startsalt
{S-
This is a long
text in a textarea
.
"
}
@endsalt
```



14.3 Open, close dropdown

You can open a dropdown, by adding values enclosed by ^, as:

```
@startsalt
{
^This is a closed dropdown^ |
^This is an open dropdown^^ item 1^^ item 2^ |
^This is another open dropdown^ item 1^ item 2^
}
@endsalt
```



[Ref. QA-4184]

14.4 Using grid [| and #, !, -, +]

A table is automatically created when you use an opening bracket {. And you have to use | to separate columns.

For example:

```
@startsalt
{
```



```

Login | "MyName"
Password | "****"
[Cancel] | [ OK ]
}
@endsalt

```

Login	<input type="text" value="MyName"/>
Password	<input type="password" value="****"/>
Cancel	OK

Just after the opening bracket, you can use a character to define if you want to draw lines or columns of the grid :

Symbol	Result
#	To display all vertical and horizontal lines
!	To display all vertical lines
-	To display all horizontal lines
+	To display external lines

```

@startsalt
{+
Login | "MyName"
Password | "****"
[Cancel] | [ OK ]
}
@endsalt

```

Login	<input type="text" value="MyName"/>
Password	<input type="password" value="****"/>
Cancel	OK

14.5 Group box [^]

```

@startsalt
{^"My group box"
Login | "MyName"
Password | "****"
[Cancel] | [ OK ]
}
@endsalt

```

My group box	
Login	<input type="text" value="MyName"/>
Password	<input type="password" value="****"/>
Cancel	OK

[Ref. QA-5840]

14.6 Using separator [.., ==, ~~, -]

You can use several horizontal lines as separator.

```

@startsalt
{
Text1
..
"Some field"
==
Note on usage
~~

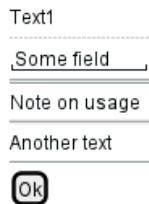
```



```

Another text
--
[Ok]
}
@endsalt

```



14.7 Tree widget [T]

To have a Tree, you have to start with {T} and to use + to denote hierarchy.

```

@startsalt
{
{T
+ World
++ America
+++ Canada
+++ USA
++++ New York
++++ Boston
+++ Mexico
++ Europe
+++ Italy
+++ Germany
++++ Berlin
++ Africa
}
}
@endsalt

```



14.8 Tree table [T]

You can combine trees with tables.

```

@startsalt
{
{T
+Region | Population | Age
+ World | 7.13 billion | 30
++ America | 964 million | 30
+++ Canada | 35 million | 30
+++ USA | 319 million | 30
++++ NYC | 8 million | 30

```



```

++++ Boston    | 617 thousand | 30
+++ Mexico    | 117 million  | 30
++ Europe     | 601 million  | 30
+++ Italy      | 61 million   | 30
+++ Germany    | 82 million   | 30
++++ Berlin    | 3 million    | 30
++ Africa     | 1 billion    | 30
}
}
@endsalt

```

Region	Population	Age
World	7.13 billion	30
America	964 million	30
Canada	35 million	30
USA	319 million	30
NYC	8 million	30
Boston	617 thousand	30
Mexico	117 million	30
Europe	601 million	30
Italy	61 million	30
Germany	82 million	30
Berlin	3 million	30
Africa	1 billion	30

And add lines.

```

@startsalt
{
..
== with T!
{T!
+Region      | Population   | Age
+ World      | 7.13 billion | 30
++ America    | 964 million  | 30
}
..
== with T-
{T-
+Region      | Population   | Age
+ World      | 7.13 billion | 30
++ America    | 964 million  | 30
}
..
== with T+
{T+
+Region      | Population   | Age
+ World      | 7.13 billion | 30
++ America    | 964 million  | 30
}
..
== with T#
{T#
+Region      | Population   | Age
+ World      | 7.13 billion | 30
++ America    | 964 million  | 30
}
..
}
@endsalt

```



with T!		
Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T-		
Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T+		
Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T#		
Region	Population	Age
World	7.13 billion	30
America	964 million	30

[Ref. QA-1265]

14.9 Enclosing brackets [,]

You can define subelements by opening a new opening bracket.

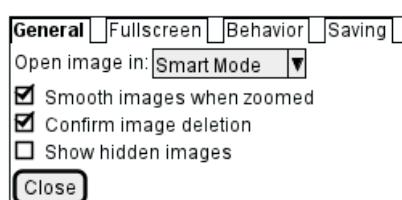
```
@startsalt
{
Name      | "
Modifiers: | { (X) public | () default | () private | () protected
           | [] abstract | [] final   | [] static }
Superclass: | { "java.lang.Object" | [Browse...]
}
@endsalt
```

Name	<input type="text"/>
Modifiers:	<input checked="" type="radio"/> public <input type="radio"/> default <input type="radio"/> private <input type="radio"/> protected <input type="checkbox"/> abstract <input type="checkbox"/> final <input type="checkbox"/> static
Superclass:	<input type="text"/> Browse...

14.10 Adding tabs [/]

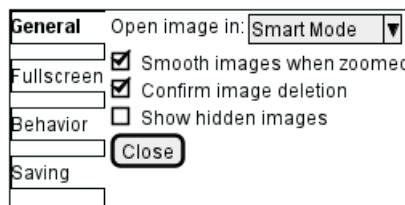
You can add tabs using{/ notation. Note that you can use HTML code to have bold text.

```
@startsalt
{+
{/ <b>General | Fullscreen | Behavior | Saving >
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```



Tab could also be vertically oriented:

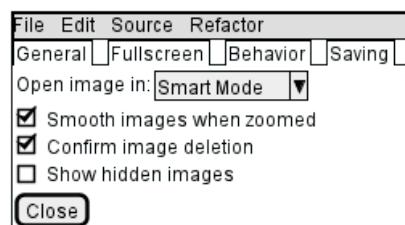
```
@startsalt
{+
{/ <b>General
Fullscreen
Behavior
Saving } |
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
[Close]
}
}
@endsalt
```



14.11 Using menu [*]

You can add a menu by using {*} notation.

```
@startsalt
{+
{* File | Edit | Source | Refactor }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```

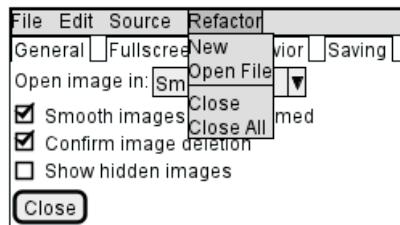


It is also possible to open a menu:

```
@startsalt
{+
{* File | Edit | Source | Refactor
Refactor | New | Open File | - | Close | Close All }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^ }
```

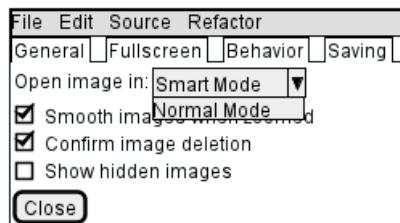


```
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```



Like it is possible to open a droplist:

```
@startsalt
{+
{* File | Edit | Source | Refactor }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^~Normal Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```



[Ref. QA-4184]

14.12 Advanced table

You can use two special notations for table :

- * to indicate that a cell with span with left
- . to denote an empty cell

```
@startsalt
{#
. | Column 2 | Column 3
Row header 1 | value 1 | value 2
Row header 2 | A long cell | *
}
@endsalt
```

	Column 2	Column 3
Row header 1	value 1	value 2
Row header 2	A long cell	*

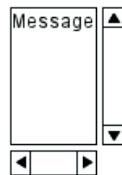


14.13 Scroll Bars [S, SI, S-]

You can use {S notation for scroll bar like in following examples:

- {S: for horizontal and vertical scrollbars

```
@startsalt
{S
Message
.
.
.
}
@endsalt
```



- {SI : for vertical scrollbar only

```
@startsalt
{SI
Message
.
.
.
}
@endsalt
```



- {S- : for horizontal scrollbar only

```
@startsalt
{S-
Message
.
.
.
}
@endsalt
```

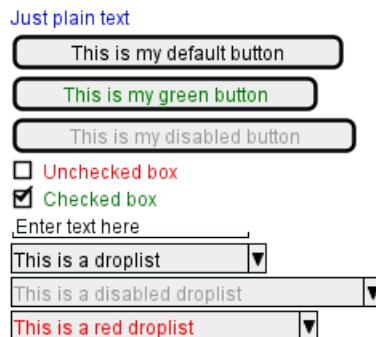


14.14 Colors

It is possible to change text color of widget.



```
@startsalt
{
<color:Blue>Just plain text
[This is my default button]
[<color:green>This is my green button]
[<color:#9a9a9a>This is my disabled button]
[] <color:red>Unchecked box
[X] <color:green>Checked box
"Enter text here"
~This is a dropdown^
~<color:#9a9a9a>This is a disabled dropdown^
~<color:red>This is a red dropdown^
}
@endsalt
```



[Ref. QA-12177]

14.15 Creole on Salt

You can use Creole or HTML Creole on salt:

```
@startsalt
{{^==Creole
This is **bold**
This is //italics//
This is ""monospaced"""
This is --stricken-out--
This is __underlined__
This is ~~wave-underlined~~
--test Unicode and icons--
This is <U+221E> long
This is a <&code> icon
Use image : <img:http://plantuml.com/logo3.png>
}|
{^<b>HTML Creole
This is <b>bold</b>
This is <i>italics</i>
This is <font:monospaced>monospaced</font>
This is <s>stroked</s>
This is <u>underlined</u>
This is <w>waved</w>
This is <s:green>stroked</s>
This is <u:red>underlined</u>
This is <w:#0000FF>waved</w>
-- other examples --
This is <color:blue>Blue</color>
This is <back:orange>Orange background</back>
This is <size:20>big</size>
```



```
}

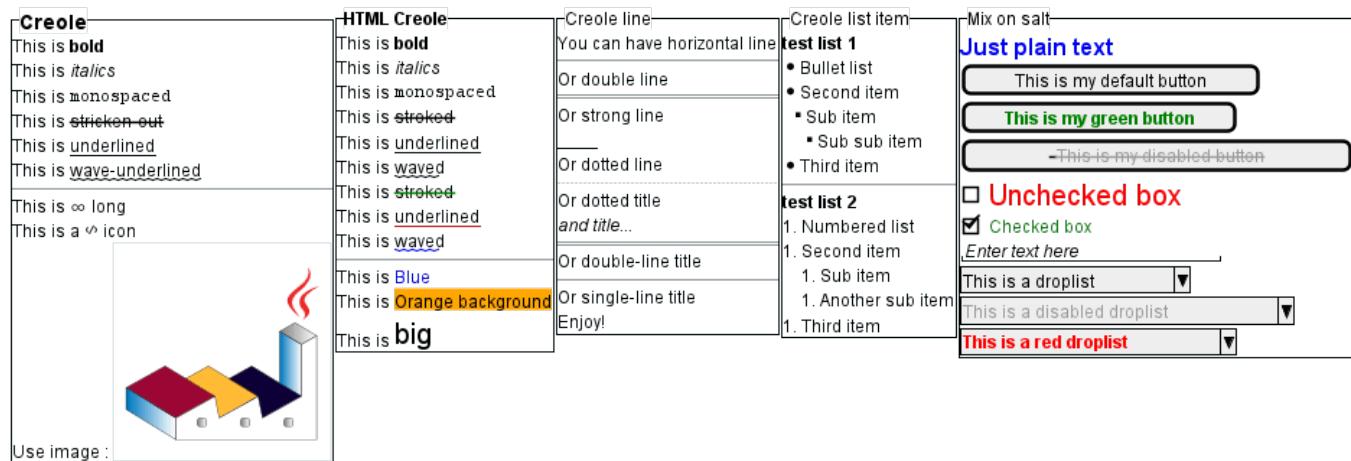
{^Creole line
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!
}

{^Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}

{^Mix on salt
==<color:Blue>Just plain text
[This is my default button]
[<b><color:green>This is my green button]
[ ---<color:#9a9a9a>This is my disabled button-- ]
[] <size:20><color:red>Unchecked box
[X] <color:green>Checked box
"/Enter text here//"
~This is a dropdown~
~<color:#9a9a9a>This is a disabled dropdown~
~<b><color:red>This is a red dropdown~
}

@endsalt
```





14.16 Pseudo sprite [«, »]

Using << and >> you can define a pseudo-sprite or sprite-like drawing and reusing it latter.

```
@startsalt
{
[X] checkbox | [] checkbox
() radio | (X) radio
This is a text | [This is my button] | This is another text
"A field" | "Another long Field" | [A button]
<<folder
.....
.XXXXXX.....
.X...X.....
.XXXXXXXXXXX.
.X.....X.
.X.....X.
.X.....X.
.X.....X.
.X.....X.
.XXXXXXXXXXX.
.....
>>|<color:blue>other folder|<<folder>>
^Dropelist^
}
@endsalt
```



[Ref. QA-5849]

14.17 OpenIconic

OpenIconic is a very nice open source icon set. Those icons have been integrated into the creole parser, so you can use them out-of-the-box. You can use the following syntax: <&ICON_NAME>.

```
@startsalt
{
Login<&person> | "MyName"
Password<&key> | "*****"
```



```
[Cancel <&circle-x>] | [OK <&account-login>]
}
@endsalt
```

Login	<input type="text" value="MyName"/>
Password	<input type="password" value="****"/>
<input type="button" value="Cancel"/>	<input type="button" value="OK"/>

The complete list is available on OpenIconic Website, or you can use the following special diagram:

```
@startuml  
listopeniconic  
@enduml
```

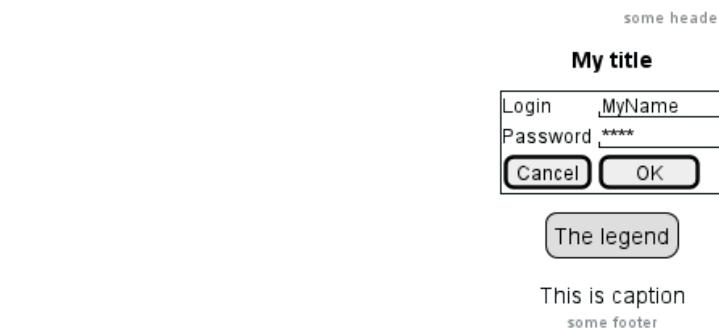
List Open Iconic	bell	cloud	excerpt	justify-right	musical-note	star
Credit to https://useiconic.com/open	✉ bell	☁️ cloud	_excerpt	☞ justify-right	📎 musical-note	★ star
▢ account-login	❸ bluetooth	☁️ cloudy	↳ expand-down	🔑 key	📎 paperclip	☀ sun
▢ account-logout	❹ bold	⌚ code	↳ expand-left	💻 laptop	✍ pencil	▢ tablet
↶ action-redo	✚ bolt	⚙ cog	↳ expand-right	☷ layers	👤 people	🏷 tag
↶ action-undo	❻ book	-collapse-down	↳ expand-up	💡 lightbulb	👤 person	🏷 tags
☰ align-center	❻ bookmark	↳ collapse-left	↳ external-link	🔗 link-broken	☎ phone	🎯 target
☰ align-left	➋ box	↳ collapse-right	👁 eye	🔗 link-intact	📊 pie-chart	☒ task
☰ align-right	❻ briefcase	↳ collapse-up	eyedropper	☷ list-rich	📍 pin	terminal
☰ align-top	💷 british-pound	⌘ command	📄 file	☷ list	● play-circle	TEXT
☰ browser	➋ comment-square	⌚ comment-square	🔥 fire	↗ location	➕ plus	👉 thumb-down
✖ brush	⌚ compass	⚡ flag	🔒 lock-locked	⚡ power-standby	👍 thumb-up	
✖ bug	ⓧ contrast	⚡ flash	🔓 lock-unlocked	🖨 print	⌚ timer	
⩲ bullhorn	⠇ copywriting	📁 folder	⟳ loop-circular	▣ project	➡ transfer	
⌚ arrow-circle-bottom	⌚ calculator	💳 credit-card	🍴 fork	⟳ loop-square	▶ pulse	-trash
⌚ arrow-circle-left	📅 calendar	▢ crop	▶ fullscreen-enter	▷ loop	🎴 puzzle-piece	underline
⌚ arrow-circle-right	⌚ camera-slr	⌚ dashboard	✖ fullscreen-exit	🔍 magnifying-glass	❓ question-mark	⬇ vertical-align-bottom
⌚ arrow-circle-top	▼ caret-bottom	⬇ data-transfer-download	🌐 globe	📍 map-marker	🌧 rain	⬇ vertical-align-center
← arrow-left	◀ caret-left	⬇ data-transfer-upload	📈 graph	gMaps	🔀 random	⬇ vertical-align-top
→ arrow-right	◀ caret-right	✖ delete	grid-four-up	⏸ media-pause	⟳ reload	▶ video
↓ arrow-thick-bottom	▲ caret-top	⌚ dial	grid-three-up	▶ media-play	▷ resize-both	▶ volume-high
← arrow-thick-left	⩲ cart	⌚ document	grid-two-up	● media-record	↑ resize-height	◀ volume-low
→ arrow-thick-right	⩲ chat	">\$ dollar	-hard-drive	◀ media-skip-backward	↔ resize-width	◀ volume-off
↑ arrow-thick-top	✓ check	“ double-quote-sans-left	HEADER	▶ media-skip-forward	▶ rss-alt	⚠ warning
↑ arrow-top	⩲ chevron-bottom	“ double-quote-sans-right	⌚ headphones	◀ media-step-backward	▶ rss	ⓘ wifi
㎇ audio-spectrum	◀ chevron-left	“ double-quote-serif-left	❤ heart	▶ media-step-forward	▶ script	🔧 wrench
۾ audio	⩲ chevron-right	“ double-quote-serif-right	⌂ home	▶ media-stop	▶ share-boxed	✖ x
➊ badge	⩲ chevron-top	滴 droplet	🖼 image	▶ medical-cross	▶ share	¥ yen
➊ ban	circle-check	▲ eject	✉ inbox	☰ menu	🛡 shield	Ⓜ zoom-in
▢ bar-chart	➊ circle-x	▲ elevator	♾ infinity	🎤 microphone	📶 signal	Ⓜ zoom-out
⩲ basket	⩲ clipboard	„ ellipses	ℹ info	‐ minus	▶ signpost	
▢ battery-empty	⌚ clock	✉ envelope-closed	italic	💻 monitor	▶ sort-ascending	
▢ battery-full	⩲ cloud-download	✉ envelope-open	justify-center	🌙 moon	▶ sort-descending	
⩲ beaker	⩲ cloud-upload	€ euro	justify-left	+ move	spreadsheet	

14.18 Add title, header, footer, caption or legend

```
@startsalt
title My title
header some header
footer some footer
caption This is caption
legend
The legend
end legend
```

```
{+  
    Login      | "MyName"      "  
    Password   | "****"        "  
    [Cancel]   | [ OK ]  
}
```

@endsalt

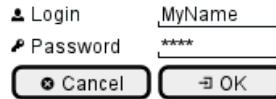


(See also: *Common commands*)

14.19 Zoom, DPI

14.19.1 Whitout zoom (by default)

```
@startsalt
{
    <&person> Login | "MyName"
    <&key> Password | "****"
    [<&circle-x> Cancel] | [<&account-login> OK]
}
@endsalt
```



14.19.2 Scale

You can use the `scale` command to zoom the generated image.

You can use either a number or a fraction to define the scale factor. You can also specify either width or height (in pixel). And you can also give both width and height: the image is scaled to fit inside the specified dimension.

```
@startsalt
scale 2
{
    <&person> Login | "MyName"
    <&key> Password | "****"
    [<&circle-x> Cancel] | [<&account-login> OK]
}
@endsalt
```



(See also: *Zoom on Common commands*)

14.19.3 DPI

You can also use the `skinparam dpi` command to zoom the generated image.

```
@startsalt
```



```

skinparam dpi 200
{
    <&person> Login | "MyName"
    <&key> Password | "****"
    [<&circle-x> Cancel] | [ <&account-login> OK ]
}
@endsalt

```



14.20 Include Salt "on activity diagram"

You can read the following explanation.

```

@startuml
(*) --> "
{{ salt
{+
<b>an example
choose one option
()one
()two
[ok]
}
}}
" as choose

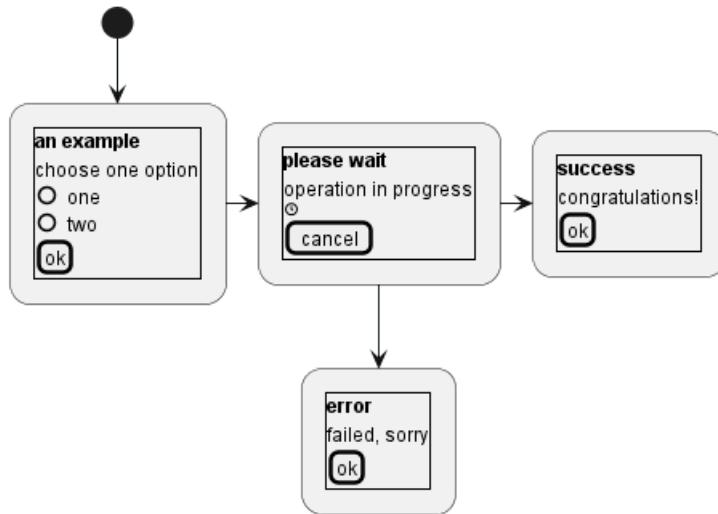
choose -right-> "
{{ salt
{+
<b>please wait
operation in progress
<&clock>
[cancel]
}
}}
" as wait
wait -right-> "
{{ salt
{+
<b>success
congratulations!
[ok]
}
}}
" as success

wait -down-> "
{{ salt

```



```
{
<b>error
failed, sorry
[ok]
}
}
"
@enduml
```



It can also be combined with define macro.

```
@startuml
!unquoted procedure SALT($x)
"{
salt
%invoke_procedure("_"+$x)
}" as $x
!endprocedure

!procedure _choose()
{+
<b>an example
choose one option
()one
()two
[ok]
}
!endprocedure

!procedure _wait()
{+
<b>please wait
operation in progress
<&clock>
[cancel]
}
!endprocedure

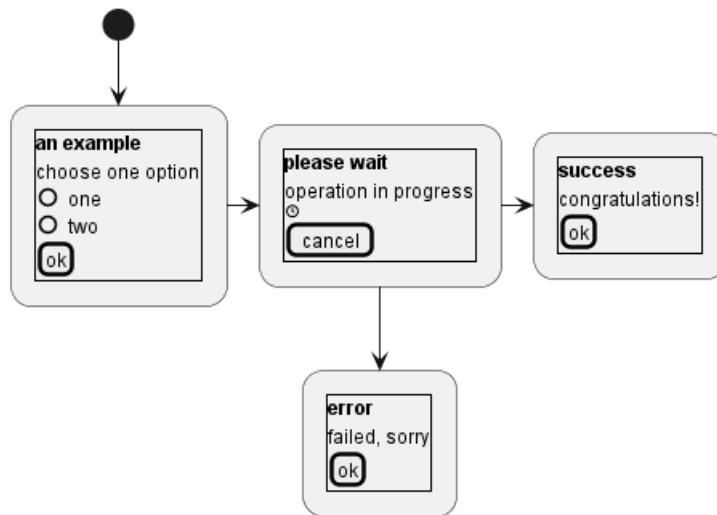
!procedure _success()
{+
<b>success
congratulations!
}
```



```
[ok]
}
!endprocedure

!procedure _error()
{+
<b>error
failed, sorry
[ok]
}
!endprocedure

(*) --> SALT(choose)
-right-> SALT(wait)
wait -right-> SALT(success)
wait -down-> SALT(error)
@enduml
```

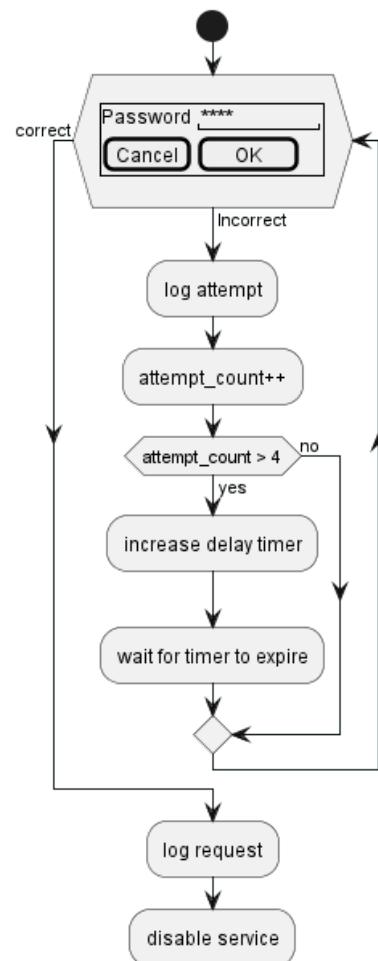


14.21 Include salt "on while condition of activity diagram"

You can include salt on while condition of activity diagram.

```
@startuml
start
while (\n{\nsalt\n{+\nPassword | "****" "\n[Cancel] | [ OK ]}\n}) is (Incorrect)
:log attempt;
:attempt_count++;
if (attempt_count > 4) then (yes)
:increase delay timer;
:wait for timer to expire;
else (no)
endif
endwhile (correct)
:log request;
:disable service;
@enduml
```





[Ref. QA-8547]

14.22 Include salt "on repeat while condition of activity diagram"

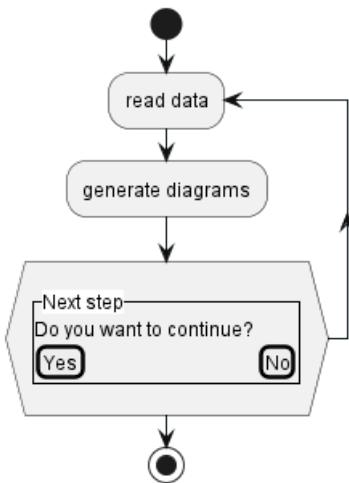
You can include salt on 'repeat while' condition of activity diagram.

```

@startuml
start
repeat :read data;
    :generate diagrams;
repeat while (\n{\nsalt\n{"Next step"\n  Do you want to continue? \n[Yes] | [No]\n}}\n)
stop
@enduml

```





[Ref. QA-14287]

14.23 Skinparam

You can use [only] some skinparam command to change the skin of the drawing.

Some example:

```
@startsalt
skinparam Backgroundcolor palegreen
{+
  Login | "MyName"
  Password | "****"
  [Cancel] | [ OK ]
}
@endsalt
```



```
@startsalt
skinparam handwritten true
{+
  Login | "MyName"
  Password | "****"
  [Cancel] | [ OK ]
}
@endsalt
```



TODO: FIXME FYI, some other skinparam does not work with salt, as:

```
@startsalt
skinparam defaultFontName monospaced
{+
  Login | "MyName"
  Password | "****"
  [Cancel] | [ OK ]
}
@endsalt
```





14.24 Style

You can use [only] some style command to change the skin of the drawing.

Some example:

```
@startsalt
<style>
saltDiagram {
    backgroundColor palegreen
}
</style>
{+
    Login | "MyName"
    Password | "****"
    [Cancel] | [ OK ]
}
@endsalt
```



TODO: FIXME FYI, some other style does not work with salt, as:

```
@startsalt
<style>
saltDiagram {
    Fontname Monospaced
    FontSize 10
    FontStyle italic
    LineThickness 0.5
    LineColor red
}
</style>
{+
    Login | "MyName"
    Password | "****"
    [Cancel] | [ OK ]
}
@endsalt
```



[Ref. QA-13460]



15 Archimate Diagram

This is only a proposal and subject to change.

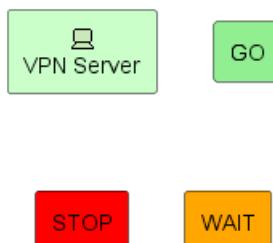
You are very welcome to create a new discussion on this future syntax. Your feedbacks, ideas and suggestions help us to find the right solution.

15.1 Archimate keyword

You can use the `archimate` keyword to define an element. Stereotype can optionally specify an additional icon. Some colors (Business, Application, Motivation, Strategy, Technology, Physical, Implementation) are also available.

```
@startuml
archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
@enduml
```



15.2 Defining Junctions

Using the `circle` keyword and the preprocessor, you can also create junctions.

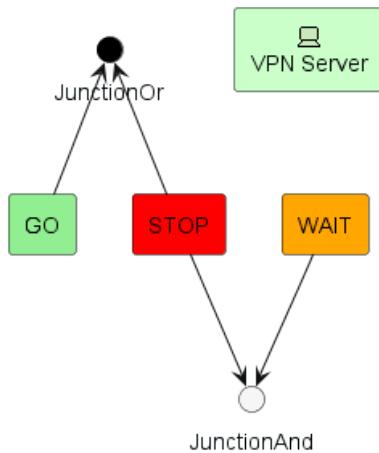
```
@startuml
!define Junction_Or circle #black
!define Junction_And circle #whitesmoke

Junction_And JunctionAnd
Junction_Or JunctionOr

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
GO -up-> JunctionOr
STOP -up-> JunctionOr
STOP -down-> JunctionAnd
WAIT -down-> JunctionAnd
@enduml
```





15.3 Example 1

```

@startuml
skinparam rectangle<<behavior>> {
roundCorner 25
}
sprite $bProcess jar:archimate/business-process
sprite $aService jar:archimate/application-service
sprite $aComponent jar:archimate/application-component

rectangle "Handle claim" as HC <<$bProcess>><<behavior>> #Business
rectangle "Capture Information" as CI <<$bProcess>><<behavior>> #Business
rectangle "Notify\nAdditional Stakeholders" as NAS <<$bProcess>><<behavior>> #Business
rectangle "Validate" as V <<$bProcess>><<behavior>> #Business
rectangle "Investigate" as I <<$bProcess>><<behavior>> #Business
rectangle "Pay" as P <<$bProcess>><<behavior>> #Business

HC *--down- CI
HC *--down- NAS
HC *--down- V
HC *--down- I
HC *--down- P

CI -right->> NAS
NAS -right->> V
V -right->> I
I -right->> P

rectangle "Scanning" as scanning <<$aService>><<behavior>> #Application
rectangle "Customer admnistration" as customerAdministration <<$aService>><<behavior>> #Application
rectangle "Claims admnistration" as claimsAdministration <<$aService>><<behavior>> #Application
rectangle Printing <<$aService>><<behavior>> #Application
rectangle Payment <<$aService>><<behavior>> #Application

scanning -up-> CI
customerAdministration -up-> CI
claimsAdministration -up-> NAS
claimsAdministration -up-> V
claimsAdministration -up-> I
Payment -up-> P

Printing -up-> V
Printing -up-> P
  
```



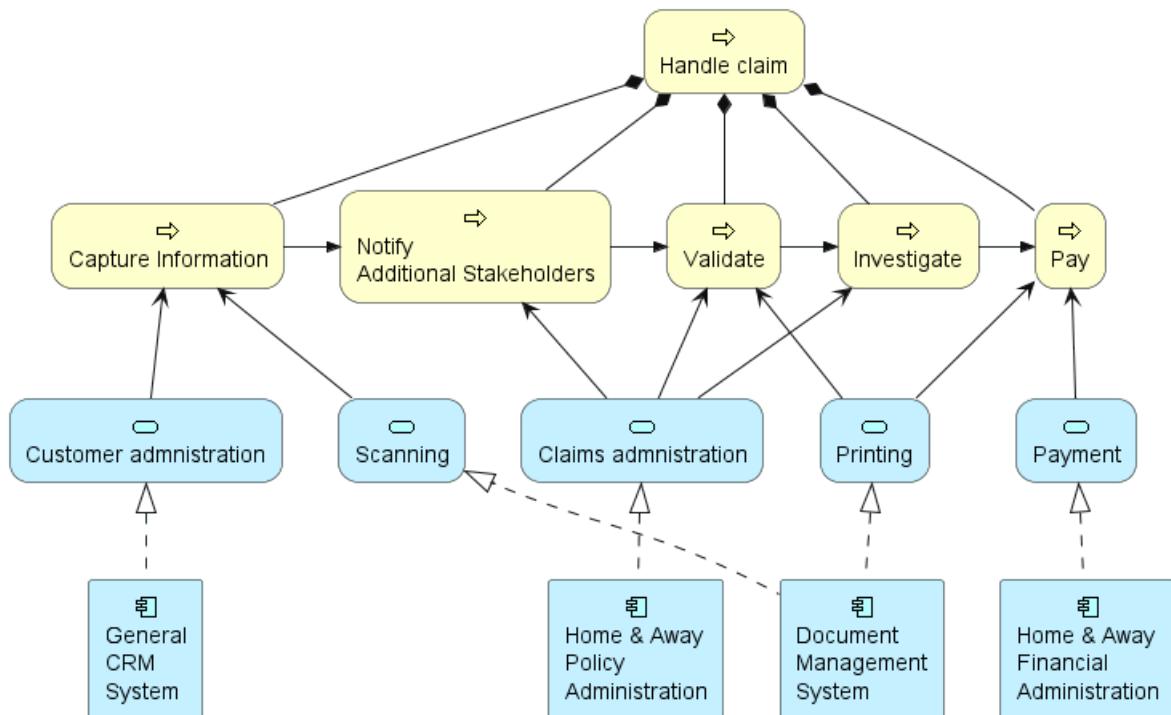
```

rectangle "Document\Management\System" as DMS <<$aComponent>> #Application
rectangle "General\CRM\System" as CRM <<$aComponent>> #Application
rectangle "Home & Away\Policy\Administration" as HAPA <<$aComponent>> #Application
rectangle "Home & Away\Financial\Administration" as HFPA <<$aComponent>> #Application

DMS .up.|> scanning
DMS .up.|> Printing
CRM .up.|> customerAdministration
HAPA .up.|> claimsAdministration
HFPA .up.|> Payment

legend left
Example from the "Archisurance case study" (OpenGroup).
See
=====
<$bProcess> :business process
=====
<$aService> : application service
=====
<$aComponent> : application component
endlegend
@enduml

```



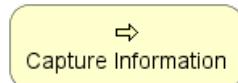
Example from the "Archisurance case study" (OpenGroup).
See
⇒ :business process
⊐ : application service
⊑ : application component

15.4 Example 2

@startuml



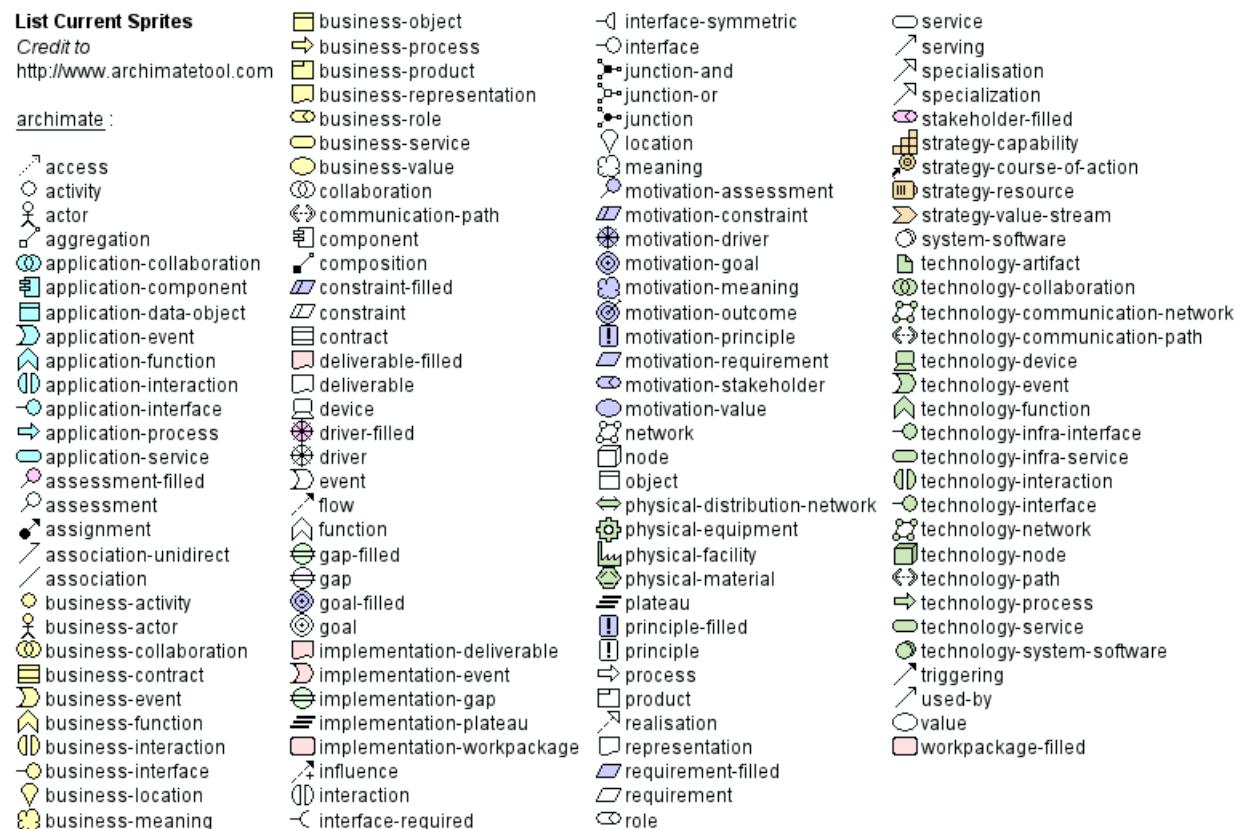
```
skinparam roundcorner 25
rectangle "Capture Information" as CI <<$archimate/business-process>> #Business
@enduml
```



15.5 List possible sprites

You can list all possible sprites for Archimate using the following diagram:

```
@startuml
listsprite
@enduml
```



15.6 ArchiMate Macros

15.6.1 Archimate Macros and Library

A list of Archimate macros are defined Archimate-PlantUML here which simplifies the creation of ArchiMate diagrams, and Archimate is natively on the Standard Library of PlantUML.

15.6.2 Archimate elements

Using the macros, creation of ArchiMate elements are done using the following format: `Category_ElementName(nameOfThe "description")`

For example:

- To define a *Stakeholder* element, which is part of Motivation category, the syntax will be `Motivation_Stakeholder("Stakeholder Description")`:



```
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
@enduml
```



- To define a *Business Service* element, `Business_Service(BService, "Business Service")`:

```
@startuml
!include <archimate/Archimate>
Business_Service(BService, "Business Service")
@enduml
```



15.6.3 Archimate relationships

The ArchiMate relationships are defined with the following pattern: `Rel_RelationType(fromElement, toElement, "description")` and to define the direction/orientation of the two elements: `Rel_RelationType_Direction toElement, "description")`

The **RelationTypes** supported are:

- Access
- Aggregation
- Assignment
- Association
- Composition
- Flow
- Influence
- Realization
- Serving
- Specialization
- Triggering

The **Directions** supported are:

- Up
- Down
- Left
- Right

For example:

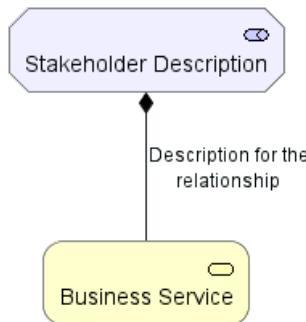
- To denote a composition relationship between the *Stakeholder* and *Business Service* defined above, the syntax will be

```
Rel_Composition(StakeholderElement, BService, "Description for the relationship")
```

```
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
```



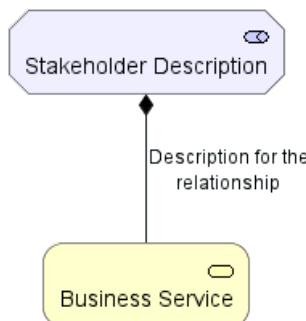
```
Business_Service(BService, "Business Service")
Rel_Composition(StakeholderElement, BService, "Description for the relationship")
@enduml
```



- Unordered List Item To orient the two elements in top - down position, the syntax will be

```
Rel_Composition_Down(StakeholderElement, BService, "Description for the relationship")

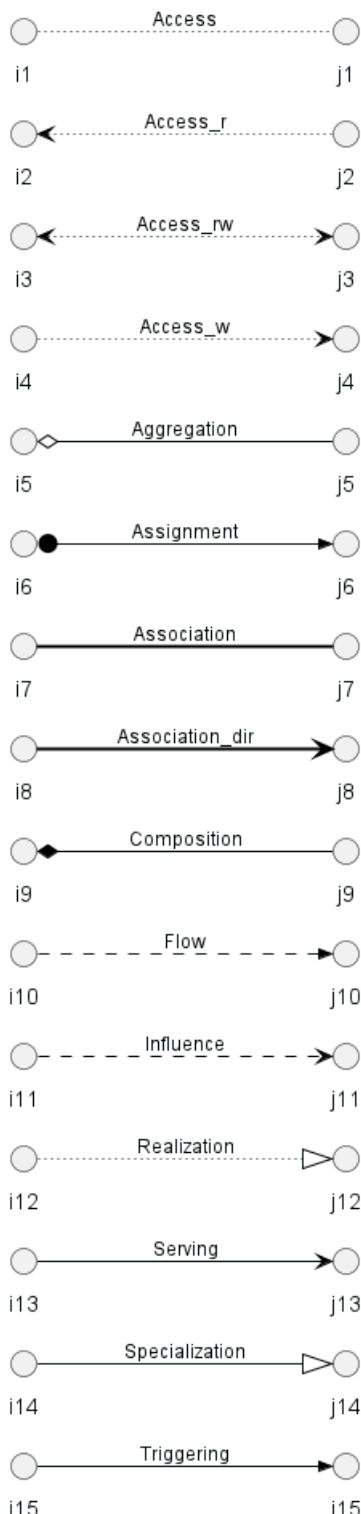
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
Business_Service(BService, "Business Service")
Rel_Composition_Down(StakeholderElement, BService, "Description for the relationship")
@enduml
```



15.6.4 Appendix: Examples of all Archimate RelationTypes

```
@startuml
left to right direction
skinparam nodesep 4
!include <archimate/Archimate>
Rel_Triggering(i15, j15, Triggering)
Rel_Specialization(i14, j14, Specialization)
Rel_Serving(i13, j13, Serving)
Rel_Realization(i12, j12, Realization)
Rel_Influence(i11, j11, Influence)
Rel_Flow(i10, j10, Flow)
Rel_Composition(i9, j9, Composition)
Rel_Association_dir(i8, j8, Association_dir)
Rel_Association(i7, j7, Association)
Rel_Assignment(i6, j6, Assignment)
Rel_Aggregation(i5, j5, Aggregation)
Rel_Access_w(i4, j4, Access_w)
Rel_Access_rw(i3, j3, Access_rw)
Rel_Access_r(i2, j2, Access_r)
Rel_Access(i1, j1, Access)
@enduml
```





```
@startuml
title ArchiMate Relationships Overview
skinparam nodesep 5
<style>
interface {
    shadowing 0
    backgroundcolor transparent
    linecolor transparent
    FontColor transparent
}
```



```

}

</style>
!include <archimate/Archimate>
left to right direction

rectangle Other {
() i14
() j14
}

rectangle Dynamic {
() i10
() j10
() i15
() j15
}

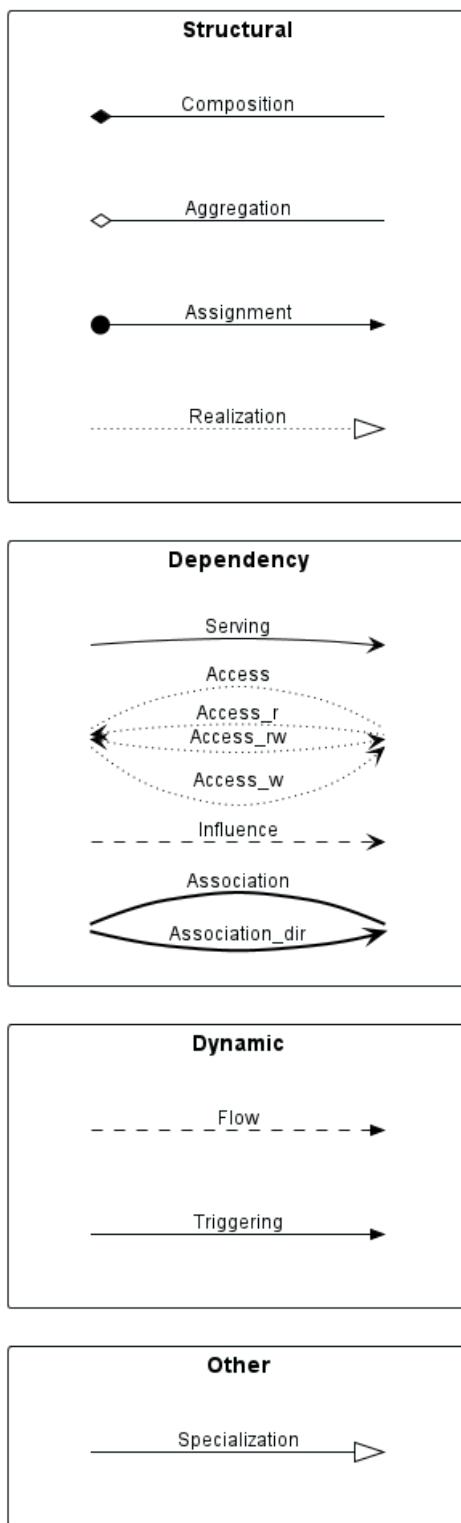
rectangle Dependency {
() i13
() j13
() i4
() j4
() i11
() j11
() i7
() j7
}

rectangle Structural {
() i9
() j9
() i5
() j5
() i6
() j6
() i12
() j12
}

Rel_Triggering(i15, j15, Triggering)
Rel_Specialization(i14, j14, Specialization)
Rel_Serving(i13, j13, Serving)
Rel_Realization(i12, j12, Realization)
Rel_Influence(i11, j11, Influence)
Rel_Flow(i10, j10, Flow)
Rel_Composition(i9, j9, Composition)
Rel_Association_dir(i7, j7, \nAssociation_dir)
Rel_Association(i7, j7, Association)
Rel_Assignment(i6, j6, Assignment)
Rel_Aggregation(i5, j5, Aggregation)
Rel_Access_w(i4, j4, Access_w)
Rel_Access_rw(i4, j4, Access_rw)
Rel_Access_r(i4, j4, Access_r)
Rel_Access(i4, j4, Access)
@enduml

```



ArchiMate Relationships Overview

[Adapted from Archimate PR#25]



16 Gantt Diagram

The Gantt is described in *natural* language, using very simple sentences (subject-verb-complement).

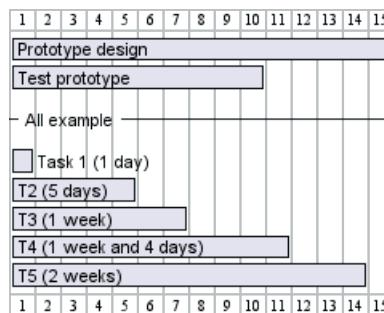
16.1 Declaring tasks

Tasks defined using square bracket.

16.1.1 Duration

Their durations are defined using the `lasts` verb:

```
@startgantt
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
-- All example --
[Task 1 (1 day)] lasts 1 day
[T2 (5 days)] lasts 5 days
[T3 (1 week)] lasts 1 week
[T4 (1 week and 4 days)] lasts 1 week and 4 days
[T5 (2 weeks)] lasts 2 weeks
@endgantt
```



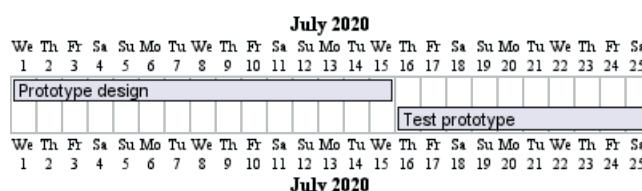
A week is a synonym for how many non-closed days are in a week. So if you specify Saturday and Sunday as closed, a week will be equivalent to 5 days

16.1.2 Start

Their beginning are defined using the `start` verb:

```
@startgantt
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
```

```
Project starts 2020-07-01
[Prototype design] starts 2020-07-01
[Test prototype] starts 2020-07-16
@endgantt
```

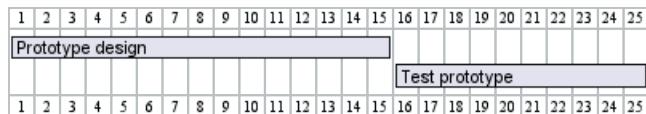


```
@startgantt
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
```

[Prototype design] starts D+0



[Test prototype] starts D+15
 @endgantt



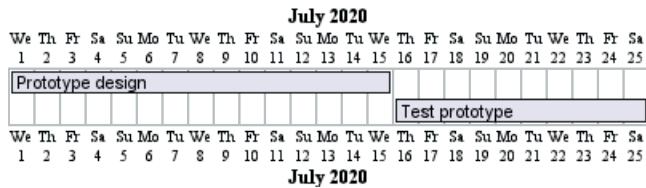
[Ref. for D+nn form: QA-14494]

16.1.3 End

Their endings are defined using the end verb:

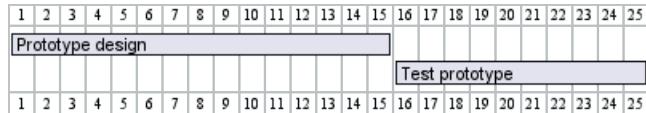
```
@startgantt
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
```

Project starts 2020-07-01
 [Prototype design] ends 2020-07-15
 [Test prototype] ends 2020-07-25
 @endgantt



```
@startgantt
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
```

[Prototype design] ends D+14
 [Test prototype] ends D+24
 @endgantt

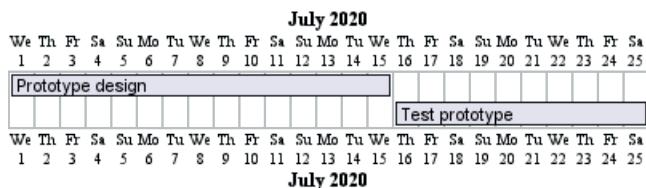


16.1.4 Start/End

It is possible to define both absolutely, by specifying dates:

```
@startgantt
Project starts 2020-07-01
[Prototype design] starts 2020-07-01
[Test prototype] starts 2020-07-16
[Prototype design] ends 2020-07-15
[Test prototype] ends 2020-07-25
```

@endgantt



@startgantt

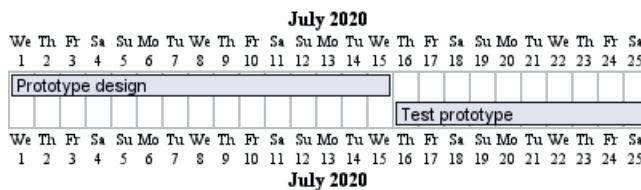
```
[Prototype design] starts D+0
[Test prototype] starts D+15
[Prototype design] ends D+14
[Test prototype] ends D+24
@endgantt
```



16.2 One-line declaration (with the and conjunction)

It is possible to combine declaration on one line with the `and` conjunction.

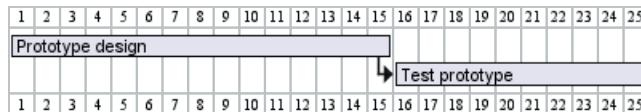
```
@startgantt
Project starts 2020-07-01
[Prototype design] starts 2020-07-01 and ends 2020-07-15
[Test prototype] starts 2020-07-16 and lasts 10 days
@endgantt
```



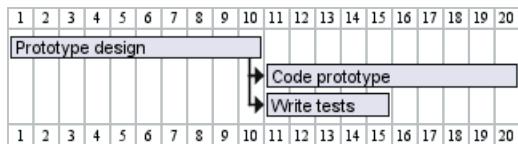
16.3 Adding constraints

It is possible to add constraints between tasks.

```
@startgantt
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
[Test prototype] starts at [Prototype design]'s end
@endgantt
```



```
@startgantt
[Prototype design] lasts 10 days
[Code prototype] lasts 10 days
[Write tests] lasts 5 days
[Code prototype] starts at [Prototype design]'s end
[Write tests] starts at [Code prototype]'s start
@endgantt
```



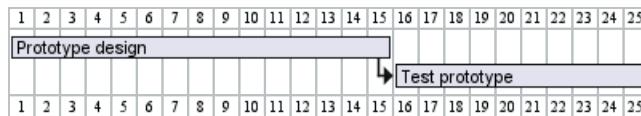
16.4 Short names

It is possible to define short name for tasks with the `as` keyword.

```
@startgantt
[Prototype design] as [D] lasts 15 days
[Test prototype] as [T] lasts 10 days
```



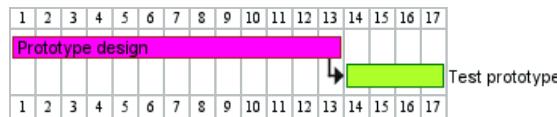
```
[T] starts at [D]'s end
@endgantt
```



16.5 Customize colors

It is also possible to customize colors with `is colored in`.

```
@startgantt
[Prototype design] lasts 13 days
[Test prototype] lasts 4 days
[Test prototype] starts at [Prototype design]'s end
[Prototype design] is colored in Fuchsia/FireBrick
[Test prototype] is colored in GreenYellow/Green
@endgantt
```



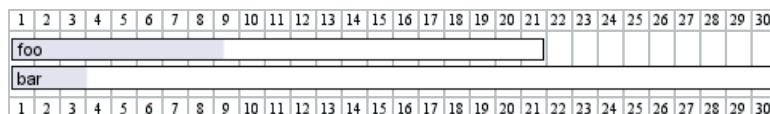
16.6 Completion status

16.6.1 Adding completion depending percentage

You can set the completion status of a task, by the command:

- `is xx% completed`
- `is xx% complete`

```
@startgantt
[foo] lasts 21 days
[foo] is 40% completed
[bar] lasts 30 days and is 10% complete
@endgantt
```



16.6.2 Change colour of completion (by style)

```
@startgantt
```

```
<style>
ganttDiagram {
    task {
        BackGroundColor GreenYellow
        LineColor Green
        unstarted {
            BackGroundColor Fuchsia
            LineColor FireBrick
        }
    }
}
</style>
```

`[Prototype design] lasts 7 days`

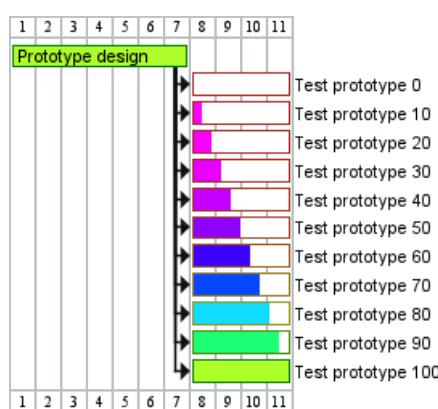


```
[Test prototype 0] lasts 4 days
[Test prototype 10] lasts 4 days
[Test prototype 20] lasts 4 days
[Test prototype 30] lasts 4 days
[Test prototype 40] lasts 4 days
[Test prototype 50] lasts 4 days
[Test prototype 60] lasts 4 days
[Test prototype 70] lasts 4 days
[Test prototype 80] lasts 4 days
[Test prototype 90] lasts 4 days
[Test prototype 100] lasts 4 days

[Test prototype 0] starts at [Prototype design]'s end
[Test prototype 10] starts at [Prototype design]'s end
[Test prototype 20] starts at [Prototype design]'s end
[Test prototype 30] starts at [Prototype design]'s end
[Test prototype 40] starts at [Prototype design]'s end
[Test prototype 50] starts at [Prototype design]'s end
[Test prototype 60] starts at [Prototype design]'s end
[Test prototype 70] starts at [Prototype design]'s end
[Test prototype 80] starts at [Prototype design]'s end
[Test prototype 90] starts at [Prototype design]'s end
[Test prototype 100] starts at [Prototype design]'s end

[Test prototype 0] is 0% complete
[Test prototype 10] is 10% complete
[Test prototype 20] is 20% complete
[Test prototype 30] is 30% complete
[Test prototype 40] is 40% complete
[Test prototype 50] is 50% complete
[Test prototype 60] is 60% complete
[Test prototype 70] is 70% complete
[Test prototype 80] is 80% complete
[Test prototype 90] is 90% complete
[Test prototype 100] is 100% complete
```

@endgantt



[Ref. QA-8297]

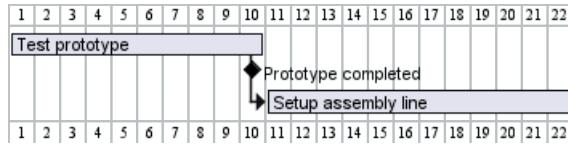
16.7 Milestone

You can define Milestones using the `happen` verb.



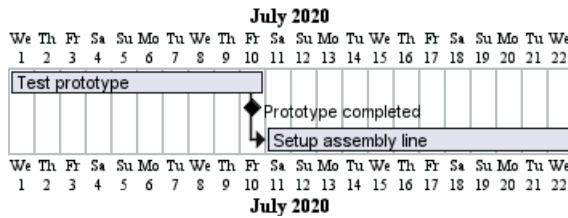
16.7.1 Relative milestone (use of constraints)

```
@startgantt
[Test prototype] lasts 10 days
[Prototype completed] happens at [Test prototype]'s end
[Setup assembly line] lasts 12 days
[Setup assembly line] starts at [Test prototype]'s end
@endgantt
```



16.7.2 Absolute milestone (use of fixed date)

```
@startgantt
Project starts 2020-07-01
[Test prototype] lasts 10 days
[Prototype completed] happens 2020-07-10
[Setup assembly line] lasts 12 days
[Setup assembly line] starts at [Test prototype]'s end
@endgantt
```



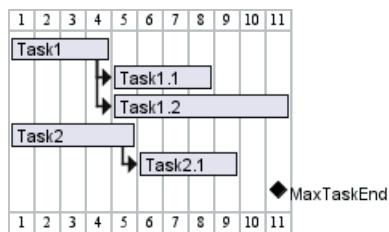
16.7.3 Milestone of maximum end of tasks

```
@startgantt
[Task1] lasts 4 days
then [Task1.1] lasts 4 days
[Task1.2] starts at [Task1]'s end and lasts 7 days

[Task2] lasts 5 days
then [Task2.1] lasts 4 days

[MaxTaskEnd] happens at [Task1.1]'s end
[MaxTaskEnd] happens at [Task1.2]'s end
[MaxTaskEnd] happens at [Task2.1]'s end
```

@endgantt



[Ref. QA-10764]

16.8 Hyperlinks

You can add hyperlinks to tasks.



```
@startgantt
[task1] lasts 10 days
[task1] links to [[http://plantuml.com]]
@endgantt
```



16.9 Calendar

You can specify a starting date for the whole project. By default, the first task starts at this date.

```
@startgantt
Project starts the 20th of september 2017
[Prototype design] as [TASK1] lasts 13 days
[TASK1] is colored in Lavender/LightBlue
@endgantt
```



September 2017 Oct

16.10 Coloring days

It is possible to add colors to some days.

```
@startgantt
Project starts the 2020/09/01
```

2020/09/07 is colored in salmon

2020/09/13 to 2020/09/16 are colored in lightblue

```
[Prototype design] as [TASK1] lasts 22 days
[TASK1] is colored in Lavender/LightBlue
[Prototype completed] happens at [TASK1]'s end
@endgantt
```



September 2020

16.11 Changing scale

You can change scale for very long project, with one of those parameters:

- printscale
- ganttscale
- projectscale

and one of the values:

- daily (*by default*)
- weekly
- monthly



- quarterly
- yearly

(See QA-11272, QA-9041 and QA-10948)

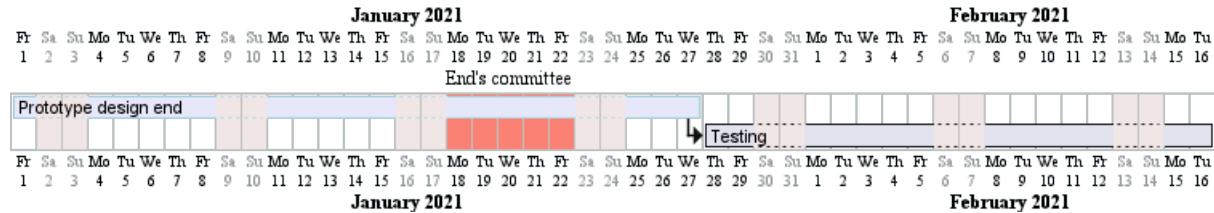
16.11.1 Daily (by default)

```
@startgantt
saturday are closed
sunday are closed
```

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon

```
@endgantt
```



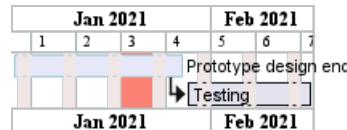
16.11.2 Weekly

```
@startgantt
printscale weekly
saturday are closed
sunday are closed
```

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon

```
@endgantt
```

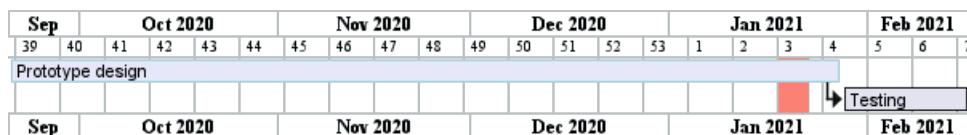


```
@startgantt
printscale weekly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-01-22 are named [End's committee]



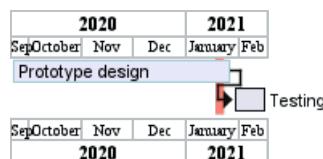
2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.11.3 Monthly

```
@startgantt
projectscale monthly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
```

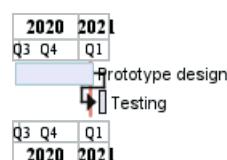
2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.11.4 Quarterly

```
@startgantt
projectscale quarterly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
```

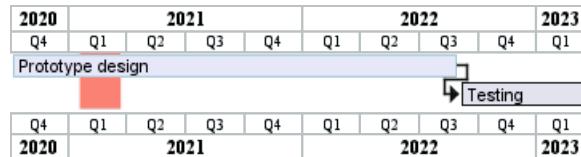
2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



```
@startgantt
projectscale quarterly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-03-22 are colored in salmon
 @endgantt

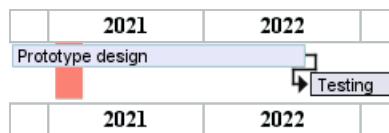




16.11.5 Yearly

```
@startgantt
projectscale yearly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-03-22 are colored in salmon
@endgantt



16.12 Zoom (example for all scale)

You can change zoom, with the parameter:

- `zoom <integer>`

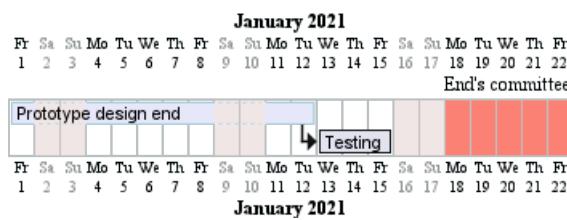
16.12.1 Zoom on weekly scale

16.12.2 Without zoom

```
@startgantt
printscale daily
saturday are closed
sunday are closed
```

Project starts the 1st of january 2021
[Prototype design end] as [TASK1] lasts 8 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 3 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt



16.12.3 With zoom

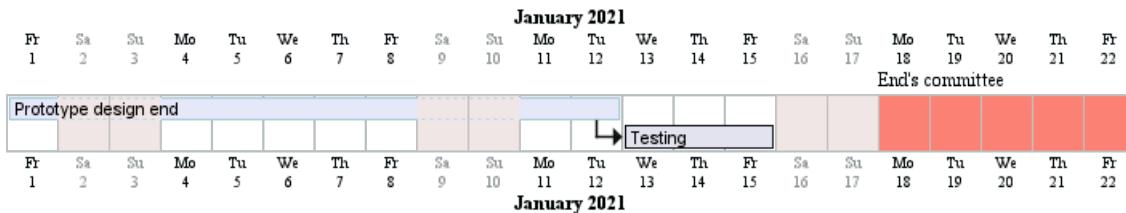
```
@startgantt
printscale daily zoom 2
```



saturday are closed
sunday are closed

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] lasts 8 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 3 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



[Ref. QA-13725]

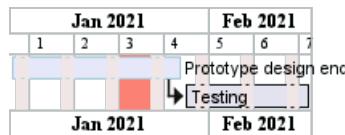
16.12.4 Zoom on weekly scale

16.12.5 Without zoom

@startgantt
 printscale weekly
 saturday are closed
 sunday are closed

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.12.6 With zoom

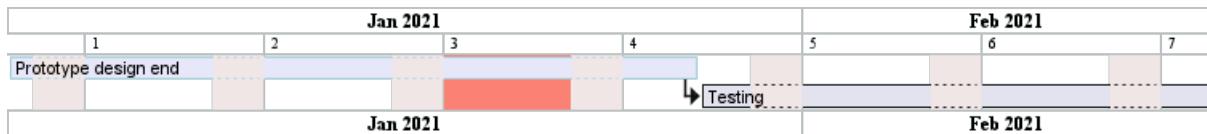
@startgantt
 printscale weekly zoom 4
 saturday are closed
 sunday are closed

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]



2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.12.7 Zoom on monthly scale

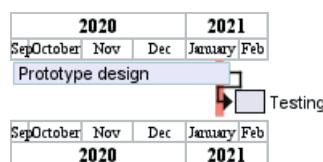
16.12.8 Without zoom

```
@startgantt
projectscale monthly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-01-22 are named [End's committee]

2021-01-18 to 2021-01-22 are colored in salmon

@endgantt



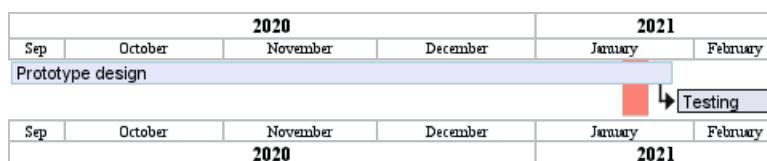
16.12.9 With zoom

```
@startgantt
projectscale monthly zoom 3
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-01-22 are named [End's committee]

2021-01-18 to 2021-01-22 are colored in salmon

@endgantt



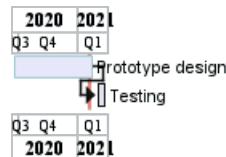
16.12.10 Zoom on quarterly scale

16.12.11 Without zoom

```
@startgantt
projectscale quarterly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
```



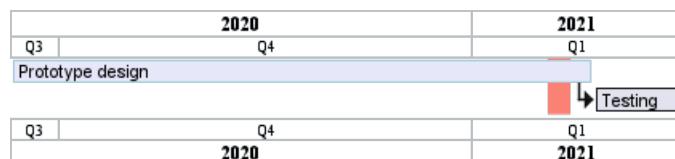
2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.12.12 With zoom

```
@startgantt
projectscale quarterly zoom 7
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.12.13 Zoom on yearly scale

16.12.14 Without zoom

```
@startgantt
projectscale yearly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-03-22 are colored in salmon
 @endgantt

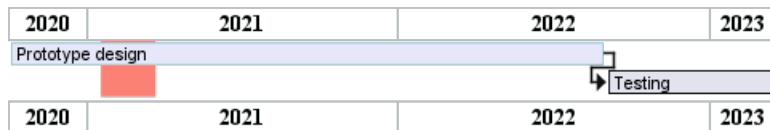


16.12.15 With zoom

```
@startgantt
projectscale yearly zoom 2
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]
```



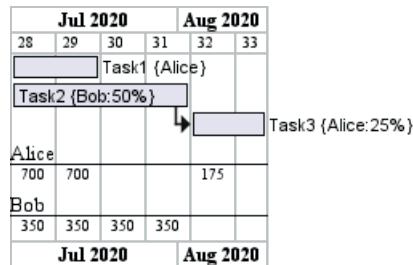
2021-01-18 to 2021-03-22 are colored in salmon
@endgantt



16.13 Weekscale with Weeknumbers or Calendar Date

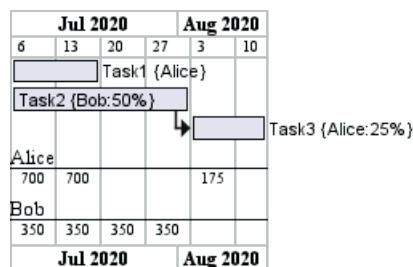
16.13.1 With Weeknumbers (by default)

```
@startgantt
printscale weekly
Project starts the 6th of July 2020
[Task1] on {Alice} lasts 2 weeks
[Task2] on {Bob:50%} lasts 2 weeks
then [Task3] on {Alice:25%} lasts 3 days
@endgantt
```



16.13.2 With Calendar Date

```
@startgantt
printscale weekly with calendar date
Project starts the 6th of July 2020
[Task1] on {Alice} lasts 2 weeks
[Task2] on {Bob:50%} lasts 2 weeks
then [Task3] on {Alice:25%} lasts 3 days
@endgantt
```



[Ref. QA-11630]

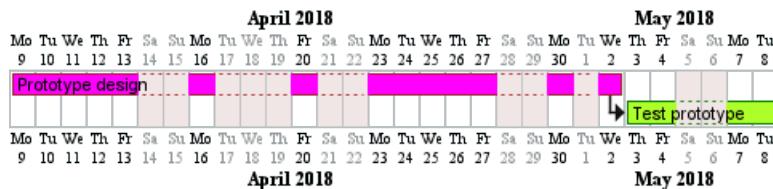
16.14 Close day

It is possible to close some day.

```
@startgantt
project starts the 2018/04/09
saturday are closed
sunday are closed
2018/05/01 is closed
2018/04/17 to 2018/04/19 is closed
[Prototype design] lasts 14 days
```



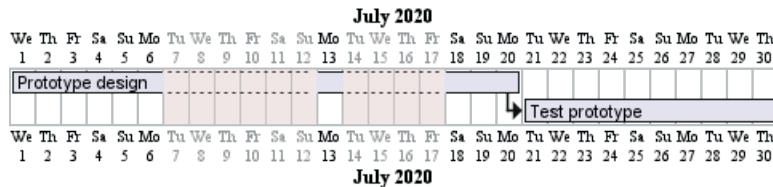
[Test prototype] lasts 4 days
 [Test prototype] starts at [Prototype design]'s end
 [Prototype design] is colored in Fuchsia/FireBrick
 [Test prototype] is colored in GreenYellow/Green
 @endgantt



Then it is possible to open some closed day.

@startgantt
 2020-07-07 to 2020-07-17 is closed
 2020-07-13 is open

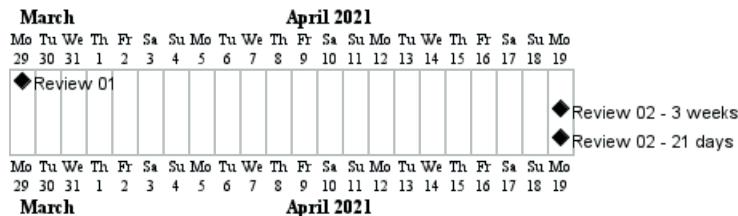
Project starts the 2020-07-01
 [Prototype design] lasts 10 days
 Then [Test prototype] lasts 10 days
 @endgantt



16.15 Definition of a week depending of closed days

A week is a synonym for how many non-closed days are in a week, as:

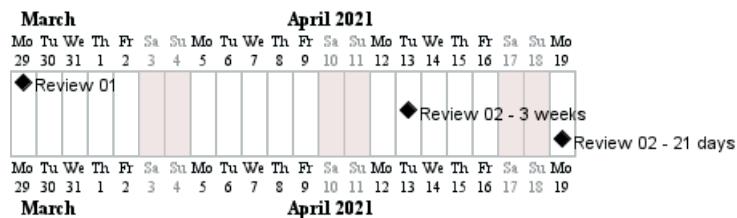
@startgantt
 Project starts 2021-03-29
 [Review 01] happens at 2021-03-29
 [Review 02 - 3 weeks] happens on 3 weeks after [Review 01]'s end
 [Review 02 - 21 days] happens on 21 days after [Review 01]'s end
 @endgantt



So if you specify Saturday and Sunday as closed, a week will be equivalent to 5 days, as:

@startgantt
 Project starts 2021-03-29
 saturday are closed
 sunday are closed
 [Review 01] happens at 2021-03-29
 [Review 02 - 3 weeks] happens on 3 weeks after [Review 01]'s end
 [Review 02 - 21 days] happens on 21 days after [Review 01]'s end
 @endgantt





[Ref. QA-13434]

16.16 Working days

It is possible to manage working days.

@startgantt

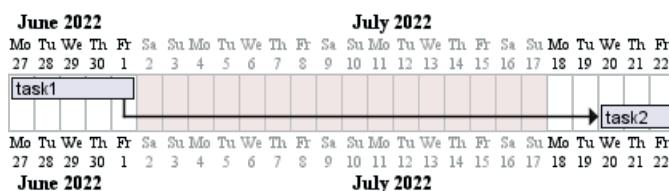
saturday are closed
sunday are closed
2022-07-04 to 2022-07-15 is closed

Project starts 2022-06-27

[task1] starts at 2022-06-27 and lasts 1 week

[task2] starts 2 working days after [task1]'s end and lasts 3 days

@endgantt



[Ref. QA-16188]

16.17 Simplified task succession

It's possible to use the then keyword to denote consecutive tasks.

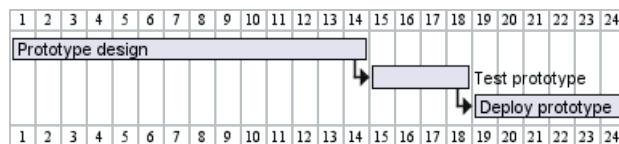
@startgantt

[Prototype design] lasts 14 days

then [Test prototype] lasts 4 days

then [Deploy prototype] lasts 6 days

@endgantt



You can also use arrow ->

@startgantt

[Prototype design] lasts 14 days

[Build prototype] lasts 4 days

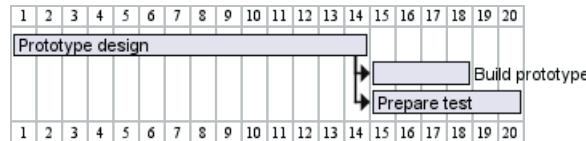
[Prepare test] lasts 6 days

[Prototype design] -> [Build prototype]

[Prototype design] -> [Prepare test]

@endgantt

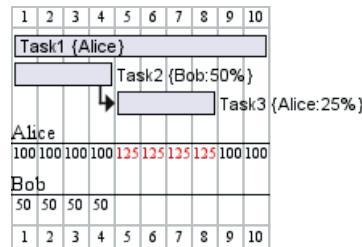




16.18 Working with resources

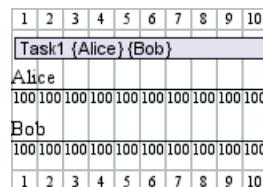
You can affect tasks on resources using the `on` keyword and brackets for resource name.

```
@startgantt
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
@endgantt
```



Multiple resources can be assigned to a task:

```
@startgantt
[Task1] on {Alice} {Bob} lasts 20 days
@endgantt
```



Resources can be marked as off on specific days:

```
@startgantt
project starts on 2020-06-19
[Task1] on {Alice} lasts 10 days
{Alice} is off on 2020-06-24 to 2020-06-26
@endgantt
```

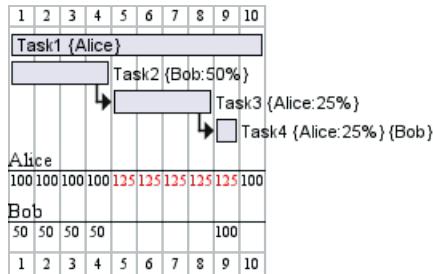


16.19 Hide resources

16.19.1 Without any hiding (by default)

```
@startgantt
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%} {Bob} lasts 1 days
@endgantt
```

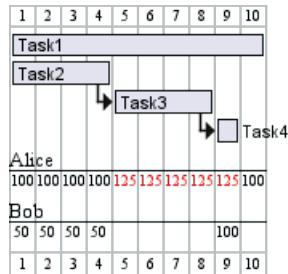




16.19.2 Hide resources names

You can hide resources names and percentage, on tasks, using the `hide resources names` keywords.

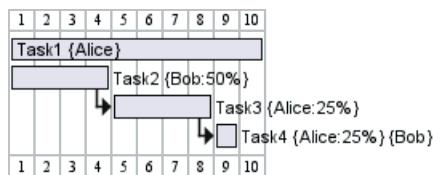
```
@startgantt
hide resources names
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%}{Bob} lasts 1 days
@endgantt
```



16.19.3 Hide resources footbox

You can also hide resources names on bottom of the diagram using the `hide resources footbox` keywords.

```
@startgantt
hide resources footbox
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%}{Bob} lasts 1 days
@endgantt
```



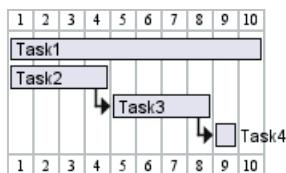
16.19.4 Hide the both (resources names and resources footbox)

You can also hide the both.

```
@startgantt
hide resources names
hide resources footbox
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%}{Bob} lasts 1 days
```



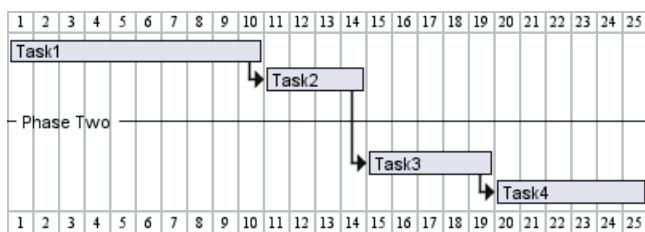
@endgantt



16.20 Horizontal Separator

You can use -- to separate sets of tasks.

```
@startgantt
[Task1] lasts 10 days
then [Task2] lasts 4 days
-- Phase Two --
then [Task3] lasts 5 days
then [Task4] lasts 6 days
@endgantt
```



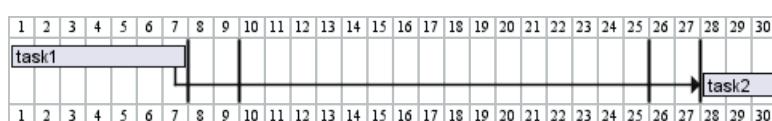
16.21 Vertical Separator

You can add Vertical Separators with the syntax: Separator just [at].

```
@startgantt  
[task1] lasts 1 week  
[task2] starts 20 days after [task1]'s end and lasts 3 days
```

Separator just at [task1]'s end
Separator just 2 days after [task1]'s end

```
Separator just at [task2]'s start  
Separator just 2 days before [task2]'s start  
@endgantt
```



[Ref. QA-16247]

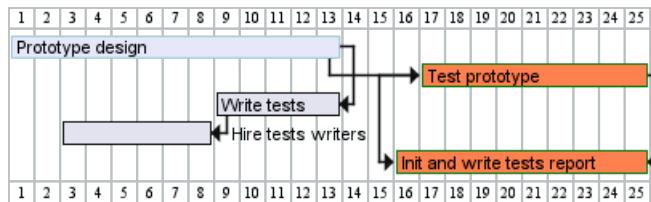
16.22 Complex example

It is also possible to use the **and** conjunction.

You can also add delays in constraints

```
@startgantt
[Prototype design] lasts 13 days and is colored in Lavender/LightBlue
[Test prototype] lasts 9 days and is colored in Coral/Green and starts 3 days after [Prototype design]
[Write tests] lasts 5 days and ends at [Prototype design]'s end
[Hire tests writers] lasts 6 days and ends at [Write tests]'s start
[Init and write tests report] is colored in Coral/Green
```

[Init and write tests report] starts 1 day before [Test prototype]'s start and ends at [Test prototype]'s end
 @endgantt



16.23 Comments

As is mentioned on Common Commands page: blockquote Everything that starts with `simple quote '` is a comment.

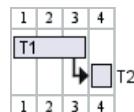
You can also put comments on several lines using `'` to start and `/` to end. blockquote (*i.e.: the first character (except space character) of a comment line must be a simple quote '*)

```
@startgantt
' This is a comment
```

[T1] lasts 3 days

```
/' this comment
is on several lines '/
```

[T2] starts at [T1]'s end and lasts 1 day
 @endgantt

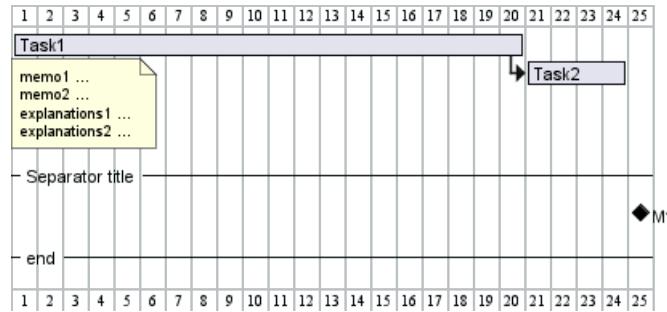


16.24 Using style

16.24.1 Without style (by default)

```
@startgantt
[Task1] lasts 20 days
note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note
[Task2] lasts 4 days
[Task1] -> [Task2]
-- Separator title --
[M1] happens on 5 days after [Task1]'s end
-- end --
@endgantt
```





16.24.2 With style

You can use style to change rendering of elements.

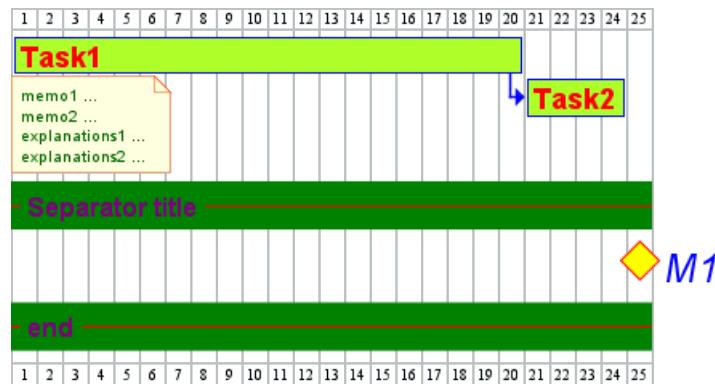
```
@startgantt
<style>
ganttDiagram {
task {
    FontName Helvetica
    FontColor red
    FontSize 18
    FontStyle bold
    BackGroundColor GreenYellow
    LineColor blue
}
milestone {
    FontColor blue
    FontSize 25
    FontStyle italic
    BackGroundColor yellow
    LineColor red
}
note {
    FontColor DarkGreen
    FontSize 10
    LineColor OrangeRed
}
arrow {
    FontName Helvetica
    FontColor red
    FontSize 18
    FontStyle bold
    BackGroundColor GreenYellow
    LineColor blue
}
separator {
    LineColor red
    BackGroundColor green
    FontSize 16
    FontStyle bold
    FontColor purple
}
}
</style>
[Task1] lasts 20 days
note bottom
    memo1 ...
    memo2 ...
```



```

explanations1 ...
explanations2 ...
end note
[Task2] lasts 4 days
[Task1] -> [Task2]
-- Separator title --
[M1] happens on 5 days after [Task1]'s end
-- end --
@endgantt

```



[Ref. QA-10835, QA-12045, QA-11877 and PR-438]

16.24.3 With style (full example)

```

@startgantt
<style>
ganttDiagram {
task {
FontName Helvetica
FontColor red
FontSize 18
FontStyle bold
BackGroundColor GreenYellow
LineColor blue
}
milestone {
FontColor blue
FontSize 25
FontStyle italic
BackGroundColor yellow
LineColor red
}
note {
FontColor DarkGreen
FontSize 10
LineColor OrangeRed
}
arrow {
FontName Helvetica
FontColor red
FontSize 18
FontStyle bold
BackGroundColor GreenYellow
LineColor blue
LineStyle 8.0;13.0
LineThickness 3.0
}

```



```
separator {
    BackgroundColor lightGreen
    LineStyle 8.0;3.0
    LineColor red
    LineThickness 1.0
    FontSize 16
    FontStyle bold
    FontColor purple
    Margin 5
    Padding 20
}
timeline {
    BackgroundColor Bisque
}
closed {
    BackgroundColor pink
    FontColor red
}
}
</style>
Project starts the 2020-12-01

[Task1] lasts 10 days
sunday are closed

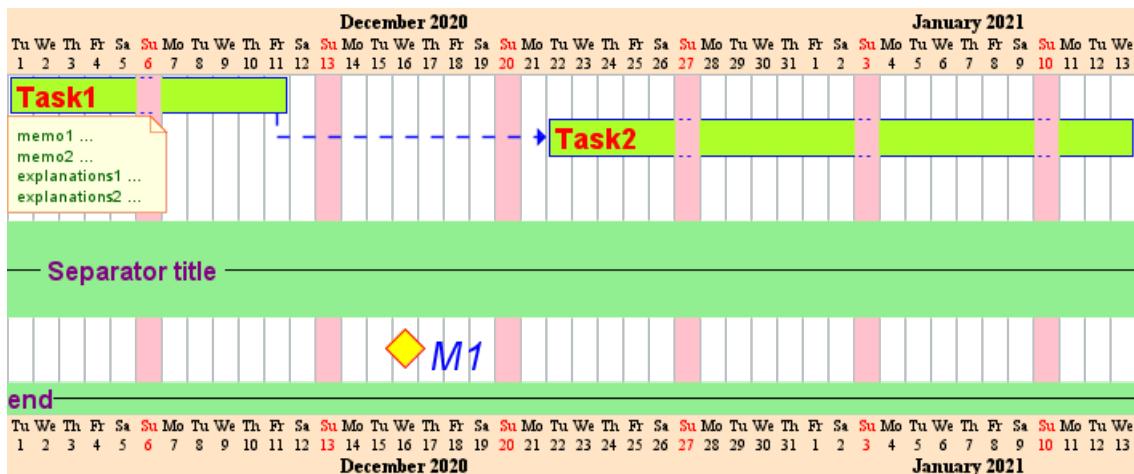
note bottom
    memo1 ...
    memo2 ...
    explanations1 ...
    explanations2 ...
end note

[Task2] lasts 20 days
[Task2] starts 10 days after [Task1]'s end
-- Separator title --
[M1] happens on 5 days after [Task1]'s end

<style>
separator {
    LineColor black
    Margin 0
    Padding 0
}
</style>

-- end --
@endgantt
```





[Ref. QA-13570, QA-13672]

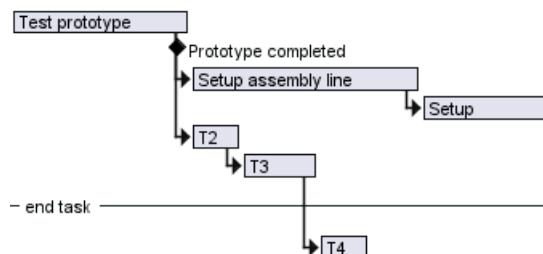
TODO: DONE Thanks for style for Separator and all style for Arrow (thickness...)

16.24.4 Clean style

With style, you can also clean a Gantt diagram (*showing tasks, dependencies and relative durations only - but no actual start date and no actual scale*):

```
@startgantt
<style>
ganttDiagram {
    timeline {
        LineColor transparent
        FontColor transparent
    }
}
</style>

hide footbox
[Test prototype] lasts 7 days
[Prototype completed] happens at [Test prototype]'s end
[Setup assembly line] lasts 9 days
[Setup assembly line] starts at [Test prototype]'s end
then [Setup] lasts 5 days
[T2] lasts 2 days and starts at [Test prototype]'s end
then [T3] lasts 3 days
-- end task --
then [T4] lasts 2 days
@endgantt
```



[Ref. QA-13971]

Or:

```
@startgantt
```



```

<style>
ganttDiagram {
    timeline {
        LineColor transparent
        FontColor transparent
    }
    closed {
        FontColor transparent
    }
}
</style>

hide footbox
project starts the 2018/04/09
saturday are closed
sunday are closed
2018/05/01 is closed
2018/04/17 to 2018/04/19 is closed
[Prototype design] lasts 9 days
[Test prototype] lasts 5 days
[Test prototype] starts at [Prototype design]'s end
[Prototype design] is colored in Fuchsia/FireBrick
[Test prototype] is colored in GreenYellow/Green
@endgantt

```



[Ref. QA-13464]

16.25 Add notes

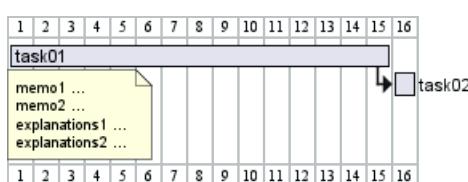
```

@startgantt
[task01] lasts 15 days
note bottom
    memo1 ...
    memo2 ...
    explanations1 ...
    explanations2 ...
end note

[task01] -> [task02]

@endgantt

```



Example with overlap.

```

@startgantt
[task01] lasts 15 days
note bottom
    memo1 ...
    memo2 ...

```



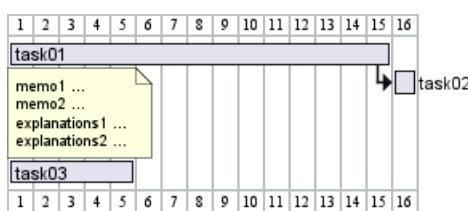
```

explanations1 ...
explanations2 ...
end note

[task01] -> [task02]
[task03] lasts 5 days

@endgantt

```



```
@startgantt
```

```
-- test01 --
```

```

[task01] lasts 4 days
note bottom
'note left
memo1 ...
memo2 ...
explanations1 ...
explanations2 ...
end note

```

```

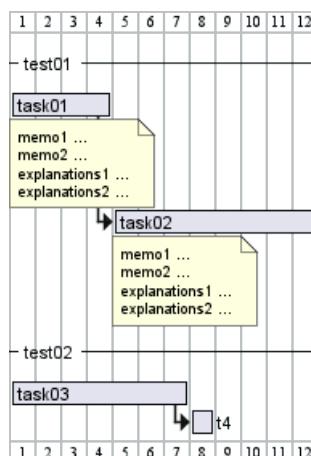
[task02] lasts 8 days
[task01] -> [task02]
note bottom
'note left
memo1 ...
memo2 ...
explanations1 ...
explanations2 ...
end note
-- test02 --

```

```

[task03] as [t3] lasts 7 days
[t3] -> [t4]
@endgantt

```



TODO: DONE *Thanks for correction (of #386 on v1.2020.18) when overlapping*

@startgantt

Project starts 2020-09-01

[taskA] starts 2020-09-01 and lasts 3 days
 [taskB] starts 2020-09-10 and lasts 3 days
 [taskB] displays on same row as [taskA]

[task01] starts 2020-09-05 and lasts 4 days

then [task02] lasts 8 days

note bottom

 note for task02

 more notes

end note

then [task03] lasts 7 days

note bottom

 note for task03

 more notes

end note

-- separator --

[taskC] starts 2020-09-02 and lasts 5 days
 [taskD] starts 2020-09-09 and lasts 5 days
 [taskD] displays on same row as [taskC]

[task 10] starts 2020-09-05 and lasts 5 days

then [task 11] lasts 5 days

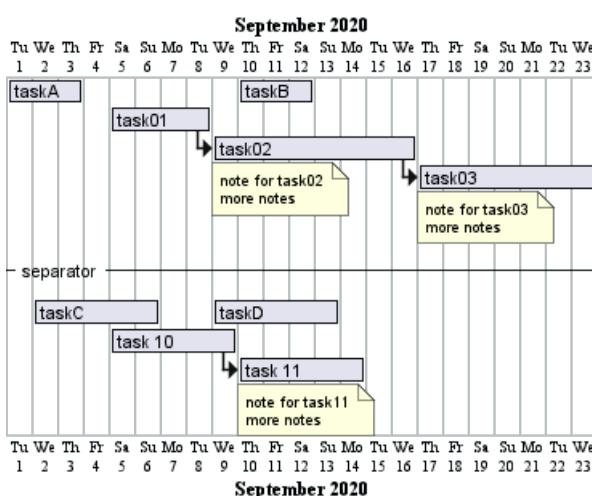
note bottom

 note for task11

 more notes

end note

@endgantt



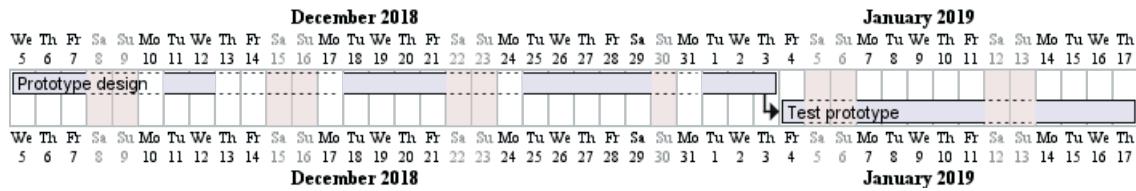
16.26 Pause tasks

@startgantt

Project starts the 5th of december 2018
 saturday are closed



sunday are closed
 2018/12/29 is opened
 [Prototype design] lasts 17 days
 [Prototype design] pauses on 2018/12/13
 [Prototype design] pauses on 2018/12/14
 [Prototype design] pauses on monday
 [Test prototype] starts at [Prototype design]'s end and lasts 2 weeks
 @endgantt

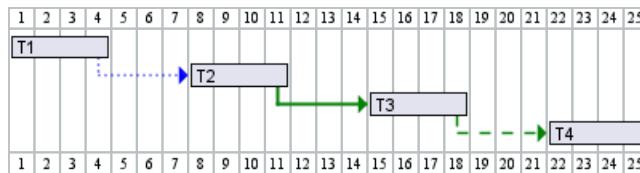


16.27 Change link colors

You can change link colors:

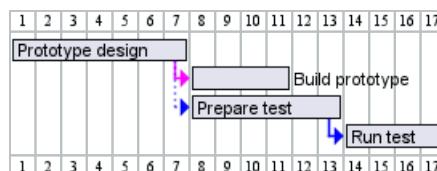
- with this syntax: with <color> <style> link

```
@startgantt
[T1] lasts 4 days
[T2] lasts 4 days and starts 3 days after [T1]'s end with blue dotted link
[T3] lasts 4 days and starts 3 days after [T2]'s end with green bold link
[T4] lasts 4 days and starts 3 days after [T3]'s end with green dashed link
@endgantt
```



- or directly by using arrow style

```
@startgantt
<style>
ganttDiagram {
arrow {
  LineColor blue
}
}
</style>
[Prototype design] lasts 7 days
[Build prototype] lasts 4 days
[Prepare test] lasts 6 days
[Prototype design] -[#FF00FF]-> [Build prototype]
[Prototype design] -[dotted]-> [Prepare test]
Then [Run test] lasts 4 days
@endgantt
```



[Ref. QA-13693]

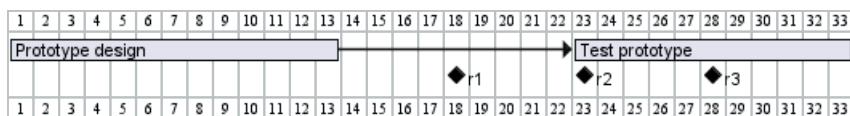


16.28 Tasks or Milestones on the same line

You can put Tasks or Milestones on the same line, with this syntax:

- [T|M] displays on same row as [T|M]

```
@startgantt
[Prototype design] lasts 13 days
[Test prototype] lasts 4 days and 1 week
[Test prototype] starts 1 week and 2 days after [Prototype design]'s end
[Test prototype] displays on same row as [Prototype design]
[r1] happens on 5 days after [Prototype design]'s end
[r2] happens on 5 days after [r1]'s end
[r3] happens on 5 days after [r2]'s end
[r2] displays on same row as [r1]
[r3] displays on same row as [r1]
@endgantt
```

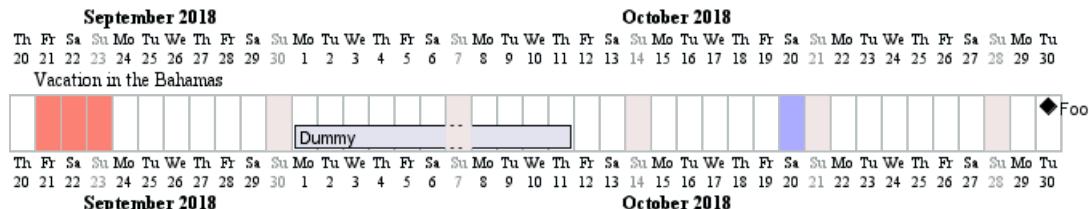


16.29 Highlight today

```
@startgantt
Project starts the 20th of september 2018
sunday are close
2018/09/21 to 2018/09/23 are colored in salmon
2018/09/21 to 2018/09/30 are named [Vacation in the Bahamas]

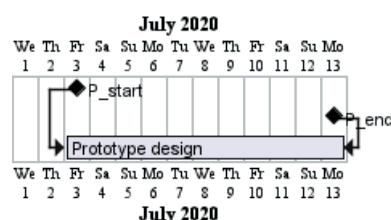
today is 30 days after start and is colored in #AAF
[Foo] happens 40 days after start
[Dummy] lasts 10 days and starts 10 days after start

@endgantt
```



16.30 Task between two milestones

```
@startgantt
project starts on 2020-07-01
[P_start] happens 2020-07-03
[P_end] happens 2020-07-13
[Prototype design] occurs from [P_start] to [P_end]
@endgantt
```



16.31 Grammar and verbal form

Verbal form	Example
[T] starts	
[M] happens	

16.32 Add title, header, footer, caption or legend

```
@startgantt
```

```
header some header
```

```
footer some footer
```

```
title My title
```

```
[Prototype design] lasts 13 days
```

```
legend
```

```
The legend
```

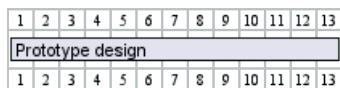
```
end legend
```

```
caption This is caption
```

```
@endgantt
```

some header

My title



The legend

This is caption

some footer

(See also: Common commands)

16.33 Removing Foot Boxes (example for all scale)

You can use the `hide footbox` keywords to remove the foot boxes of the gantt diagram (*as for sequence diagram*).

Examples on:

- daily scale (*without project start*)

```
@startgantt
```

```
hide footbox
```

```
title Foot Box removed
```

```
[Prototype design] lasts 15 days
```

```
[Test prototype] lasts 10 days
```

```
@endgantt
```

Foot Box removed



- daily scale

```
@startgantt
```

Project starts the 20th of september 2017
 [Prototype design] as [TASK1] lasts 13 days
 [TASK1] is colored in Lavender/LightBlue

```
hide footbox
```

```
@endgantt
```



- weekly scale

```
@startgantt
```

```
hide footbox
```

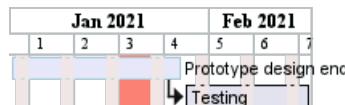
printscale weekly
 saturday are closed
 sunday are closed

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]

2021-01-18 to 2021-01-22 are colored in salmon

```
@endgantt
```



- monthly scale

```
@startgantt
```

```
hide footbox
```

projectscale monthly
 Project starts the 20th of september 2020
 [Prototype design] as [TASK1] lasts 130 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 20 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]

2021-01-18 to 2021-01-22 are colored in salmon

```
@endgantt
```



- quarterly scale

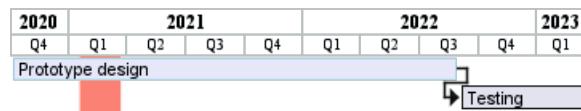
```
@startgantt
```



```
hide footbox
```

```
projectscale quarterly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-03-22 are colored in salmon
@endgantt



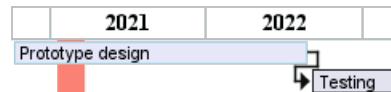
- yearly scale

```
@startgantt
```

```
hide footbox
```

```
projectscale yearly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-03-22 are colored in salmon
@endgantt



16.34 Language of the calendar

You can choose the language of the Gantt calendar, with the `language <xx>` command where `<xx>` is the ISO 639 code of the language.

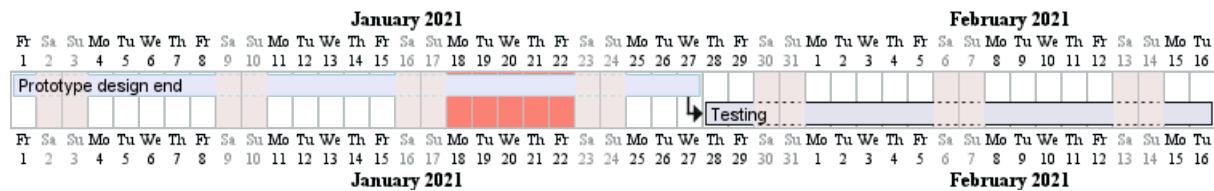
16.34.1 English (*en, by default*)

```
@startgantt
saturday are closed
sunday are closed
```

Project starts 2021-01-01
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
@endgantt



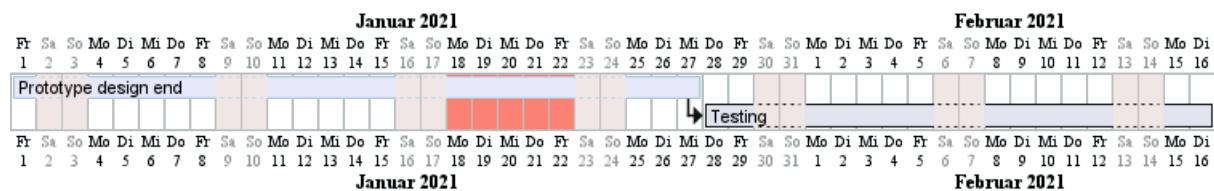


16.34.2 Deutsch (de)

```
@startgantt
language de
saturday are closed
sunday are closed
```

Project starts 2021-01-01
 [Prototype design end] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt

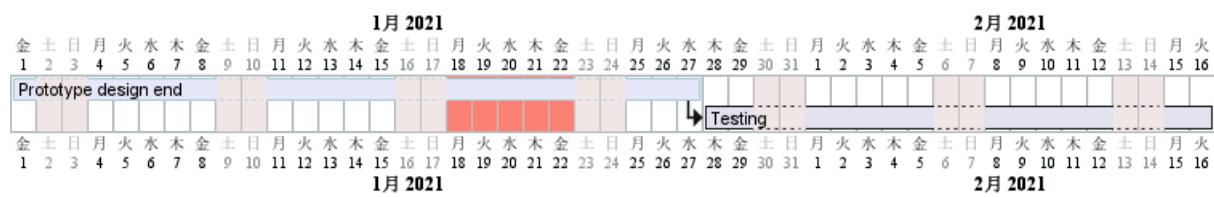


16.34.3 Japanese (ja)

```
@startgantt
language ja
saturday are closed
sunday are closed
```

Project starts 2021-01-01
 [Prototype design end] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.34.4 Chinese (zh)

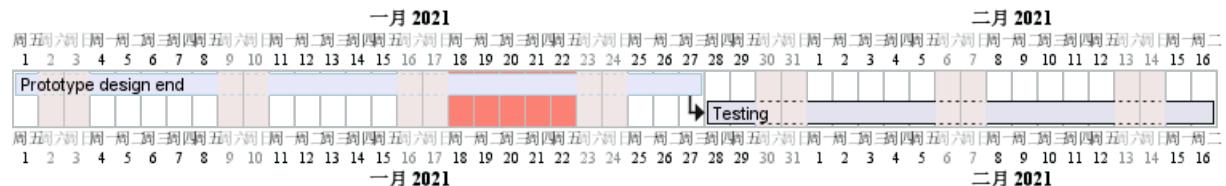
```
@startgantt
language zh
saturday are closed
sunday are closed
```



Project starts 2021-01-01
 [Prototype design end] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon

@endgantt



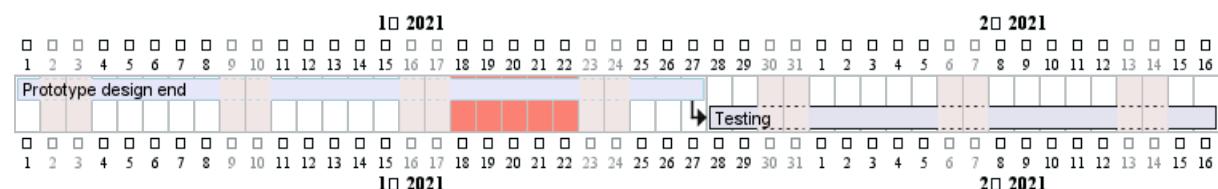
16.34.5 Korean (ko)

```
@startgantt
language ko
saturday are closed
sunday are closed
```

Project starts 2021-01-01
 [Prototype design end] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon

@endgantt



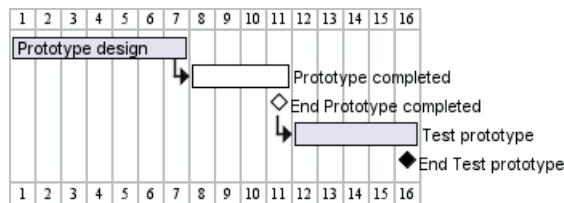
16.35 Delete Tasks or Milestones

You can mark some Tasks or Milestones as `deleted` instead of normally completed to distinguish tasks that may possibly have been discarded, postponed or whatever.

```
@startgantt
[Prototype design] lasts 1 weeks
then [Prototype completed] lasts 4 days
[End Prototype completed] happens at [Prototype completed]'s end
then [Test prototype] lasts 5 days
[End Test prototype] happens at [Test prototype]'s end

[Prototype completed] is deleted
[End Prototype completed] is deleted
@endgantt
```





[Ref. QA-9129]

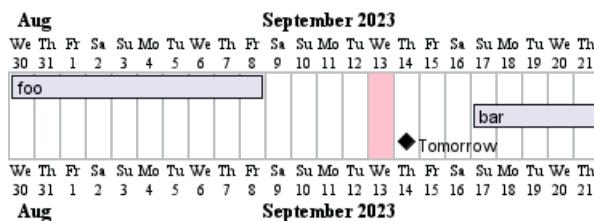
16.36 Start a project, a task or a milestone a number of days before or after today

You can start a project, a task or a milestone a number of days before or after today, using the builtin functions %now and %date:

```

@startgantt
title Today is %date("YYYY-MM-dd")
!$now = %now()
!$past = %date("YYYY-MM-dd", $now - 14*24*3600)
Project starts $past
today is colored in pink
[foo] lasts 10 days
[bar] lasts 5 days and starts %date("YYYY-MM-dd", $now + 4*24*3600)
[Tomorrow] happens %date("YYYY-MM-dd", $now + 1*24*3600)
@endgantt
  
```

Today is 2023-09-13



[Ref. QA-16285]

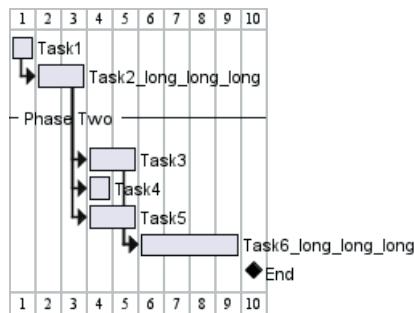
16.37 Change Label position

16.37.1 The labels are near elements (*by default*)

```

@startgantt
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
  
```



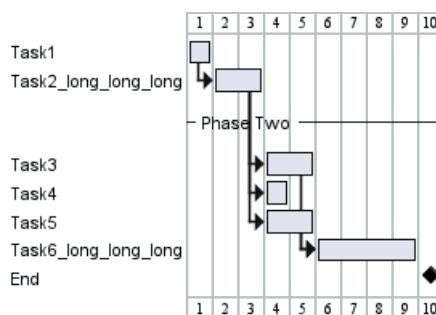


To change the label position, you can use the command `label`:

16.37.2 Label on first column

- Left aligned

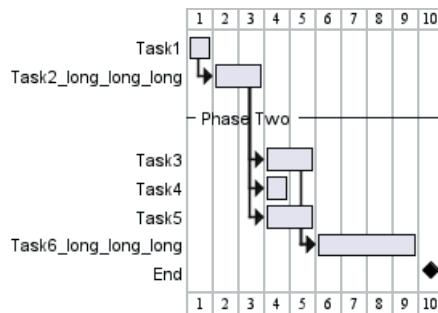
```
@startgantt
Label on first column and left aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```



- Right aligned

```
@startgantt
Label on first column and right aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```

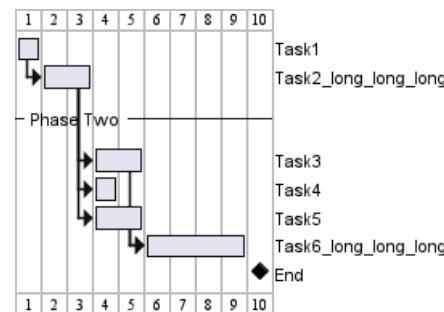




16.37.3 Label on last column

- Left aligned

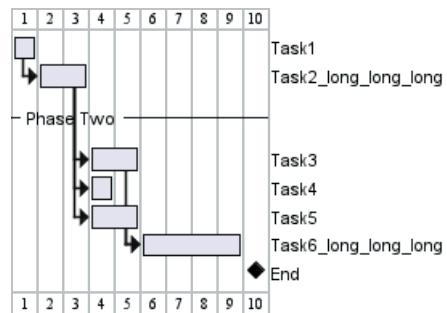
```
@startgantt
Label on last column and left aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```



- Right aligned

```
@startgantt
Label on last column and right aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```





[Ref. QA-12433]



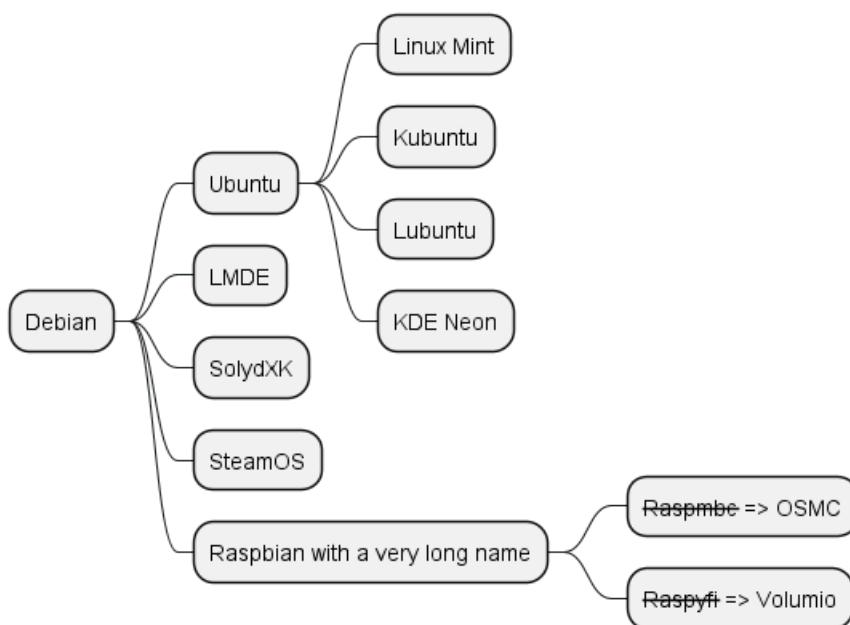
17 MindMap

MindMap diagram are still in beta: the syntax may change without notice.

17.1 Sintaxis OrgMode

Esta sintaxis es compatible con OrgMode

```
@startmindmap
* Debian
** Ubuntu
*** Linux Mint
*** Kubuntu
*** Lubuntu
*** KDE Neon
** LMDE
** SolydXK
** SteamOS
** Raspbian with a very long name
*** <s>Raspmbc</s> => OSMC
*** <s>Raspyfi</s> => Volumio
@endmindmap
```

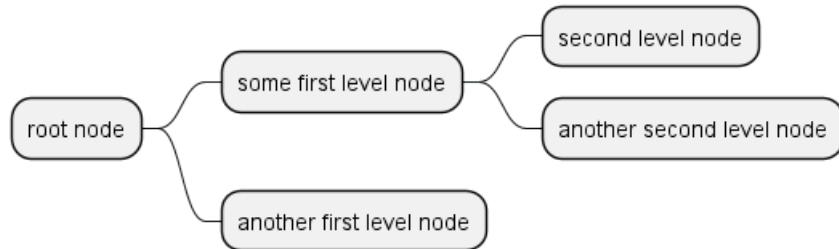


17.2 Markdown syntax

This syntax is compatible with Markdown

```
@startmindmap
* root node
* some first level node
* second level node
* another second level node
* another first level node
@endmindmap
```

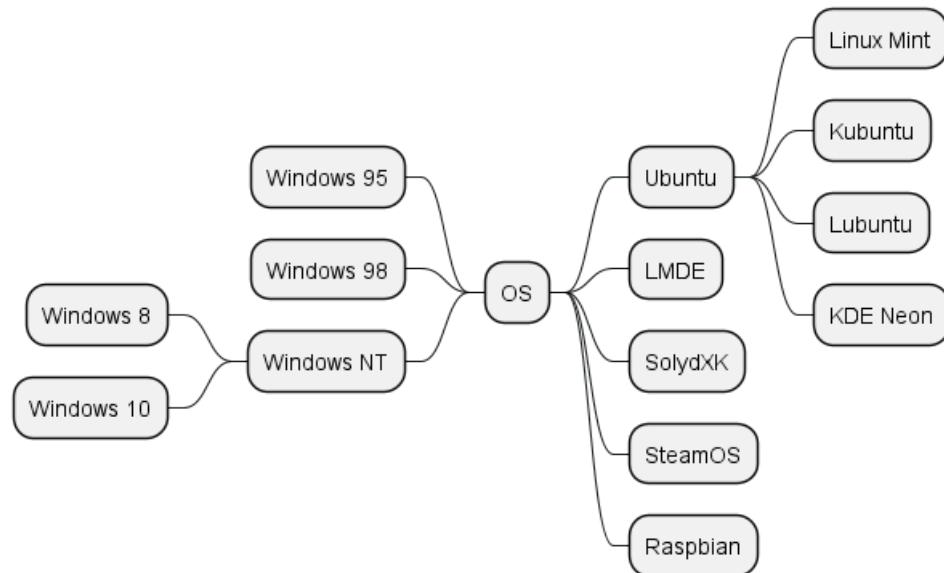




17.3 Notación aritmética

Puede utilizar la siguiente notación para elegir el lado del diagrama

```
@startmindmap
+ OS
++ Ubuntu
+++ Linux Mint
+++ Kubuntu
+++ Lubuntu
+++ KDE Neon
++ LMDE
++ SolydXK
++ SteamOS
++ Raspbian
-- Windows 95
-- Windows 98
-- Windows NT
--- Windows 8
--- Windows 10
@endmindmap
```



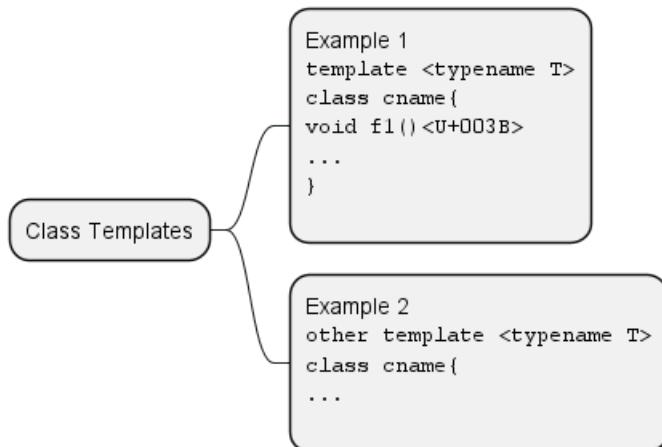
17.4 Multilines

You can use : and ; to have multilines box.

```
@startmindmap
* Class Templates
```



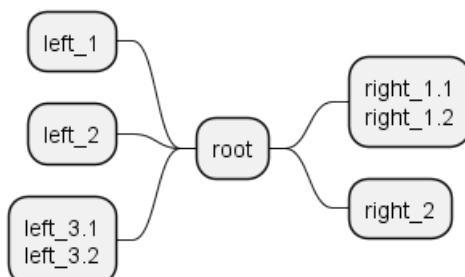
```
**:Example 1
<code>
template <typename T>
class cname{
void f1()<U+003B>
...
}
</code>
;
**:Example 2
<code>
other template <typename T>
class cname{
...
</code>
;
@endmindmap
```



```
@startmindmap
+ root
**:right_1.1
right_1.2;
++ right_2

left side

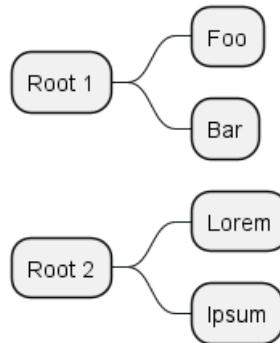
-- left_1
-- left_2
**:left_3.1
left_3.2;
@endmindmap
```



17.5 Multiroot Mindmap

You can create multiroot mindmap, as:

```
@startmindmap
* Root 1
** Foo
** Bar
* Root 2
** Lorem
** Ipsum
@endmindmap
```



[Ref. QH-773]

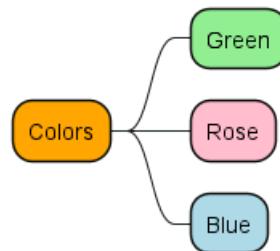
17.6 Colors

It is possible to change node color.

17.6.1 With inline color

- OrgMode syntax mindmap

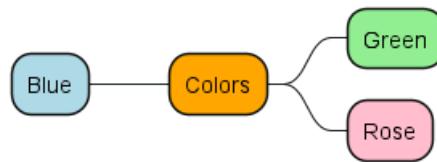
```
@startmindmap
*[#Orange] Colors
**[#lightgreen] Green
**[#FFBBCC] Rose
**[#lightblue] Blue
@endmindmap
```



- Arithmetic notation syntax mindmap

```
@startmindmap
+[#Orange] Colors
++[#lightgreen] Green
++[#FFBBCC] Rose
--[#lightblue] Blue
@endmindmap
```

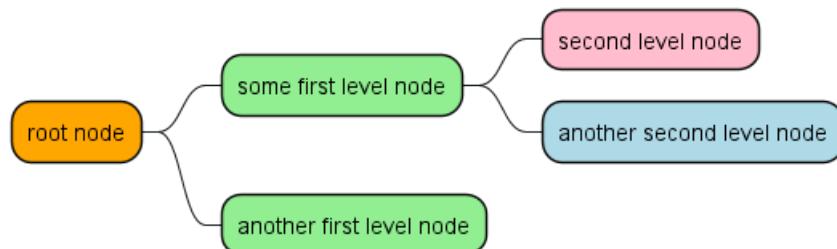




- Markdown syntax mindmap

```

@startmindmap
*[#Orange] root node
 *[#lightgreen] some first level node
  *[#FFBBCC] second level node
  *[#lightblue] another second level node
 *[#lightgreen] another first level node
@endmindmap
  
```



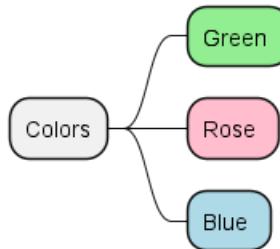
17.6.2 With style color

- OrgMode syntax mindmap

```

@startmindmap
<style>
mindmapDiagram {
    .green {
        BackgroundColor lightgreen
    }
    .rose {
        BackgroundColor #FFBBCC
    }
    .your_style_name {
        BackgroundColor lightblue
    }
}
</style>
* Colors
** Green <<green>>
** Rose <<rose>>
** Blue <<your_style_name>>
@endmindmap
  
```

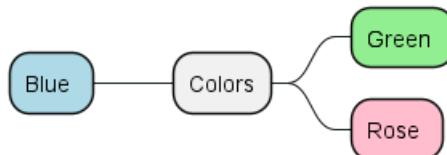




- Arithmetic notation syntax mindmap

```

@startmindmap
<style>
mindmapDiagram {
    .green {
        BackgroundColor lightgreen
    }
    .rose {
        BackgroundColor #FFBBCC
    }
    .your_style_name {
        BackgroundColor lightblue
    }
}
</style>
+ Colors
++ Green <><>
++ Rose <><>
-- Blue <><><><>
@endmindmap
  
```



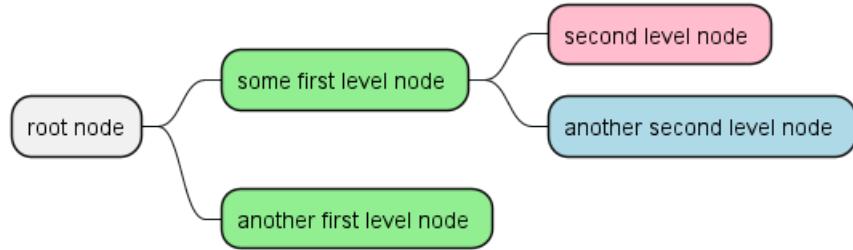
- Markdown syntax mindmap

```

@startmindmap
<style>
mindmapDiagram {
    .green {
        BackgroundColor lightgreen
    }
    .rose {
        BackgroundColor #FFBBCC
    }
    .your_style_name {
        BackgroundColor lightblue
    }
}
</style>
* root node
* some first level node <><>
  * second level node <><>
    * another second level node <><><><>
  
```

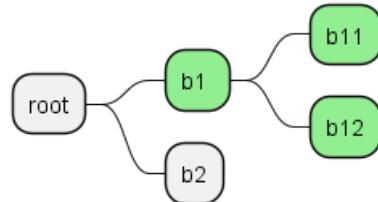


```
* another first level node <<green>>
@endmindmap
```



- Apply style to a branch

```
@startmindmap
<style>
mindmapDiagram {
    .myStyle * {
        backgroundColor lightgreen
    }
}
</style>
+ root
++ b1 <<myStyle>>
+++ b11
+++ b12
++ b2
@endmindmap
```



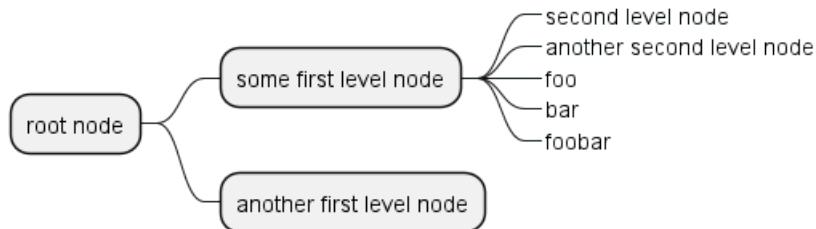
[Ref. GA-920]

17.7 Removing box

You can remove the box drawing using an underscore.

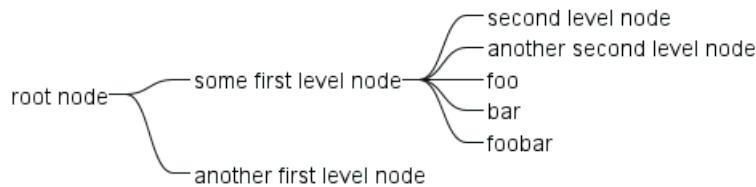
```
@startmindmap
* root node
** some first level node
***__ second level node
***__ another second level node
***__ foo
***__ bar
***__ foobar
** another first level node
@endmindmap
```





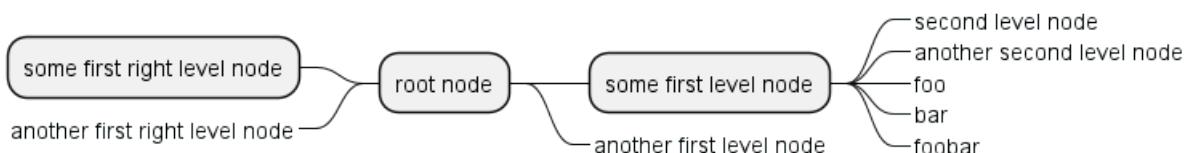
```

@startmindmap
*_ root node
**_ some first level node
***_ second level node
***_ another second level node
***_ foo
***_ bar
***_ foobar
**_ another first level node
@endmindmap
  
```



```

@startmindmap
+ root node
++ some first level node
+++ second level node
+++ another second level node
+++ foo
+++ bar
+++ foobar
++ another first level node
-- some first right level node
-- another first right level node
@endmindmap
  
```



17.8 Changing diagram direction

It is possible to use both sides of the diagram.

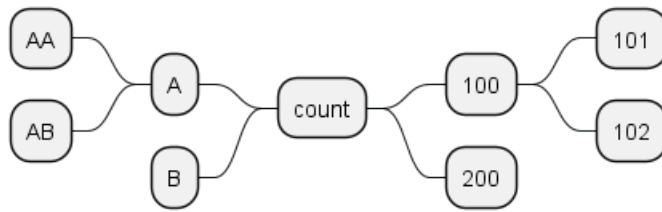
```

@startmindmap
* count
** 100
*** 101
*** 102
** 200
  
```



```
left side
```

```
** A
*** AA
*** AB
** B
@endmindmap
```



17.9 Complete example

```
@startmindmap
caption figure 1
title My super title

* <&flag>Debian
** <&globe>Ubuntu
*** Linux Mint
*** Kubuntu
*** Lubuntu
*** KDE Neon
** <&graph>LMDE
** <&pulse>SolydXK
** <&people>SteamOS
** <&star>Raspbian with a very long name
*** <s>Raspmbc</s> => OSMC
*** <s>Raspyfi</s> => Volumio
```

```
header
My super header
endheader
```

```
center footer My super footer
```

```
legend right
Short
legend
endlegend
@endmindmap
```



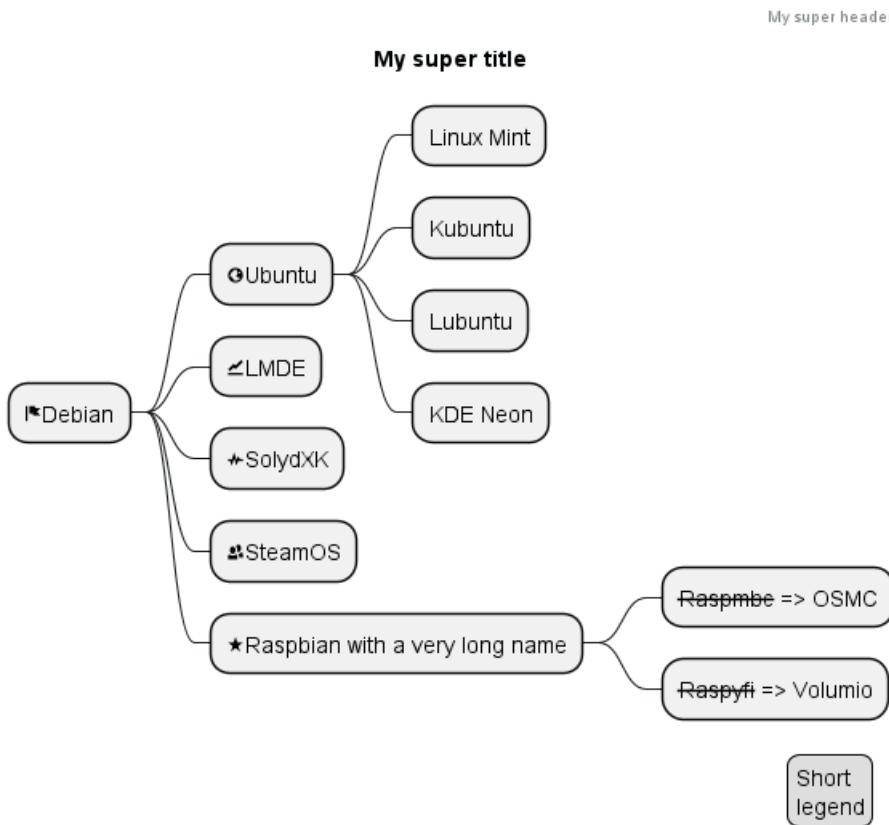


figure 1

17.10 Changing style

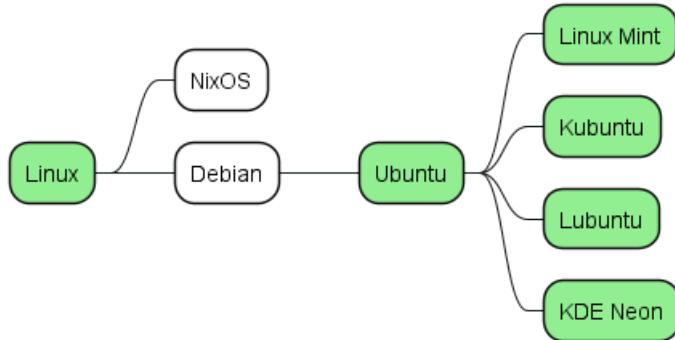
17.10.1 node, depth

```

@startmindmap
<style>
mindmapDiagram {
    node {
        backgroundColor lightGreen
    }
    :depth(1) {
        BackGroundColor white
    }
}
</style>
* Linux
** NixOS
** Debian
*** Ubuntu
**** Linux Mint
**** Kubuntu
**** Lubuntu
**** KDE Neon
@endmindmap

```

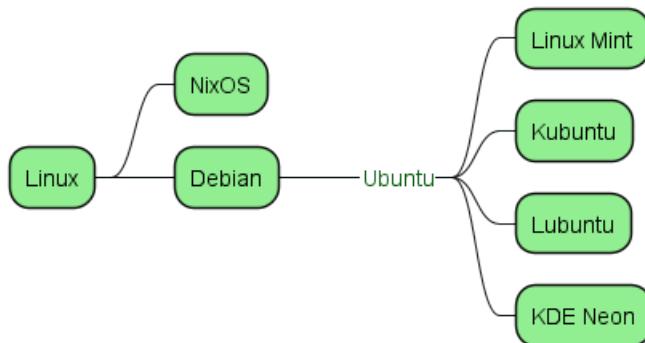




17.10.2 boxless

```

@startmindmap
<style>
mindmapDiagram {
    node {
        BackgroundColor lightGreen
    }
    boxless {
        FontColor darkgreen
    }
}
</style>
* Linux
** NixOS
** Debian
***_ Ubuntu
**** Linux Mint
**** Kubuntu
**** Lubuntu
**** KDE Neon
@endmindmap
  
```



17.11 Word Wrap

Using `MaximumWidth` setting you can control automatic word wrap. Unit used is pixel.

```
@startmindmap
```

```
<style>
node {
```



```

Padding 12
Margin 3
HorizontalAlignment center
LineColor blue
LineThickness 3.0
BackgroundColor gold
RoundCorner 40
MaximumWidth 100
}

rootNode {
    LineStyle 8.0;3.0
    LineColor red
    BackgroundColor white
    LineThickness 1.0
    RoundCorner 0
    Shadowing 0.0
}

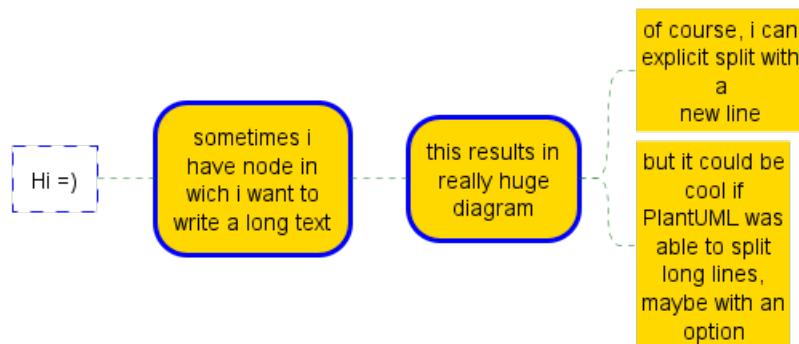
leafNode {
    LineColor gold
    RoundCorner 0
    Padding 3
}

arrow {
    LineStyle 4
    LineThickness 0.5
    LineColor green
}
</style>

* Hi =)
** sometimes i have node in which i want to write a long text
*** this results in really huge diagram
**** of course, i can explicit split with a\nnew line
**** but it could be cool if PlantUML was able to split long lines, maybe with an option

@endmindmap

```



17.12 Creole on Mindmap diagram

You can use Creole or HTML Creole on Mindmap:

```
@startmindmap
* Creole on Mindmap
```



```

left side
**==Creole
This is **bold**
This is //italics//
This is ""monospaced"""
This is --stricken-out--
This is __underlined__
This is ~~wave-underlined~~
--test Unicode and icons--
This is <U+221E> long
This is a <&code> icon
Use image : <img: http://plantuml.com/logo3.png>
;

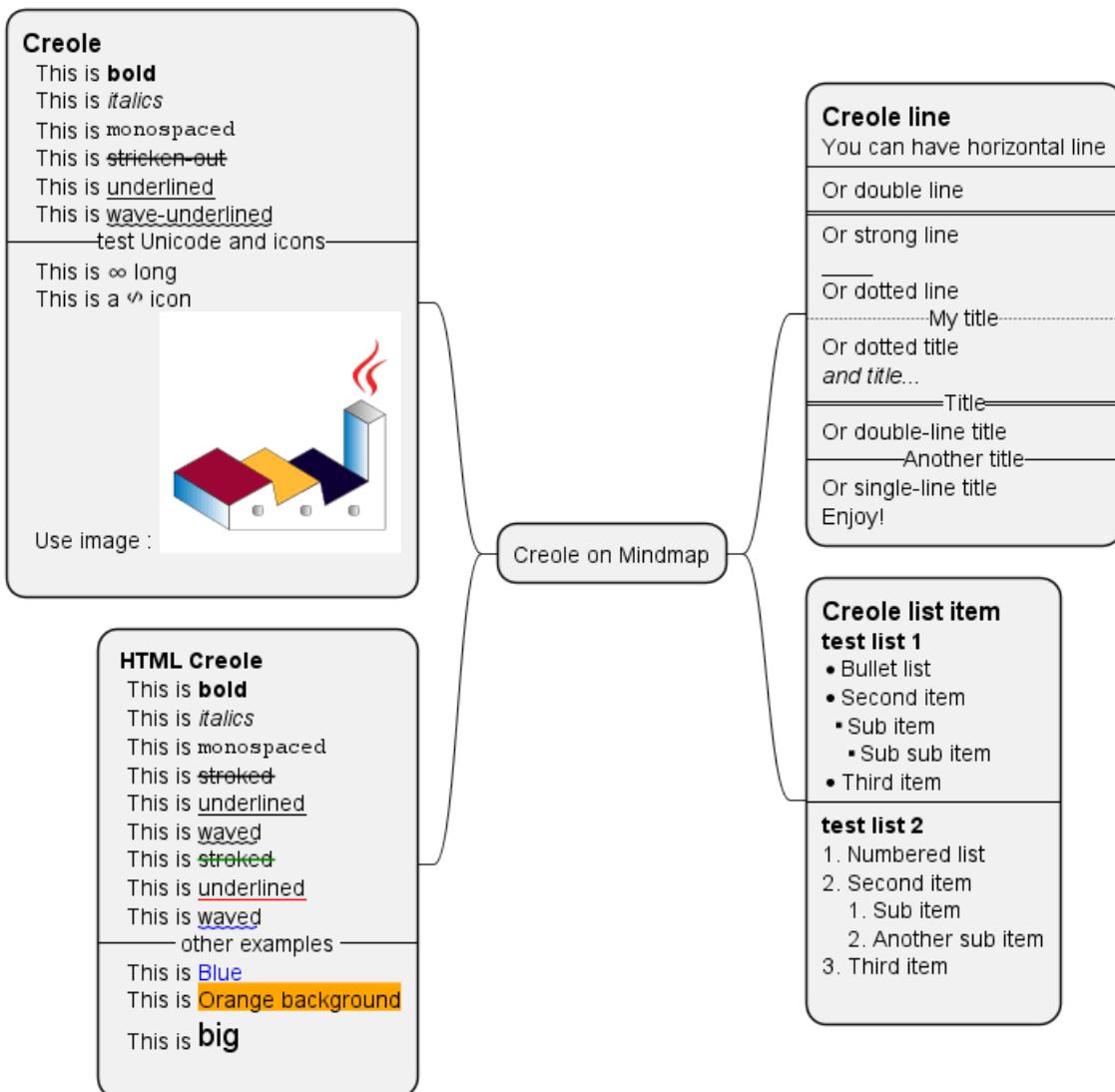
**: <b>HTML Creole
This is <b>bold</b>
This is <i>italics</i>
This is <font:monospaced>monospaced</font>
This is <s>stroked</s>
This is <u>underlined</u>
This is <w>waved</w>
This is <s:green>stroked</s>
This is <u:red>underlined</u>
This is <w:#0000FF>waved</w>
-- other examples --
This is <color:blue>Blue</color>
This is <back:orange>Orange background</back>
This is <size:20>big</size>
;
right side
**==Creole line
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;

**==Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item

```



```
# Third item
;
@endmindmap
```



[Ref. QA-17838]



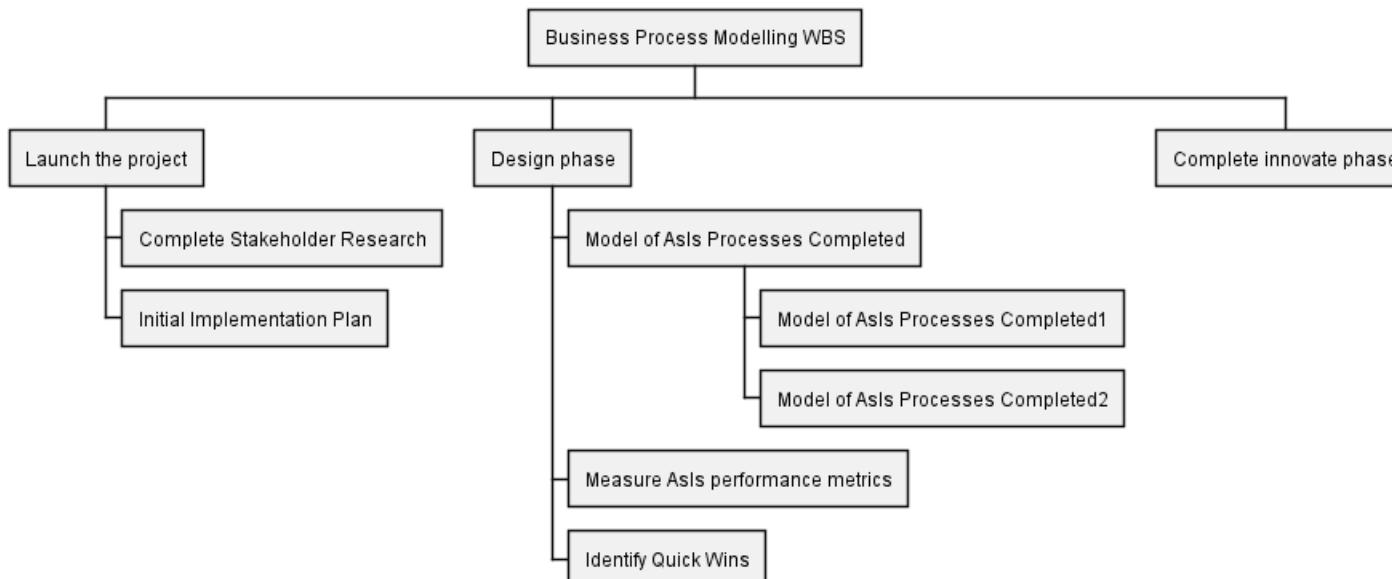
18 Work Breakdown Structure (WBS)

WBS diagram are still in beta: the syntax may change without notice.

18.1 OrgMode syntax

This syntax is compatible with OrgMode

```
@startwbs
* Business Process Modelling WBS
** Launch the project
*** Complete Stakeholder Research
**** Initial Implementation Plan
** Design phase
**** Model of AsIs Processes Completed
***** Model of AsIs Processes Completed1
***** Model of AsIs Processes Completed2
**** Measure AsIs performance metrics
**** Identify Quick Wins
** Complete innovate phase
@endwbs
```

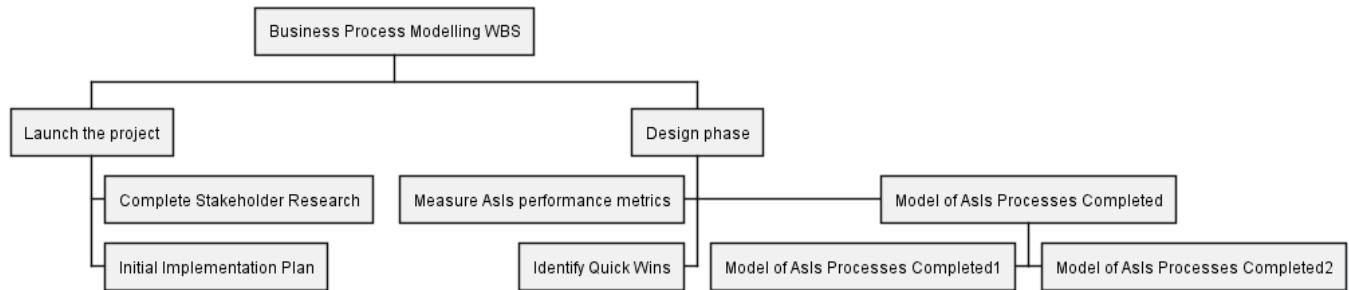


18.2 Change direction

You can change direction using < and >

```
@startwbs
* Business Process Modelling WBS
** Launch the project
*** Complete Stakeholder Research
**** Initial Implementation Plan
** Design phase
**** Model of AsIs Processes Completed
*****< Model of AsIs Processes Completed1
*****> Model of AsIs Processes Completed2
****< Measure AsIs performance metrics
****< Identify Quick Wins
@endwbs
```



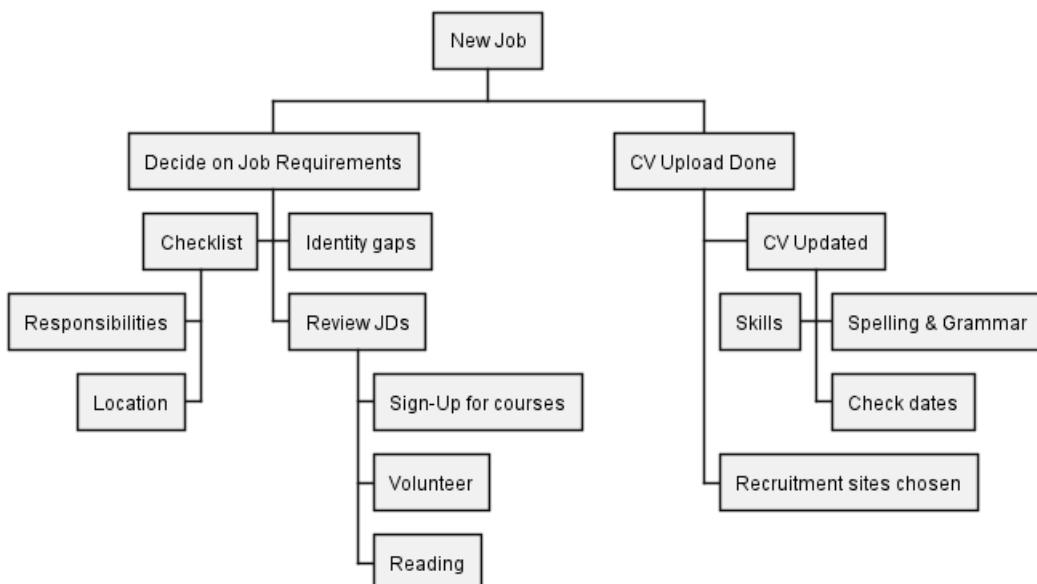


18.3 Arithmetic notation

You can use the following notation to choose diagram side.

```

@startwbs
+ New Job
++ Decide on Job Requirements
+++ Identity gaps
+++ Review JDs
++++ Sign-Up for courses
++++ Volunteer
++++ Reading
+++ Checklist
+++ Responsibilities
+++ Location
++ CV Upload Done
+++ CV Updated
++++ Spelling & Grammar
++++ Check dates
---- Skills
+++ Recruitment sites chosen
@endwbs
  
```



18.4 Multilines

You can use : and ; to have multilines box, as on MindMap.

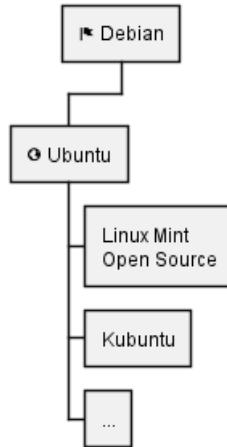
```
@startwbs
```



```
* <&flag> Debian
** <&globe> Ubuntu

***:Linux Mint
Open Source;

*** Kubuntu
*** ...
@endwbs
```



[Ref. QA-13945]

18.5 Removing box

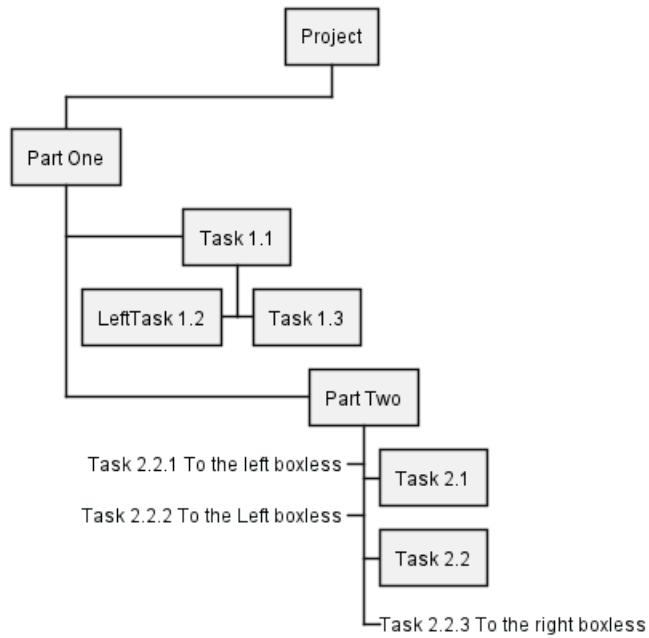
You can use underscore _ to remove box drawing.

18.5.1 Boxless on Arithmetic notation

18.5.2 Several boxless node

```
@startwbs
+ Project
+ Part One
+ Task 1.1
- LeftTask 1.2
+ Task 1.3
+ Part Two
+ Task 2.1
+ Task 2.2
-_ Task 2.2.1 To the left boxless
-_ Task 2.2.2 To the Left boxless
+_ Task 2.2.3 To the right boxless
@endwbs
```

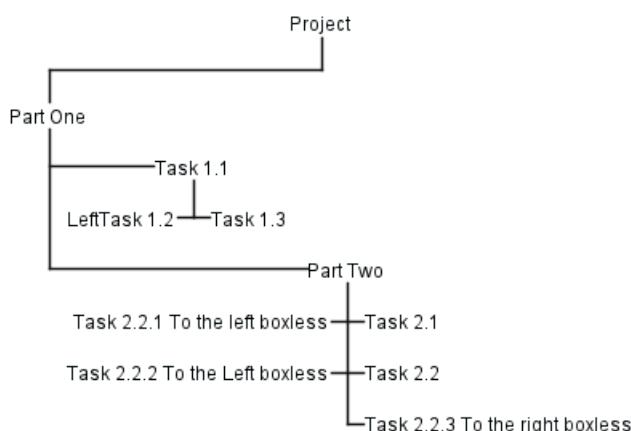




18.5.3 All boxless node

```

@startwbs
+_ Project
+_ Part One
+_ Task 1.1
-_ LeftTask 1.2
+_ Task 1.3
+_ Part Two
+_ Task 2.1
+_ Task 2.2
-_ Task 2.2.1 To the left boxless
-_ Task 2.2.2 To the Left boxless
+_ Task 2.2.3 To the right boxless
@endwbs
  
```



18.5.4 Boxless on OrgMode syntax

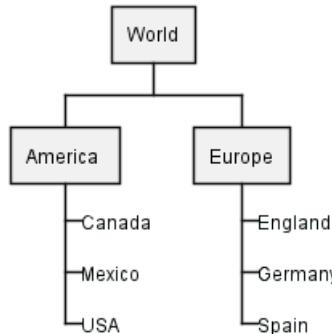
18.5.5 Several boxless node

```

@startwbs
* World
** America
  
```



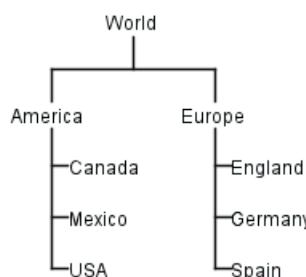
```
***_ Canada
***_ Mexico
***_ USA
** Europe
***_ England
***_ Germany
***_ Spain
@endwbs
```



[Ref. QA-13297]

18.5.6 All boxless node

```
@startwbs
*_ World
**_ America
***_ Canada
***_ Mexico
***_ USA
**_ Europe
***_ England
***_ Germany
***_ Spain
@endwbs
```



[Ref. QA-13355]

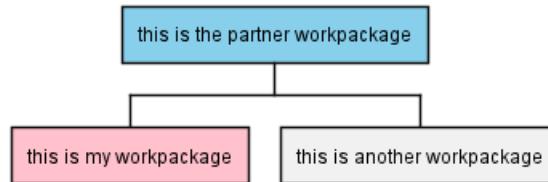
18.6 Colors (with inline or style color)

It is possible to change node color:

- with inline color

```
@startwbs
*[#SkyBlue] this is the partner workpackage
**[#pink] this is my workpackage
** this is another workpackage
@endwbs
```

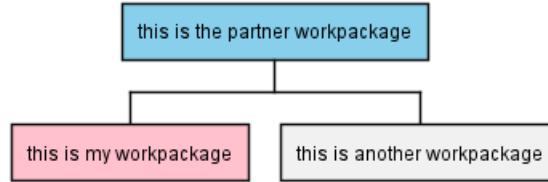




```

@startwbs
+[#SkyBlue] this is the partner workpackage
++[#pink] this is my workpackage
++ this is another workpackage
@endwbs

```



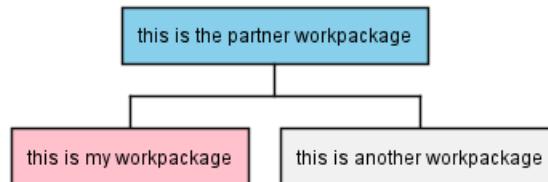
[Ref. QA-12374, only from v1.2020.20]

- with style color

```

@startwbs
<style>
wbsDiagram {
    .pink {
        BackgroundColor pink
    }
    .your_style_name {
        BackgroundColor SkyBlue
    }
}
</style>
* this is the partner workpackage <<your_style_name>>
** this is my workpackage <<pink>>
** this is another workpackage
@endwbs

```



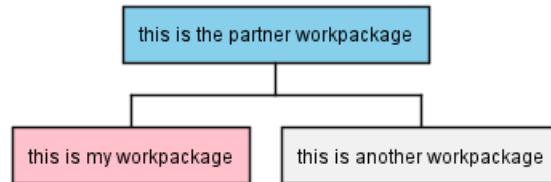
```

@startwbs
<style>
wbsDiagram {
    .pink {
        BackgroundColor pink
    }
    .your_style_name {
        BackgroundColor SkyBlue
    }
}
</style>
+ this is the partner workpackage <<your_style_name>>
++ this is my workpackage <<pink>>

```



```
++ this is another workpackage
@endwbs
```



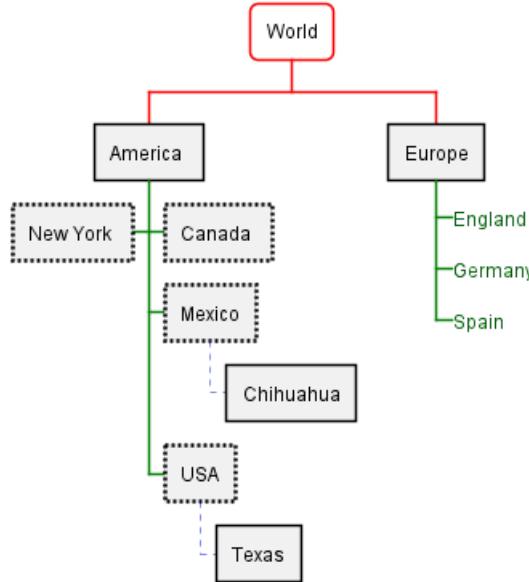
18.7 Using style

It is possible to change diagram style.

```
@startwbs
<style>
wbsDiagram {
    // all lines (meaning connector and borders, there are no other lines in WBS) are black by default
    Linecolor black
    arrow {
        // note that connector are actually "arrow" even if they don't look like as arrow
        // This is to be consistent with other UML diagrams. Not 100% sure that it's a good idea
        // So now connector are green
        LineColor green
    }
    :depth(0) {
        // will target root node
        BackgroundColor White
        RoundCorner 10
        LineColor red
        // Because we are targetting depth(0) for everything, border and connector for level 0 will be
    }
    arrow {
        :depth(2) {
            // Targetting only connector between Mexico-Chihuahua and USA-Texas
            LineColor blue
            LineStyle 4
            LineThickness .5
        }
    }
    node {
        :depth(2) {
            LineStyle 2
            LineThickness 2.5
        }
    }
    boxless {
        // will target boxless node with '_'
        FontColor darkgreen
    }
}
</style>
* World
** America
*** Canada
*** Mexico
**** Chihuahua
*** USA
**** Texas
```



```
***< New York
** Europe
***_ England
***_ Germany
***_ Spain
@endwbs
```



18.8 Word Wrap

Using `MaximumWidth` setting you can control automatic word wrap. Unit used is pixel.

```
@startwbs
```

```
<style>
node {
    Padding 12
    Margin 3
    HorizontalAlignment center
    LineColor blue
    LineThickness 3.0
    BackgroundColor gold
    RoundCorner 40
    MaximumWidth 100
}

rootNode {
    LineStyle 8.0;3.0
    LineColor red
    BackgroundColor white
    LineThickness 1.0
    RoundCorner 0
    Shadowing 0.0
}

leafNode {
    LineColor gold
    RoundCorner 0
    Padding 3
```

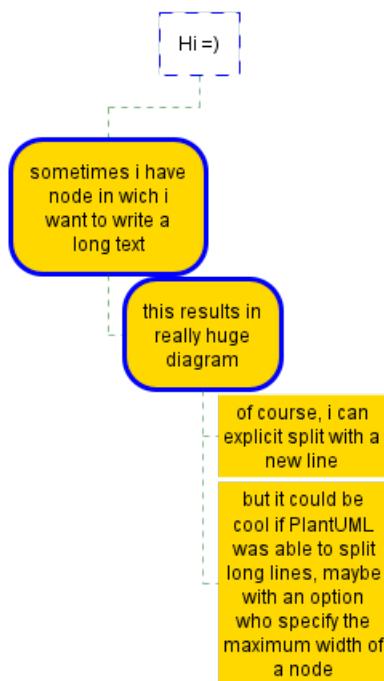


```
}
```

```
arrow {
    LineStyle 4
    LineThickness 0.5
    LineColor green
}
</style>

* Hi =)
** sometimes i have node in which i want to write a long text
*** this results in really huge diagram
**** of course, i can explicit split with a\nnew line
**** but it could be cool if PlantUML was able to split long lines, maybe with an option who specify

@endwbs
```



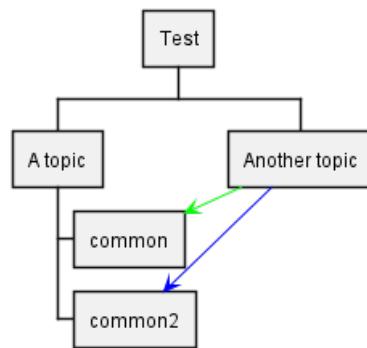
18.9 Add arrows between WBS elements

You can add arrows between WBS elements.

Using alias with as:

```
@startwbs
<style>
.foo {
    LineColor #00FF00;
}
</style>
* Test
** A topic
*** "common" as c1
*** "common2" as c2
** "Another topic" as t2
t2 -> c1 <<foo>>
t2 ..> c2 #blue
@endwbs
```

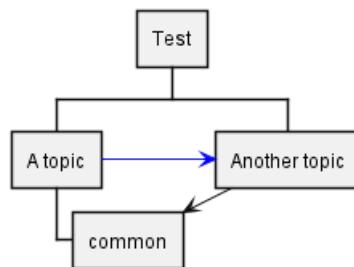




Using alias in parentheses:

```

@startwbs
* Test
**(b) A topic
***(c1) common
**(t2) Another topic
t2 --> c1
b -> t2 #blue
@endwbs
  
```



[Ref. QA-16251]

18.10 Creole on WBS diagram

You can use Creole or HTML Creole on WBS:

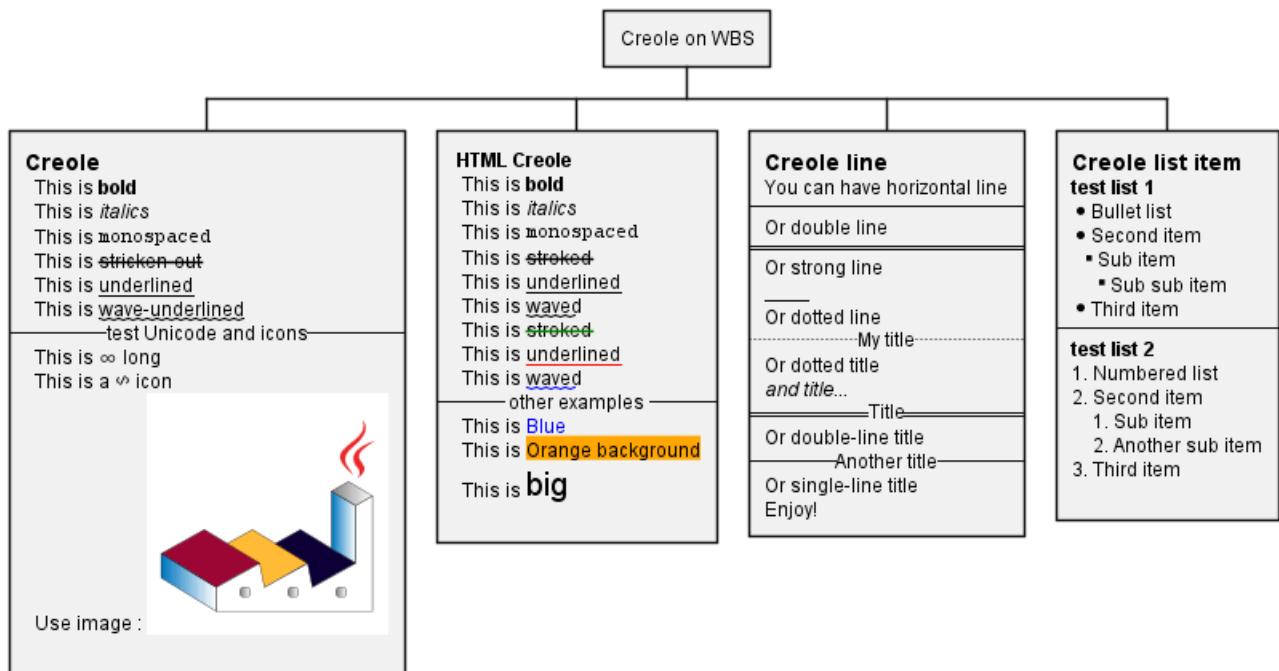
```

@startwbs
* Creole on WBS
**==Creole
  This is **bold**
  This is //italics//
  This is ""monospaced"""
  This is --stricken-out--
  This is __underlined__
  This is ~~wave-underlined~~
--test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
  Use image : <img: http://plantuml.com/logo3.png>
;
**: <b>HTML Creole
  This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
  This is <s>stroked</s>
  This is <u>underlined</u>
  This is <w>waved</w>
  
```



```
This is <s:green>stroked</s>
This is <u:red>underlined</u>
This is <w:#0000FF>waved</w>
-- other examples --
This is <color:blue>Blue</color>
This is <back:orange>Orange background</back>
This is <size:20>big</size>
;
**==Creole line
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;
**==Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
;
@endwbs
```

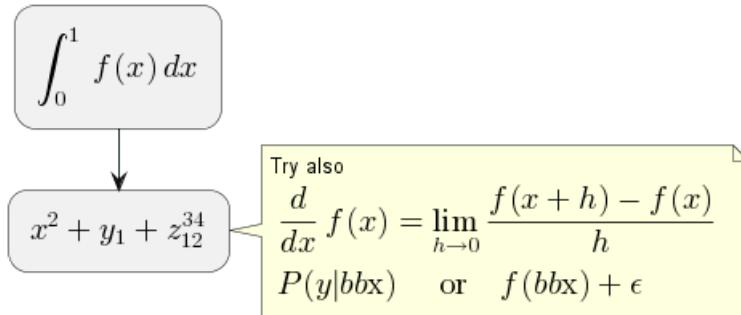




19 Maths

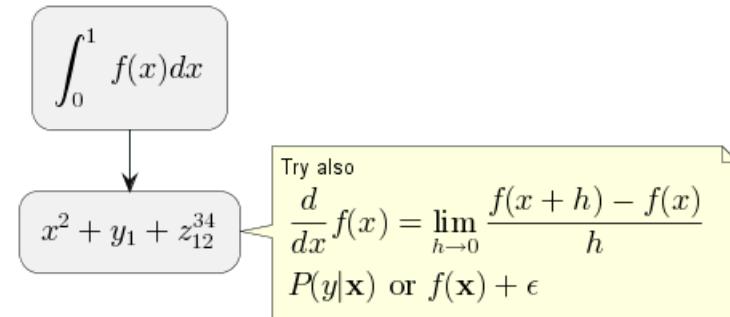
Within PlantUML, you can use AsciiMath notation:

```
@startuml
: $\int_0^1 f(x) dx$ ;
: $x^2 + y_1 + z_{12}^{34}$ ;
note right
Try also
 $d/dx f(x) = \lim_{h \rightarrow 0} (f(x+h) - f(x))/h$ 
 $P(y|bbx)$  or  $f(bbx) + \epsilon$ 
end note
@enduml
```



or JLaTeXMath notation:

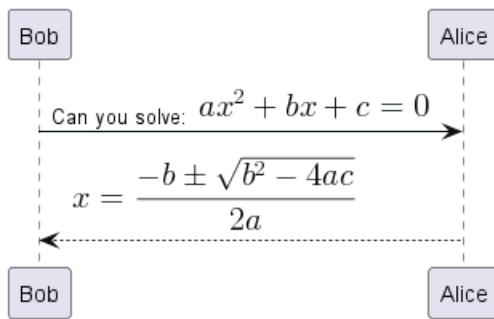
```
@startuml
: $\int_0^1 f(x) dx$ ;
: $x^2 + y_1 + z_{12}^{34}$ ;
note right
Try also
 $\frac{d}{dx} f(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$ 
 $P(y|\mathbf{x})$  or  $f(\mathbf{x}) + \epsilon$ 
end note
@enduml
```



Here is another example:

```
@startuml
Bob -> Alice : Can you solve: <math>ax^2+bx+c=0</math>
Alice --> Bob: <math>x = (-b \pm \sqrt{b^2-4ac})/(2a)</math>
@enduml
```





19.1 Standalone diagram

You can also use `@startmath/@endmath` to create standalone AsciiMath formula.

```

@startmath
f(t)=(a_0)/2 + sum_{n=1}^oo a_ncos((npit)/L)+sum_{n=1}^oo b_n\ sin((npit)/L)
@endmath
  
```

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi t}{L}\right) + \sum_{n=1}^{\infty} b_n \sin\left(\frac{n\pi t}{L}\right)$$

Or use `@startlatex/@endlatex` to create standalone JLaTeXMath formula.

```

@startlatex
\sum_{i=0}^{n-1} (a_i + b_i^2)
@endlatex
  
```

$$\sum_{i=0}^{n-1} (a_i + b_i^2)$$

19.2 How is this working?

To draw those formulas, PlantUML uses two open source projects:

- AsciiMath that converts AsciiMath notation to LaTeX expression;
- JLatexMath that displays mathematical formulas written in LaTeX. JLaTeXMath is the best Java library to display LaTeX code.

ASCIIMathTeXImg.js is small enough to be integrated into PlantUML standard distribution.

Since JLatexMath is bigger, you have to download it separately, then unzip the 4 jar files (*batik-all-1.7.jar*, *jlatexmath-minimal-1.0.3.jar*, *jlm_cyrillic.jar* and *jlm_greek.jar*) in the same folder as PlantUML.jar.



20 Diagrama de Relación de Entidades

Basado en la notación de Ingeniería de la Información.

Esta es una extensión del Diagrama de Clases existente. Esta extensión añade:

- Relaciones adicionales para la notación de Ingeniería de la Información.
- Un alias `entity` que mapea al diagrama de clase `class`.
- Un modificador de visibilidad adicional `*` para identificar atributos obligatorios.

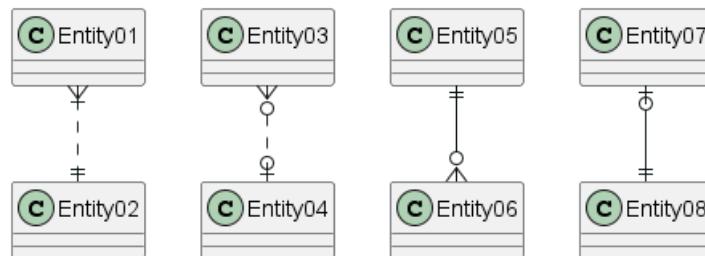
Por lo demás, la sintaxis para dibujar diagramas es la misma que para los diagramas de clase. Todas las demás características de los diagramas de clase también son compatibles

20.1 Information Engineering Relations

Type	Symbol
Zero or One	o--
Exactly One	--
Zero or Many	}o--
One or Many	} --

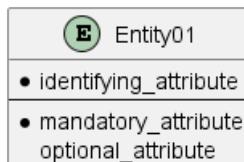
Examples:

```
@startuml
Entity01 }|...|| Entity02
Entity03 }o..o| Entity04
Entity05 ||--o{ Entity06
Entity07 |o--|| Entity08
@enduml
```



20.2 Entities

```
@startuml
entity Entity01 {
    * identifying_attribute
    --
    * mandatory_attribute
    optional_attribute
}
@enduml
```

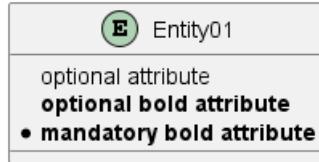


Again, this is the normal class diagram syntax (aside from use of `entity` instead of `class`). Anything that you can do in a class diagram can be done here.



The * visibility modifier can be used to identify mandatory attributes. A space can be used after the modifier character to avoid conflicts with the creole bold:

```
@startuml
entity Entity01 {
    optional attribute
    **optional bold attribute**
    * **mandatory bold attribute**
}
@enduml
```



20.3 Ejemplo completo

```
@startuml

' hide the spot
hide circle

' avoid problems with angled crows feet
skinparam linetype ortho

entity "Entity01" as e01 {
    *e1_id : number <<generated>>
    --
    *name : text
    description : text
}

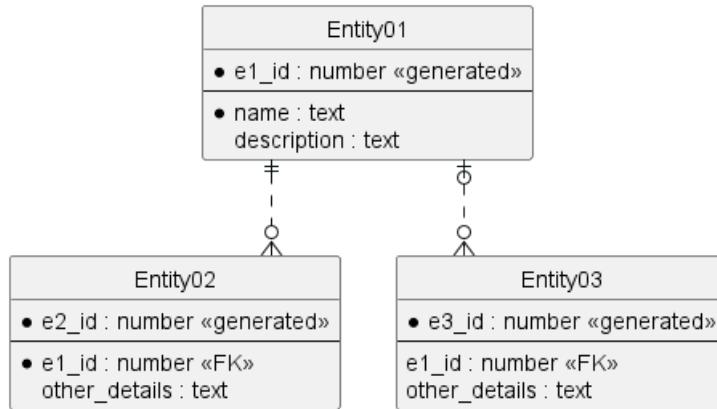
entity "Entity02" as e02 {
    *e2_id : number <<generated>>
    --
    *e1_id : number <<FK>>
    other_details : text
}

entity "Entity03" as e03 {
    *e3_id : number <<generated>>
    --
    e1_id : number <<FK>>
    other_details : text
}

e01 ||..o{ e02
e01 |o..o{ e03

@enduml
```





Actualmente las patas de gallo no se ven muy bien cuando la relación se dibuja en ángulo con la entidad. Esto puede evitarse utilizando el parámetro de piel `linetype ortho`



21 Comandos comunes en PlantUML

Descubra los comandos fundamentales universalmente aplicables a todos los tipos de diagramas en PlantUML. Estos comandos le permiten inyectar versatilidad y detalles personalizados en sus diagramas. A continuación, desglosamos estos comandos comunes en tres categorías principales:

21.0.1 Elementos globales

- **Comentarios:** Añada comentarios o notas explicativas en el guión de su diagrama para transmitir información adicional o para dejar recordatorios para futuras modificaciones.
- **Notas:** Incorpore información suplementaria directamente en su diagrama para ayudar a la comprensión o para resaltar aspectos importantes.
- **Control de Tamaño:** Ajuste las dimensiones de los distintos elementos según sus preferencias, garantizando un diagrama equilibrado y bien proporcionado.
- **Título y leyendas:** Defina un título adecuado y añada subtítulos para aclarar el contexto o para anotar partes específicas de su diagrama.

21.0.2 Descripción de la sintaxis criolla

Aproveche el poder de la sintaxis criolla para dar más formato al contenido de cualquier elemento dentro de su diagrama. Este estilo de marcado wiki permite:

- **Formato de Texto:** Personalice la apariencia de su texto con varios estilos y alineaciones.
- **Listas:** Cree listas ordenadas o desordenadas para presentar la información de forma ordenada.
- **Enlaces:** Integre hipervínculos para facilitar la navegación rápida a recursos relevantes.

21.0.3 Comando de Control de Estilo

Obtenga un control completo sobre el estilo de presentación de sus elementos de diagrama utilizando el comando **style**. Utilícelo para:

- **Definir Estilos:** Establecer estilos uniformes para los elementos para mantener un tema visual cohesivo.
- **Personalizar Colores:** Elija colores específicos para varios elementos para mejorar el atractivo visual y crear clasificaciones distintas.

Explore estos comandos para crear diagramas que sean tanto funcionales como estéticamente agradables, adaptando cada elemento a sus especificaciones exactas.

21.1 Comments

21.1.1 Simple comment

Everything that starts with `simple quote '` is a comment.

```
@startuml
'Line comments use a single apostrophe
@enduml
```

21.1.2 Block comment

Block comment use C-style comments except that instead of `*` you use an apostrophe `'`, then you can also put comments on several lines using `/'` to start and `'/` to end.

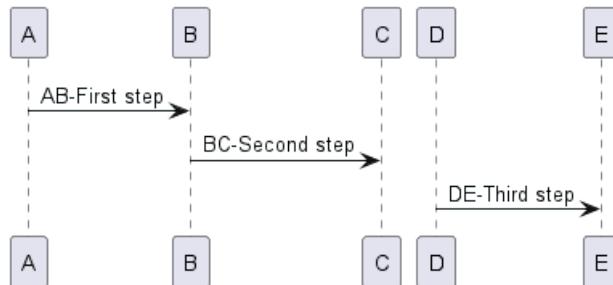
```
@startuml
/'
many lines comments
here
'/
@enduml
```



[Ref. QA-1353]

Then you can also put block comment on the same line, as:

```
@startuml
/* case 1 */
A -> B : AB-First step
B -> C : BC-Second step
/* case 2 */
D -> E : DE-Third step
@enduml
```

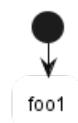


[Ref. QA-3906 and QA-3910]

21.1.3 Full example

```
@startuml
skinparam activity {
    ' this is a comment
    BackgroundColor White
    BorderColor Black /* this is a comment */
    BorderColor Red   ' this is not a comment and this line is ignored
}

start
:foo1;
@enduml
```



[Ref. GH-214]

21.2 Zoom

You can use the `scale` command to zoom the generated image.

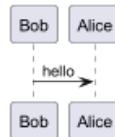
You can use either *a number* or *a fraction* to define the scale factor. You can also specify either `width` or `height` (*in pixel*). And you can also give both `width` and `height`: the image is scaled to fit inside the specified dimension.

- `scale 1.5`
- `scale 2/3`
- `scale 200 width`
- `scale 200 height`
- `scale 200*100`
- `scale max 300*200`
- `scale max 1024 width`



- scale max 800 height

```
@startuml
scale 180*90
Bob->Alice : hello
@enduml
```



21.3 Title

The `title` keyword is used to put a title. You can add newline using `\n` in the title description.

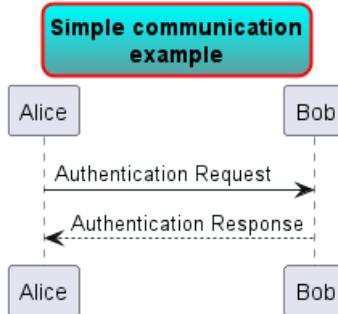
Some `skinparam` settings are available to put borders on the title.

```
@startuml
skinparam titleBorderRoundCorner 15
skinparam titleBorderThickness 2
skinparam titleBorderColor red
skinparam titleBackgroundColor Aqua-CadetBlue

title Simple communication\nexample

Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response

@enduml
```



You can use creole formatting in the title.

You can also define title on several lines using `title` and `end title` keywords.

```
@startuml

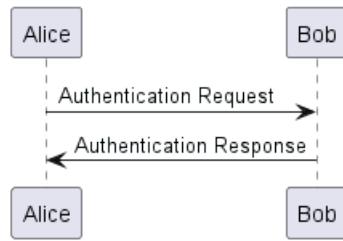
title
<u>Simple</u> communication example
on <i>several</i> lines and using <back:cadetblue>creole tags</back>
end title

Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response

@enduml
```



**Simple communication example
on several lines and using creole tags**



21.4 Caption

There is also a `caption` keyword to put a caption under the diagram.

```
@startuml
```

```
caption figure 1
Alice -> Bob: Hello
```

```
@enduml
```

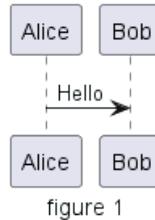


figure 1

21.5 Footer and header

You can use the commands `header` or `footer` to add a footer or a header on any generated diagram.

You can optionally specify if you want a `center`, `left` or `right` footer/header, by adding a keyword.

As with title, it is possible to define a header or a footer on several lines.

It is also possible to put some HTML into the header or footer.

```
@startuml
```

```
Alice -> Bob: Authentication Request
```

```
header
```

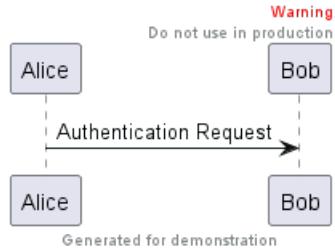
```
<font color=red>Warning:</font>
```

```
Do not use in production.
```

```
endheader
```

```
center footer Generated for demonstration
```

```
@enduml
```

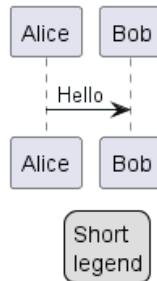


21.6 Legend the diagram

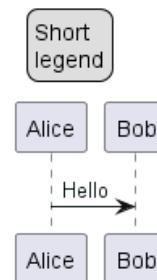
The `legend` and `end legend` are keywords used to put a legend.

You can optionally specify to have `left`, `right`, `top`, `bottom` or `center` alignment for the legend.

```
@startuml
Alice -> Bob : Hello
legend right
Short
legend
endlegend
@enduml
```



```
@startuml
Alice -> Bob : Hello
legend top left
Short
legend
endlegend
@enduml
```



21.7 Appendix: Examples on all diagram

21.7.1 Activity

```
@startuml
header some header

footer some footer

title My title

caption This is caption

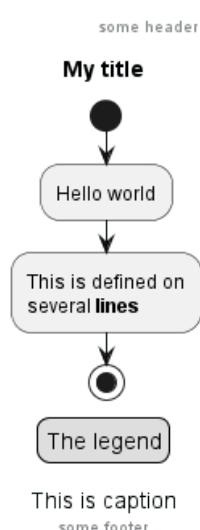
legend
The legend
end legend

start
```



```
:Hello world;
:This is defined on
several **lines**;
stop

@enduml
```



21.7.2 Archimate

```
@startuml
header some header

footer some footer

title My title

caption This is caption

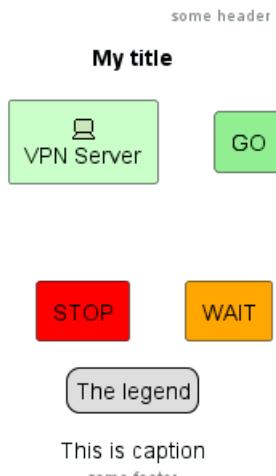
legend
The legend
end legend

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange

@enduml
```





21.7.3 Class

```
@startuml
header some header

footer some footer

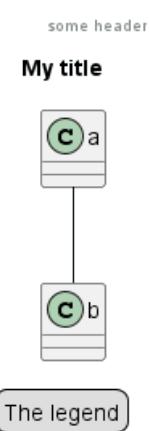
title My title

caption This is caption

legend
The legend
end legend
```

```
a -- b
```

```
@enduml
```



21.7.4 Component, Deployment, Use-Case

```
@startuml
header some header

footer some footer
```



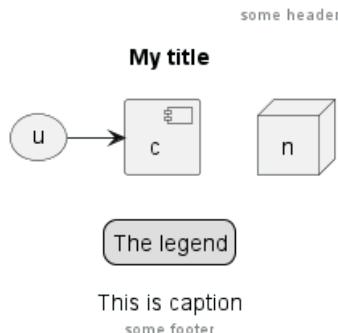
```

title My title
caption This is caption
legend
The legend
end legend

node n
(u) -> [c]

@enduml

```



21.7.5 Gantt project planning

```

@startgantt
header some header

footer some footer

title My title
caption This is caption

legend
The legend
end legend

```

[t] lasts 5 days

```

@endgantt

```

some header

My title

1	2	3	4	5
t				
1	2	3	4	5

The legend

This is caption
some footer

TODO: DONE *[(Header, footer) corrected on V1.2020.18]*

21.7.6 Object

```
@startuml
```

```
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

```
legend
```

```
The legend
```

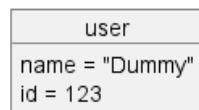
```
end legend
```

```
object user {
    name = "Dummy"
    id = 123
}
```

```
@enduml
```

some header

My title



The legend

This is caption

some footer

21.7.7 MindMap

```
@startmindmap
```

```
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

```
legend
```

```
The legend
```

```
end legend
```

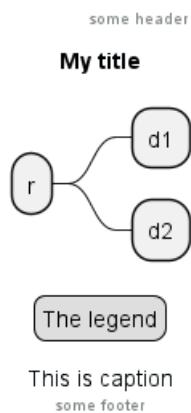
```
* r
```

```
** d1
```

```
** d2
```

```
@endmindmap
```





21.7.8 Network (nwdiag)

```
@startuml
header some header

footer some footer

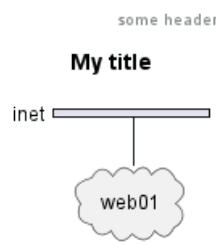
title My title

caption This is caption

legend
The legend
end legend

nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}
```

```
@enduml
```



21.7.9 Sequence

```
@startuml
header some header

footer some footer
```



```

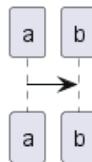
title My title
caption This is caption
legend
The legend
end legend

a->b
@enduml

```

some header

My title



The legend

This is caption

some footer

21.7.10 State

```

@startuml
header some header

```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

```
legend
The legend
end legend
```

```

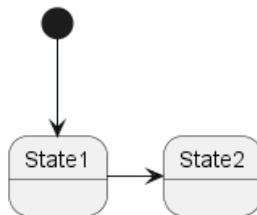
[*] --> State1
State1 -> State2

```

```
@enduml
```

some header

My title



The legend

This is caption

some footer



21.7.11 Timing

```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

robust "Web Browser" as WB
concise "Web User" as WU

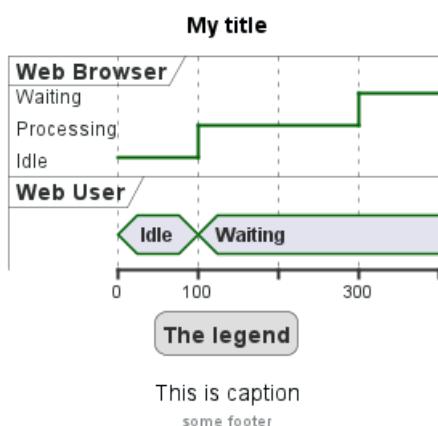
@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting

@enduml
```

some header



This is caption

some footer

21.7.12 Work Breakdown Structure (WBS)

```
@startwbs
header some header

footer some footer

title My title

caption This is caption

legend
```



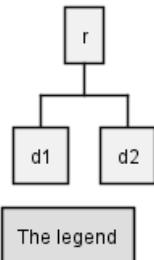
```
The legend
end legend
```

```
* r
** d1
** d2
```

```
@endwbs
```

some header

My title



This is caption

some footer

TODO: DONE [*Corrected on V1.2020.17*]

21.7.13 Wireframe (SALT)

```
@startsalt
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

```
legend
```

```
The legend
```

```
end legend
```

```
{+
    Login | "MyName"
    Password | "****"
    [Cancel] | [ OK ]
}
@endsalt
```



This is caption

some footer



TODO: DONE [Corrected on V1.2020.18]

21.8 Appendix: Examples on all diagram with style

TODO: DONE

FYI:

- all is only good for **Sequence diagram**
- **title**, **caption** and **legend** are good for all diagrams except for **salt diagram**

TODO: FIXME

- Now (*test on 1.2020.18-19*) header, footer are not good for **all other diagrams** except only for **Sequence diagram**.

To be fix; Thanks

TODO: FIXME

Here are tests of **title**, **header**, **footer**, **caption** or **legend** on all the diagram with the debug style:

```
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
```

21.8.1 Activity

```
@startuml
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}
```



```
header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

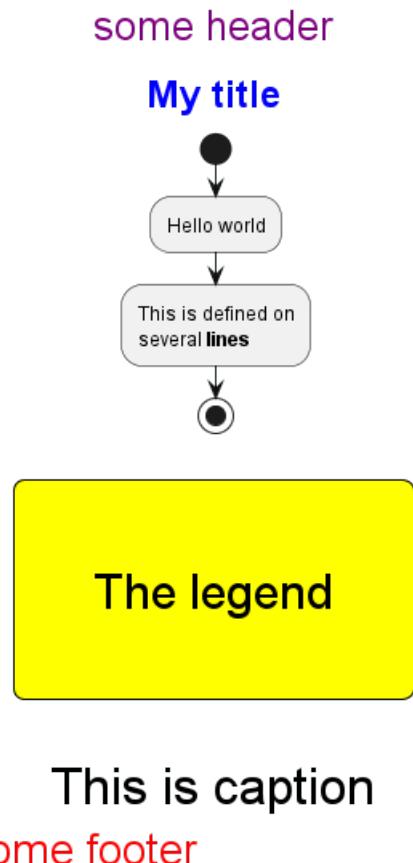
caption This is caption

legend
The legend
end legend

start
:Hello world;
:This is defined on
several **lines**;
stop

@enduml
```





21.8.2 Archimate

```

@startuml
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

```



```

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

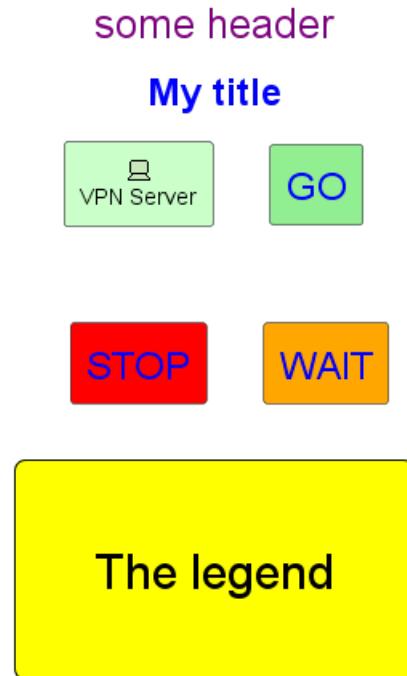
legend
The legend
end legend

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange

@enduml

```



This is caption
some footer

21.8.3 Class

```

@startuml
<style>
title {
    HorizontalAlignment right
    FontSize 24

```



```
FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

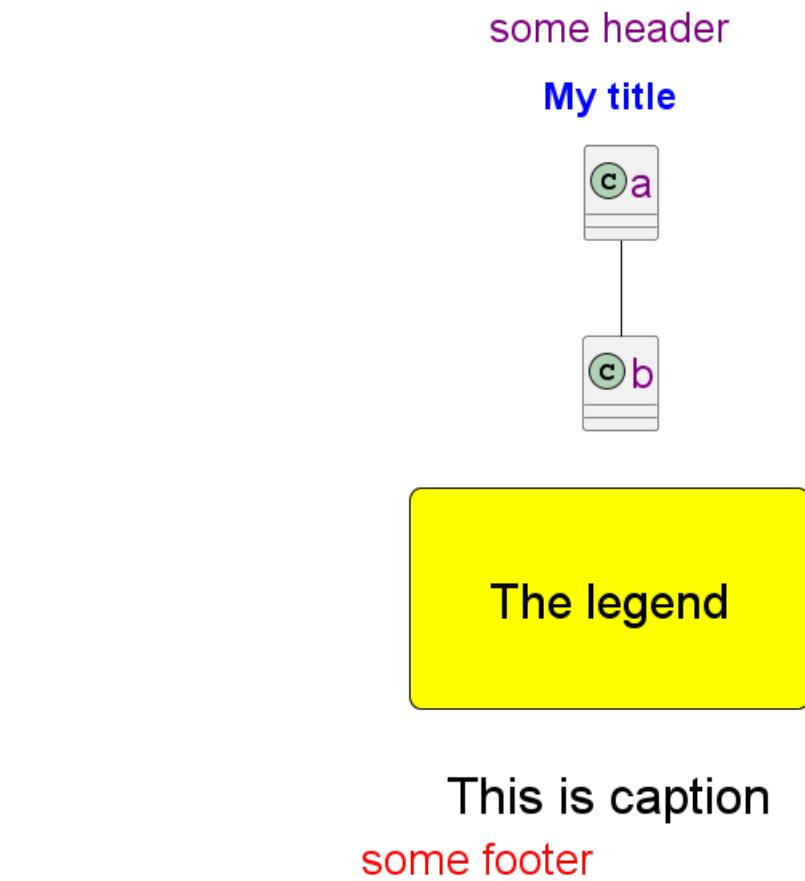
caption This is caption

legend
The legend
end legend

a -- b

@enduml
```





21.8.4 Component, Deployment, Use-Case

```

@startuml
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

```



```

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

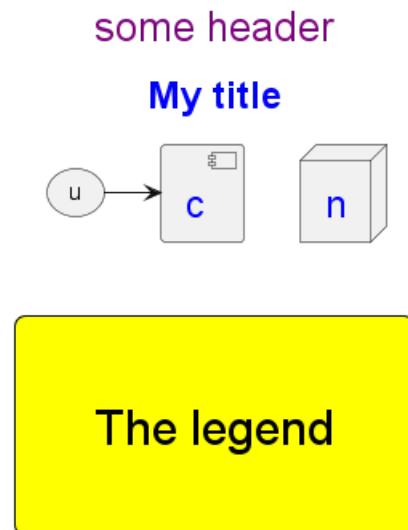
caption This is caption

legend
The legend
end legend

node n
(u) -> [c]

@enduml

```



This is caption
some footer

21.8.5 Gantt project planning

```

@startgantt
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple

```



```

}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

```

[t] lasts 5 days

@endgantt

some header

My title

1	2	3	4	5
t				
1	2	3	4	5

The legend



This is caption

some footer



21.8.6 Object

```

@startuml
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

object user {
    name = "Dummy"
    id = 123
}

@enduml

```



some header

My title

user
name = "Dummy"
id = 123

The legend

This is caption

some footer

21.8.7 MindMap

```
@startmindmap
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

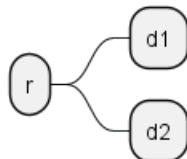
caption {
    FontSize 32
}
</style>
header some header
```



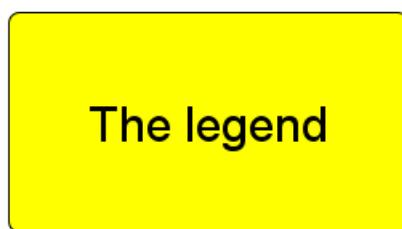
```
footer some footer  
title My title  
caption This is caption  
legend  
The legend  
end legend  
  
* r  
** d1  
** d2  
  
@endumap
```

some header

My title



The legend



This is caption
some footer

21.8.8 Network (nwdiag)

```
@startuml  
<style>  
title {  
    HorizontalAlignment right  
    FontSize 24  
    FontColor blue  
}  
  
header {  
    HorizontalAlignment center  
    FontSize 26  
    FontColor purple  
}
```



```
footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}

@enduml
```





21.8.9 Sequence

```

@startuml
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>

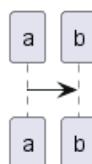
```



```
header some header  
footer some footer  
title My title  
caption This is caption  
legend  
The legend  
end legend  
a->b  
@enduml
```

some header

My title



The legend

This is caption
some footer

21.8.10 State

```
@startuml  
<style>  
title {  
    HorizontalAlignment right  
    FontSize 24  
    FontColor blue  
}  
  
header {  
    HorizontalAlignment center  
    FontSize 26  
    FontColor purple  
}  
  
footer {  
    HorizontalAlignment left  
    FontSize 28
```



```

FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

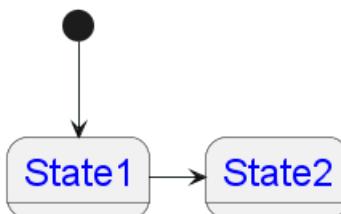
[*] --> State1
State1 -> State2

@enduml

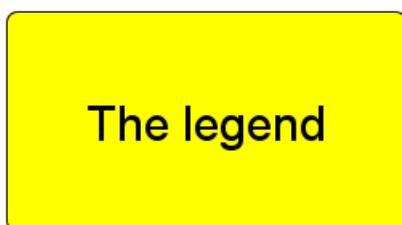
```

some header

My title



The legend



This is caption

some footer



21.8.11 Timing

```

@startuml
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

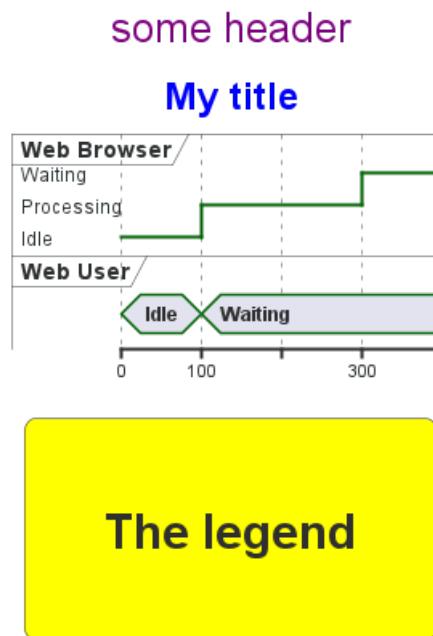
@100
WU is Waiting
WB is Processing

@300
WB is Waiting

```



```
@enduml
```



This is caption
some footer

21.8.12 Work Breakdown Structure (WBS)

```
@startwbs
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}
```



```

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

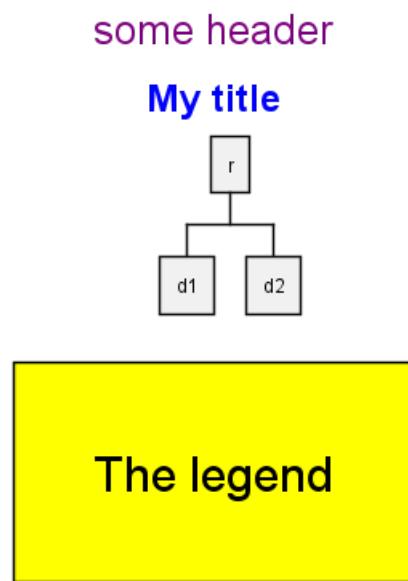
caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endwbs

```



This is caption
some footer

21.8.13 Wireframe (SALT)

TODO:FIXME Fix all (**title**, **caption**, **legend**, **header**, **footer**) for salt. **TODO:**FIXME

```

@startsalt
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}

```



```

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
@startsalt
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

{+
    Login | "MyName"
    Password | "****"
    [Cancel] | [ OK ]
}
@endsalt
some header

```



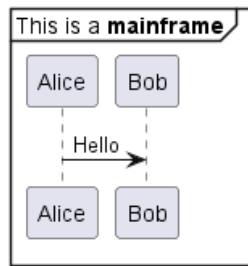
This is caption
some footer

21.9 Mainframe

```
@startuml
mainframe This is a **mainframe**
```



```
Alice->Bob : Hello
@enduml
```



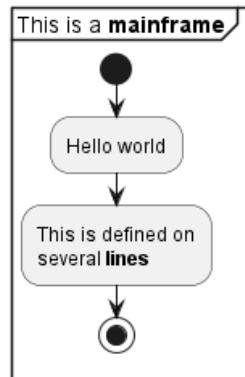
[Ref. QA-4019 and Issue #148]

21.10 Appendix: Examples of Mainframe on all diagram

21.10.1 Activity

```
@startuml
mainframe This is a **mainframe**

start
:Hello world;
:This is defined on
several **lines**;
stop
@enduml
```

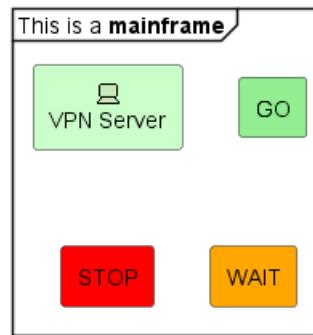


21.10.2 Archimate

```
@startuml
mainframe This is a **mainframe**

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>
rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
@enduml
```





TODO:FIXME Cropped on the top and on the left **TODO:**FIXME

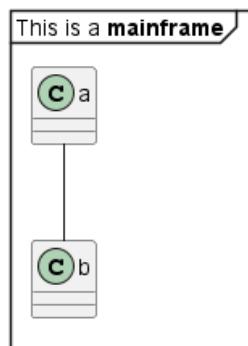
21.10.3 Class

```

@startuml
mainframe This is a **mainframe**

a -- b
@enduml

```



TODO:FIXME Cropped on the top and on the left **TODO:**FIXME

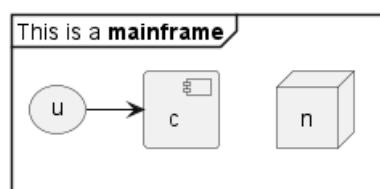
21.10.4 Component, Deployment, Use-Case

```

@startuml
mainframe This is a **mainframe**

node n
(u) -> [c]
@enduml

```



TODO:FIXME Cropped on the top and on the left **TODO:**FIXME

21.10.5 Gantt project planning

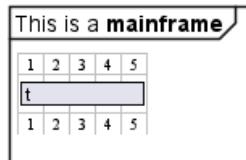
```

@startgantt
mainframe This is a **mainframe**

[t] lasts 5 days
@endgantt

```



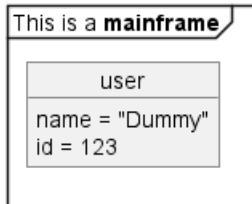


TODO:FIXME Cropped on the top and on the left **TODO:**FIXME

21.10.6 Object

```
@startuml
mainframe This is a **mainframe**

object user {
    name = "Dummy"
    id = 123
}
@enduml
```

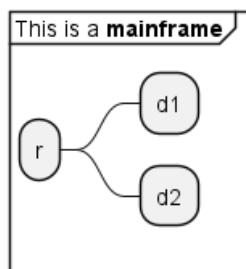


TODO:FIXME Cropped on the top! **TODO:**FIXME

21.10.7 MindMap

```
@startmindmap
mainframe This is a **mainframe**

* r
** d1
** d2
@endmindmap
```

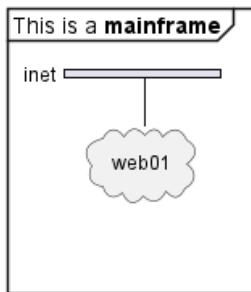


21.10.8 Network (nwdiag)

```
@startuml
mainframe This is a **mainframe**

nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}
@enduml
```





TODO:FIXME Cropped on the top! **TODO:**FIXME

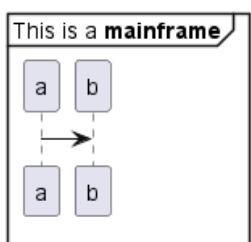
21.10.9 Sequence

```

@startuml
mainframe This is a **mainframe**

a->b
@enduml

```



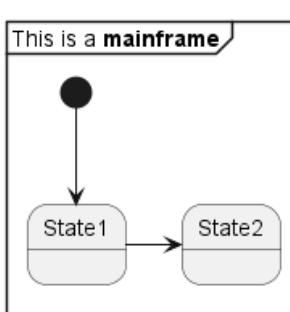
21.10.10 State

```

@startuml
mainframe This is a **mainframe**

[*] --> State1
State1 -> State2
@enduml

```



TODO:FIXME Cropped on the top and on the left **TODO:**FIXME

21.10.11 Timing

```

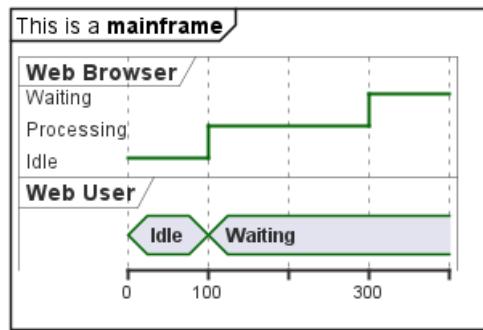
@startuml
mainframe This is a **mainframe**

robust "Web Browser" as WB
concise "Web User" as WU
@0
WU is Idle

```

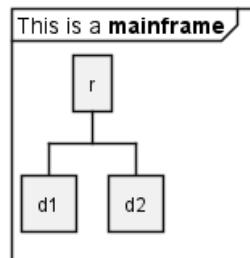


```
WB is Idle
@100
WU is Waiting
WB is Processing
@300
WB is Waiting
@enduml
```



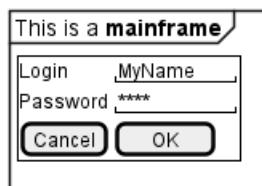
21.10.12 Work Breakdown Structure (WBS)

```
@startwbs
mainframe This is a **mainframe**
* r
** d1
** d2
@endwbs
```



21.10.13 Wireframe (SALT)

```
@startsalt
mainframe This is a **mainframe**
{+
    Login | "MyName"
    Password | "****"
    [Cancel] | [ OK ]
}
@endsalt
```



21.11 Appendix: Examples of title, header, footer, caption, legend and mainframe on all diagram

21.11.1 Activity

```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

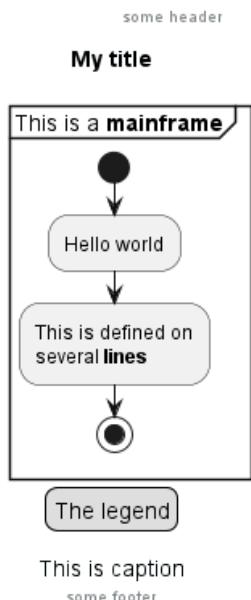
title My title

caption This is caption

legend
The legend
end legend

start
:Hello world;
:This is defined on
several **lines**;
stop

@enduml
```



This is caption
some footer

21.11.2 Archimate

```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption
```



```
legend
The legend
end legend
```

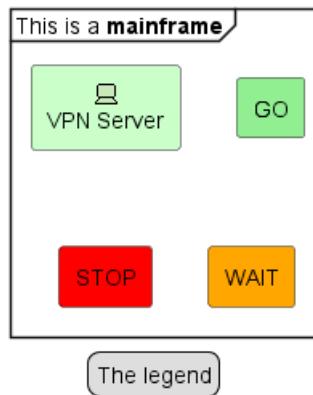
```
archimate #Technology "VPN Server" as vpnServerA <<technology-device>>
```

```
rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
```

```
@enduml
```

some header

My title



The legend

This is caption

some footer

21.11.3 Class

```
@startuml
mainframe This is a **mainframe**
header some header
```

```
footer some footer
```

```
title My title
```

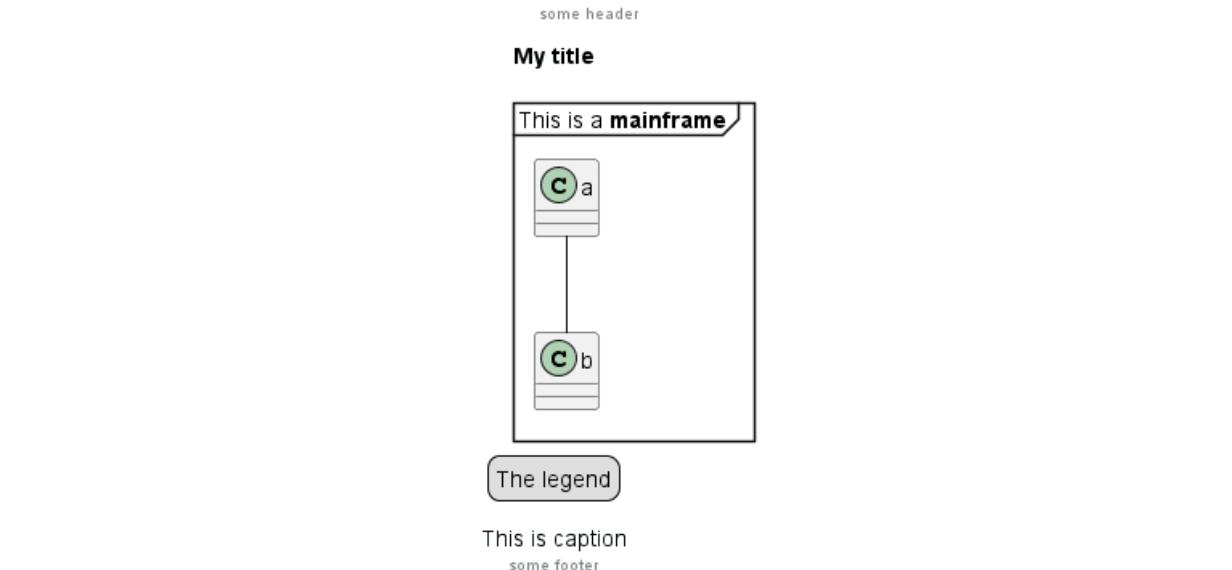
```
caption This is caption
```

```
legend
The legend
end legend
```

```
a -- b
```

```
@enduml
```





21.11.4 Component, Deployment, Use-Case

```

@startuml
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

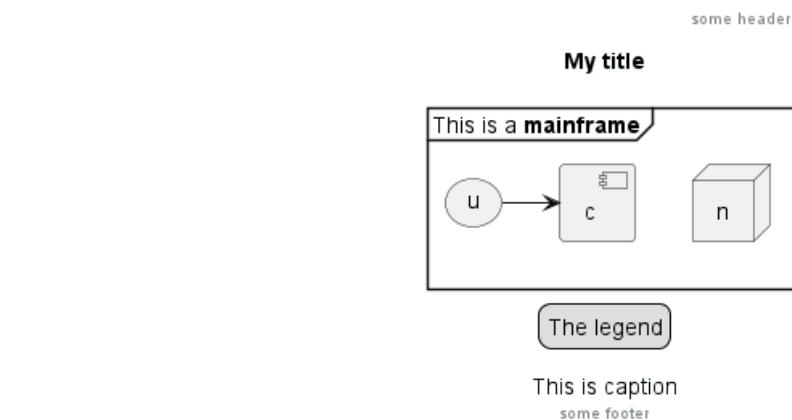
```

```

node n
(u) -> [c]

```

```
@enduml
```



21.11.5 Gantt project planning

```

@startgantt
mainframe This is a **mainframe**
header some header

```



```

footer some footer

title My title

caption This is caption

```

```

legend
The legend
end legend

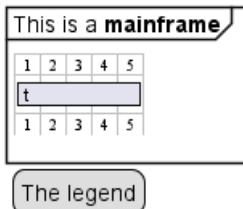
```

```
[t] lasts 5 days
```

```
@enduml
```

some header

My title



This is caption

some footer

21.11.6 Object

```

@startuml
mainframe This is a **mainframe**
header some header

```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

```

legend
The legend
end legend

```

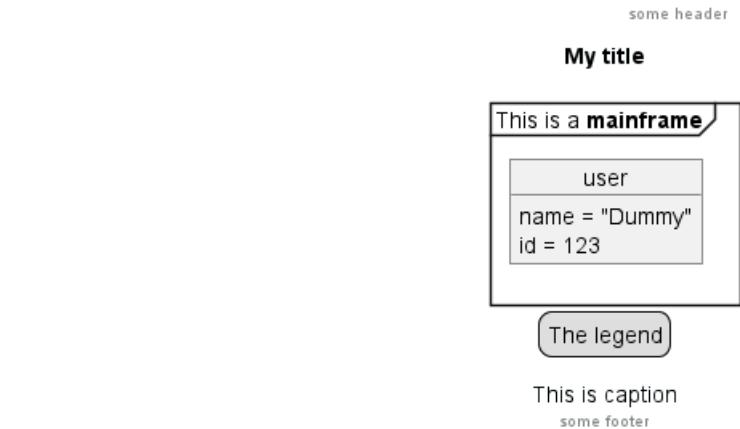
```

object user {
    name = "Dummy"
    id = 123
}

```

```
@enduml
```





21.11.7 MindMap

```

@startmindmap
mainframe This is a **mainframe**
header some header

footer some footer

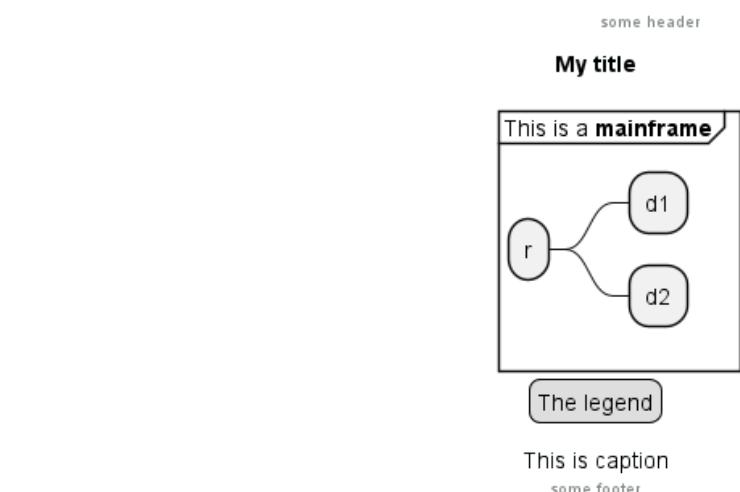
title My title

caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endmindmap
    
```



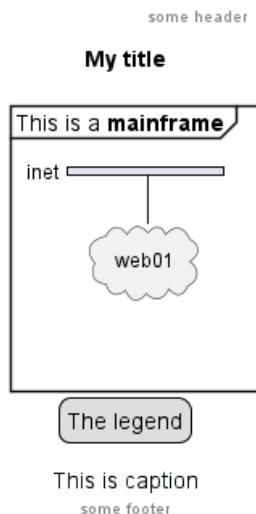
21.11.8 Network (nwdiag)

```

@startuml
mainframe This is a **mainframe**
header some header
    
```



```
footer some footer  
  
title My title  
  
caption This is caption  
  
legend  
The legend  
end legend  
  
nwdiag {  
    network inet {  
        web01 [shape = cloud]  
    }  
}  
  
@enduml
```



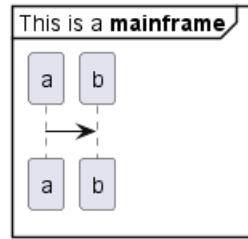
21.11.9 Sequence

```
@startuml  
mainframe This is a **mainframe**  
header some header  
  
footer some footer  
  
title My title  
  
caption This is caption  
  
legend  
The legend  
end legend  
  
a->b  
@enduml
```



some header

My title



The legend

This is caption

some footer

21.11.10 State

```

@startuml
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

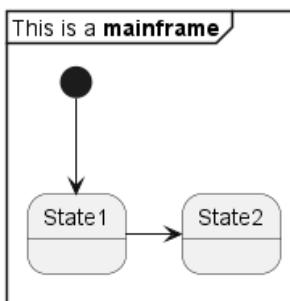
[*] --> State1
State1 -> State2

```

@enduml

some header

My title



The legend

This is caption

some footer

21.11.11 Timing

```

@startuml
mainframe This is a **mainframe**

```



```

header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

robust "Web Browser" as WB
concise "Web User" as WU

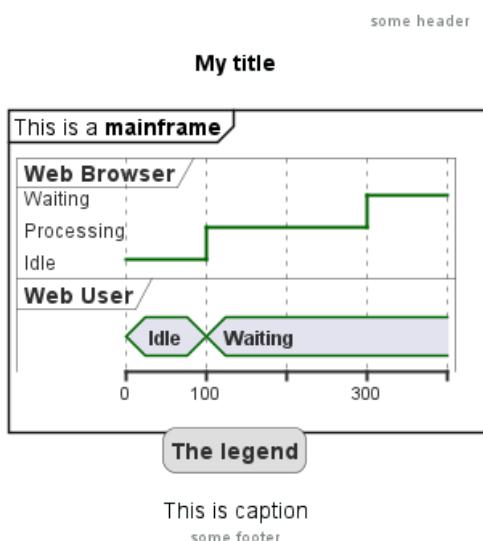
@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting

@enduml

```



21.11.12 Work Breakdown Structure (WBS)

```

@startwbs
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

```



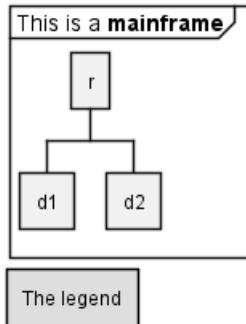
```
legend
The legend
end legend
```

```
* r
** d1
** d2
```

```
@endwbs
```

some header

My title



The legend

This is caption

some footer

21.11.13 Wireframe (SALT)

```
@startsalt
mainframe This is a **mainframe**
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

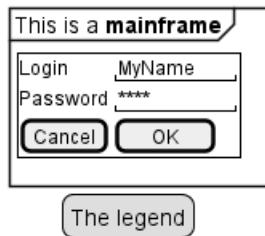
```
legend
The legend
end legend
```

```
{+
    Login | "MyName"
    Password | "*****"
    [Cancel] | [OK]
}
@endsalt
```



some header

My title



This is caption

some footer



22 Creole

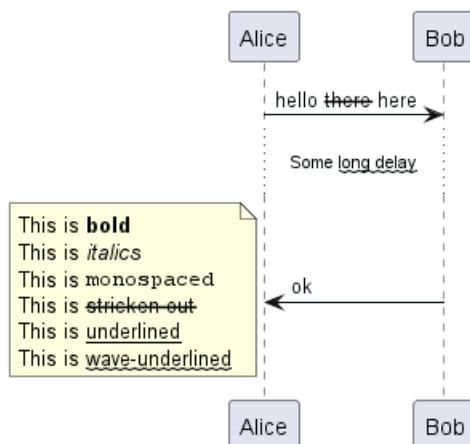
Creole is a lightweight common markup language for various wikis. A light-weight Creole engine is integrated in PlantUML to have a standardized way to emit styled text.

All diagrams support this syntax.

Note that compatibility with HTML syntax is preserved.

22.1 Emphasized text

```
@startuml
Alice -> Bob : hello --there-- here
... Some ~~long delay~~ ...
Bob -> Alice : ok
note left
    This is **bold**
    This is //italics//
    This is ""monospaced"""
    This is --stricken-out--
    This is __underlined__
    This is ~~wave-underlined~~
end note
@enduml
```



22.2 Lists

You can use numbered and bulleted lists in node text, notes, etc.

TODO: FIXME You cannot quite mix numbers and bullets in a list and its sublist.

```
@startuml
object demo {
    * Bullet list
    * Second item
}
note left
    * Bullet list
    * Second item
    ** Sub item
end note
```

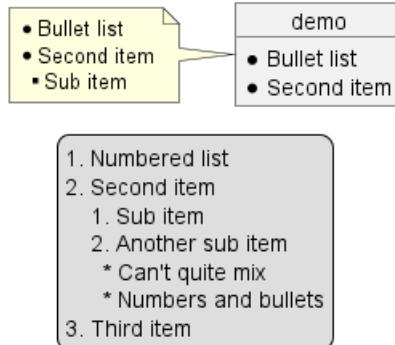
```
legend
# Numbered list
# Second item
## Sub item
```



```

## Another sub item
    * Can't quite mix
    * Numbers and bullets
# Third item
end legend
@enduml

```



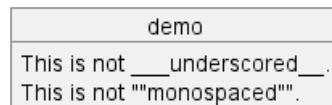
22.3 Escape character

You can use the tilde ~ to escape special creole characters.

```

@startuml
object demo {
    This is not ~__underscored__.
    This is not ~""monospaced"".
}
@enduml

```

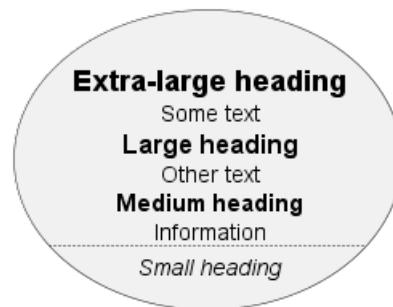


22.4 Headings

```

@startuml
usecase UC1 as "
= Extra-large heading
Some text
== Large heading
Other text
=== Medium heading
Information
.....
===== Small heading"
@enduml

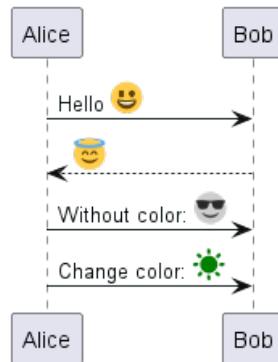
```



22.5 Emoji

All emojis from Twemoji (see EmojiTwo on Github) are available using the following syntax:

```
@startuml
Alice -> Bob : Hello <:1f600:>
return <:innocent:>
Alice -> Bob : Without color: <#0:sunglasses:>
Alice -> Bob : Change color: <#green:sunny:>
@enduml
```



Unlike Unicode Special characters that depend on installed fonts, the emoji are always available. Furthermore, emoji are already colored, but you can recolor them if you like (see examples above).

One can pick emoji from the emoji cheat sheet, the Unicode full-emoji-list, or the flat list emoji.txt in the plantuml source.

You can also use the following PlantUML command to list available emoji:

```
@startuml
emoji <block>
@enduml
```

As of 13 April 2023, you can select between 1174 emoji from the following Unicode blocks:

- Unicode block 26: 83 emoji
- Unicode block 27: 33 emoji
- Unicode block 1F3: 246 emoji
- Unicode block 1F4: 255 emoji
- Unicode block 1F5: 136 emoji
- Unicode block 1F6: 181 emoji
- Unicode block 1F9: 240 emoji

22.5.1 Unicode block 26

```
@startuml
emoji 26
@enduml
```



Emoji available on Unicode Block 26

(Blocks available: 26, 27, 1F3, 1F4, 1F5, 1F6, 1F9)

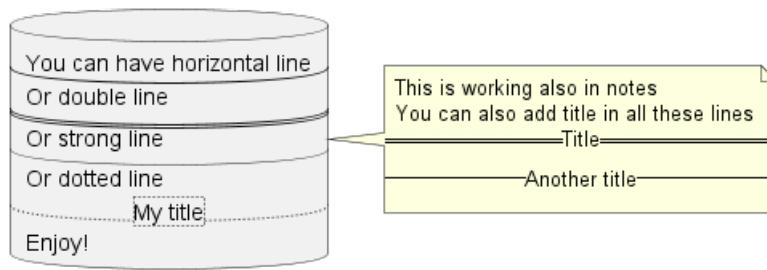
<:2600:> ☀️ ☁️ <:sunny:>	<:264d:> 🌉 🌉 <:virgo:>	<:26aa:> ⚡ ⚡ <:white_circle:>
<:2601:> ☁️ ☁️ <:cloud:>	<:264e:> ♐ ♐ <:libra:>	<:26ab:> ⚫ ⚫ <:black_circle:>
<:2602:> ☂️ ☂️ <:open_umbrella:>	<:264f:> 🌈 🌈 <:scorpius:>	<:26b0:> 💀 💀 <:coffin:>
<:2603:> 🎅️ 🎅️ <:snowman_with_snow:>	<:2650:> ✸ ✸ <:sagittarius:>	<:26b1:> 💀 💀 <:funeral_urn:>
<:2604:> 🍃 🍃 <:comet:>	<:2651:> 🌋 🌋 <:capricorn:>	<:26b2:> ⚽ ⚽ <:soccer:>
<:260e:> 📞 📞 <:phone:>	<:2652:> 🌊 🌊 <:aquarius:>	<:26be:> ⚾ ⚾ <:baseball:>
<:2611:> ✅ ✅ <:ballot_box_with_check:>	<:2653:> 🌊 🌊 <:pisces:>	<:26c4:> 🎅️ 🎅️ <:snowman:>
<:2614:> ☂️ ☂️ <:umbrella:>	<:265f:> 🕸️ 🕸️ <:chess_pawn:>	<:26c5:> ☀️ ☁️ <:partly_sunny:>
<:2615:> ☕ ☕ <:coffee:>	<:2660:> ♠️ ♠️ <:spades:>	<:26c8:> ☁️ ☁️ <:cloud_with_lightning_and_rain:>
<:2618:> 🍀 🍀 <:shamrock:>	<:2663:> ♣️ ♣️ <:clubs:>	<:26ce:> 🐍 🐍 <:ophiuchus:>
<:261d:> 🤝 🤝 <:point_up:>	<:2665:> ❤️ ❤️ <:hearts:>	<:26cf:> ✕ ✕ <:pick:>
<:2620:> 💀 💀 <:skull_and_crossbones:>	<:2666:> ♦️ ♦️ <:diamonds:>	<:26d1:> 🚒 🚒 <:rescue_worker_helmet:>
<:2622:> ☣ ☣ <:radioactive:>	<:2668:> 💧 💧 <:hotsprings:>	<:26d3:> 🔒 🔒 <:chains:>
<:2623:> ☣ ☣ <:biohazard:>	<:267b:> 🌱 🌱 <:recycle:>	<:26d4:> 🚫 🚫 <:no_entry:>
<:2626:> ✝ ✝ <:orthodox_cross:>	<:267e:> ∞ ∞ <:infinity:>	<:26e9:> 🏮 🏮 <:shinto_shrine:>
<:262a:> ☪ ☪ <:star_and_crescent:>	<:267f:> 🚹 🚹 <:wheelchair:>	<:26ea:> 🏯 🏯 <:church:>
<:262e:> ☮ ☮ <:peace_symbol:>	<:2692:> 🔨 🔨 <:hammer_and_pick:>	<:26f0:> 🏔 🏔 <:mountain:>
<:262f:> ☯ ☯ <:yin_yang:>	<:2693:> 🛩 🛩 <:anchor:>	<:26f1:> ☼ ☼ <:parasol_on_ground:>
<:2638:> ☮ ☮ <:wheel_of_dharma:>	<:2694:> ✕ ✕ <:crossed_swords:>	<:26f2:> 🌿 🌿 <:fountain:>
<:2639:> 😢 😢 <:frowning_face:>	<:2695:> 🌙 🌙 <:medical_symbol:>	<:26f3:> 🏴 🏴 <:golf:>
<:263a:> 😊 😊 <:relaxed:>	<:2696:> 🔠 🔠 <:balance_scale:>	<:26f4:> 🛰 🛰 <:ferry:>
<:2640:> ☵ ☵ <:female_sign:>	<:2697:> 🍷 🍷 <:alembic:>	<:26f5:> 🛳 🛳 <:boat:>
<:2642:> ☶ ☶ <:male_sign:>	<:2699:> 🛡 🛡 <:gear:>	<:26f7:> 🛹 🛹 <:skier:>
<:2648:> ☪ ☪ <:aries:>	<:269b:> ☣ ☣ <:atom_symbol:>	<:26f8:> 🛹 🛹 <:ice_skate:>
<:2649:> ☤ ☤ <:taurus:>	<:269c:> 🌸 🌸 <:fleur_de_lis:>	<:26f9:> 🎈 🎈 <:bouncing_ball_person:>
<:264a:> ☪ ☪ <:gemini:>	<:26a0:> ⚠️ ⚠️ <:warning:>	<:26fa:> ☺ ☺ <:tent:>
<:264b:> ☤ ☤ <:cancer:>	<:26a1:> ⚡ ⚡ <:zap:>	<:26fd:> 🛢 🛢 <:fuelpump:>
<:264c:> ☥ ☥ <:leo:>	<:26a7:> 🌈 🌈 <:transgender_symbol:>	

22.6 Horizontal lines

```
@startuml
database DB1 as "
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
Enjoy!
"
note right
This is working also in notes
You can also add title in all these lines
==Title==
--Another title--
end note

@enduml
```





22.7 Links

You can also use URL and links.

Simple links are define using two square brackets (or three square brackets for field or method on class diagram).

Example:

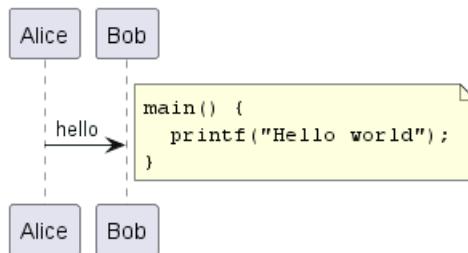
- [[http://plantuml.com]]
- [[http://plantuml.com This label is printed]]
- [[http://plantuml.com{Optional tooltip} This label is printed]]

URL can also be authenticated.

22.8 Code

You can use `<code>` to display some programming code in your diagram (sorry, syntax highlighting is not yet supported).

```
@startuml
Alice -> Bob : hello
note right
<code>
main() {
    printf("Hello world");
}
</code>
end note
@enduml
```



This is especially useful to illustrate some PlantUML code and the resulting rendering:

```
@startuml
hide footbox
note over Source
<code>
This is **bold**
This is //italics//
This is ""monospaced"""
This is --stricken-out--
This is __underlined__

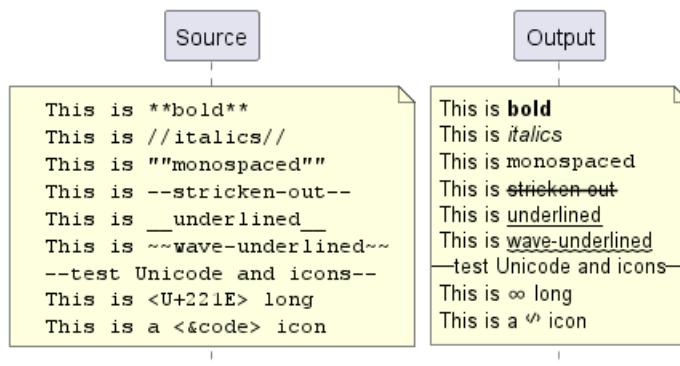
```



```

This is ~~wave-underlined~~
--test Unicode and icons--
This is <U+221E> long
This is a <&code> icon
</code>
end note
/note over Output
This is **bold**
This is //italics//
This is ""monospaced"""
This is --stricken-out--
This is __underlined__
This is ~~wave-underlined~~
--test Unicode and icons--
This is <U+221E> long
This is a <&code> icon
end note
@enduml

```



22.9 Table

22.9.1 Create a table

It is possible to build table, with | separator.

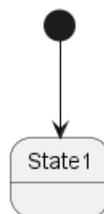
```

@startuml
skinparam titleFontSize 14
title
Example of simple table
|= |= table |= header |
| a | table | row |
| b | table | row |
end title
[*] --> State1
@enduml

```

Example of simple table

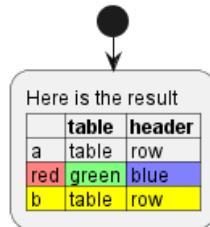
table	header
a	row
b	row



22.9.2 Add color on rows or cells

You can specify background colors of rows and cells:

```
@startuml
start
:Here is the result
|= |= table |= header |
| a | table | row |
|<#FF8080> red |<#80FF80> green |<#8080FF> blue |
<#yellow>| b | table | row |
@enduml
```



22.9.3 Add color on border and text

You can also specify colors of text and borders.

```
@startuml
title
<#lightblue,#red>|= Step |= Date |= Name |= Status |= Link |
<#lightgreen>| 1.1 | TBD | plantuml news |<#Navy><color:OrangeRed><b> Unknown | [[https://plantuml.com]]
end title
@enduml
```

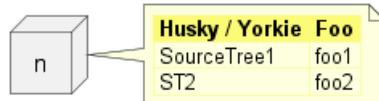
Step	Date	Name	Status	Link
1.1	TBD	plantuml news	Unknown	plantuml news

[Ref. QA-7184]

22.9.4 No border or same color as the background

You can also set the border color to the same color as the background.

```
@startuml
node n
note right of n
<#FBFB77,#FBFB77>|= Husky / Yorkie |= Foo |
| SourceTree1 | foo1 |
| ST2 | foo2 |
end note
@enduml
```



[Ref. QA-12448]

22.9.5 Bold header or not

= as the first char of a cell indicates whether to make it bold (usually used for headers), or not.



```

@startuml
note as deepCSS0
|<#white> Husky / Yorkie |
|= <#gainsboro> SourceTree0 |
endnote

note as deepCSS1
|= <#white> Husky / Yorkie |= Foo |
|<#gainsboro><r> SourceTree1 | foo1 |
endnote

note as deepCSS2
|= Husky / Yorkie |
|<#gainsboro> SourceTree2 |
endnote

note as deepCSS3
<#white>|= Husky / Yorkie |= Foo |
|<#gainsboro> SourceTree1 | foo1 |
endnote
@enduml

```



[Ref. QA-10923]

22.10 Tree

You can use `|_` characters to build a tree.

On common commands, like title:

```

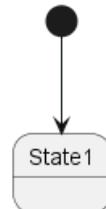
@startuml
skinparam titleFontSize 14
title
    Example of Tree
    |_ First line
    |_ **Bom (Model)**
        |_ prop1
        |_ prop2
        |_ prop3
    |_ Last line
end title
[*] --> State1
@enduml

```



Example of Tree

- First line
- Bom (Model)
- prop1
- prop2
- prop3
- Last line



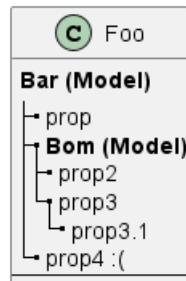
On Class diagram.

(Please note how we have to use an empty second compartment, else the parentheses in **(Model)** cause that text to be moved to a separate first compartment):

```

@startuml
class Foo {
**Bar (Model)**
|_ prop
|_ **Bom (Model)**
|_ prop2
|_ prop3
    |_ prop3.1
|_ prop4 :(
-- 
}
@enduml

```



[Ref. QA-3448]

On Component or Deployment diagrams:

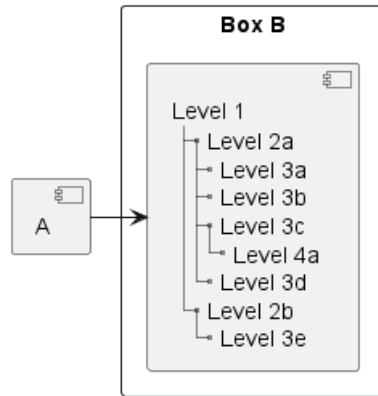
```

@startuml
[A] as A
rectangle "Box B" {
    component B [
        Level 1
        |_ Level 2a
        |_ Level 3a
        |_ Level 3b
        |_ Level 3c
        |_ Level 4a
        |_ Level 3d
        |_ Level 2b
        |_ Level 3e
    ]
}

```



```
]
}
A -> B
@enduml
```

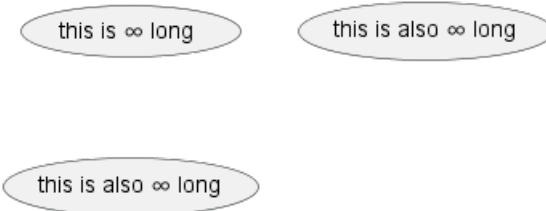


[Ref. QA-11365]

22.11 Special characters

It's possible to use any unicode character, either directly or with syntax `&#nnnnnn;` (decimal) or `<U+XXXXXX>` (hex):

```
@startuml
usecase direct as "this is ☺ long"
usecase ampHash as "this is also ☺ long"
usecase angleBrackets as "this is also <U+221E> long"
@enduml
```

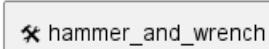


Please note that not all Unicode chars appear correctly, depending on installed fonts.

- You can use the listfonts command with a test string of your desired characters, to see which fonts may include them.
- For characters that are emoji, it's better to use the Emoji notation that doesn't depend on installed fonts, and the emoji are colored.
- The PlantUML server has the "Noto Emoji" font that has most emoji. If you want to render diagrams on your local system, you should check which fonts you have.
- Unfortunately "Noto Emoji" lacks normal chars, so you need to switch fonts, eg

```
@startuml
rectangle "<font:Noto Emoji><U+1F3F7></font> label"
rectangle "<font:Noto Emoji><U+1F527></font> wrench"
rectangle "<font:Noto Emoji><U+1F6E0></font> hammer_and_wrench"
@enduml
```





See Issue 72 for more details.

22.12 Legacy HTML

You can mix Creole with the following HTML tags:

- for bold text
- <u> or <u:#AAAAAA> or <u: [[color|colorName]]> for underline
- <i> for italic
- <s> or <s:#AAAAAA> or <s: [[color|colorName]]> for strike text
- <w> or <w:#AAAAAA> or <w: [[color|colorName]]> for wave underline text
- <plain> for plain text
- <color:#AAAAAA> or <color: [[color|colorName]]>
- <back:#AAAAAA> or <back: [[color|colorName]]> for background color
- <size:nn> to change font size
- <img:file> : the file must be accessible by the filesystem
- <img:http://plantuml.com/logo3.png> : the URL must be available from the Internet

```
@startuml
/* You can change <color:red>text color</color>
 * You can change <back:cadetblue>background color</back>
 * You can change <size:18>size</size>
 * You use <u>legacy</u> <b>HTML <i>tag</i></b>
 * You use <u:red>color</u> <s:green>in HTML</s> <w:#0000FF>tag</w>
-----
* Use image : <img:http://plantuml.com/logo3.png>
;
@enduml
```

- You can change text color
- You can change background color
- You can change size
- You use legacy HTML tag
- You use color in HTML tag



• Use image :



22.12.1 Common HTML element

```

@startuml
hide footbox
note over Source
<code>
    This is <b>bold</b>
    This is <i>italics</i>
    This is <font:monospaced>monospaced</font>
    This is <s>stroked</s>
    This is <u>underlined</u>
    This is <w>waved</w>
    This is <s:green>stroked</s>
    This is <u:red>underlined</u>
    This is <w:#0000FF>waved</w>
    This is <b>a bold text containing <plain>plain text</plain> inside</b>
    -- other examples --
    This is <color:blue>Blue</color>
    This is <back:orange>Orange background</back>
    This is <size:20>big</size>
</code>
end note
/note over Output
    This is <b>bold</b>
    This is <i>italics</i>
    This is <font:monospaced>monospaced</font>
    This is <s>stroked</s>
    This is <u>underlined</u>
    This is <w>waved</w>
    This is <s:green>stroked</s>
    This is <u:red>underlined</u>
    This is <w:#0000FF>waved</w>
    This is <b>a bold text containing <plain>plain text</plain> inside</b>
    -- other examples --
    This is <color:blue>Blue</color>
    This is <back:orange>Orange background</back>
    This is <size:20>big</size>
end note
@enduml

```



[Ref. QA-5254 for plain]



22.12.2 Subscript and Superscript element [sub, sup]

```
@startuml
:<code>
This is the "caffeine" molecule: C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>
</code>
This is the "caffeine" molecule: C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>
-----
<code>
This is the Pythagorean theorem: a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup>
</code>
This is the Pythagorean theorem: a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup>;
@enduml
```

This is the "caffeine" molecule: C₈H₁₀N₄O₂
 This is the "caffeine" molecule: C₈H₁₀N₄O₂

This is the Pythagorean theorem: a² + b² = c²
 This is the Pythagorean theorem: a² + b² = c²

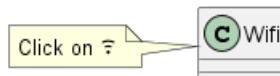
22.13 OpenIconic

OpenIconic is a very nice open-source icon set. Those icons are integrated in the creole parser, so you can use them out-of-the-box.

Use the following syntax: <&ICON_NAME>.

```
@startuml
title: <size:20><&heart>Use of OpenIconic<&heart></size>
class Wifi
note left
  Click on <&wifi>
end note
@enduml
```

♥Use of OpenIconic♥



The complete list is available with the following special command:

```
@startuml
listopeniconic
@enduml
```



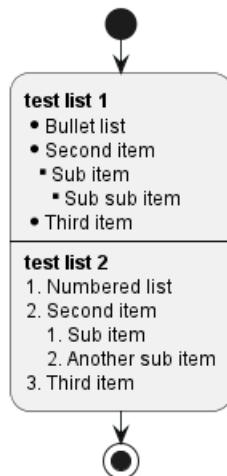
List Open Iconic	◆ bell	◆ cloud	≡ excerpt	≡ justify-right	♪ musical-note	★ star
Credit to https://useiconic.com/open	✿ bluetooth	▲ cloudy	Ξ expand-down	❖ key	☛ paperclip	☀ sun
B bold	▷ code	▷ expand-left	▷ expand-right	▢ laptop	❖ pencil	▢ tablet
✚ bolt	❖ cog	▷ collapse-down	▷ collapse-left	▢ layers	❖ people	❖ tag
↯ account-login	■ book	▷ collapse-right	▷ collapse-up	❖ lightbulb	❖ person	❖ tags
↯ account-logout	■ bookmark	▷ comment-square	▷ external-link	▷ link-broken	▢ phone	◎ target
↶ action-redo	■ box	▷ contrast	❖ eye	▷ link-intact	❖ pie-chart	☒ task
↶ action-undo	■ briefcase	▷ crop	❖ file	❖ list-rich	✚ pin	▣ terminal
≡ align-center	£ british-pound	▷ dashboard	❖ fire	≡ list	● play-circle	TeX text
≡ align-left	▢ browser	▷ data-transfer-download	❖ flag	◀ location	+ plus	☛ thumb-down
≡ align-right	✓ brush	▷ data-transfer-upload	❖ flash	❖ lock-locked	○ power-standby	◀ thumb-up
✖ aperture	✿ bug	▷ delete	❖ folder	❖ lock-unlocked	✖ print	⌚ timer
↓ arrow-bottom	﴿ bullhorn	▷ dial	❖ fork	▷ loop-circular	▷ project	☒ transfer
○ arrow-circle-bottom	■ calculator	▷ document	❖ globe	▷ loop-square	❖ pulse	❖ trash
○ arrow-circle-left	■ calendar	▷ dollar	❖ graph	▷ loop	❖ puzzle-piece	underline
○ arrow-circle-right	■ camera-slr	▷ droplet	❖ grid-four-up	❖ magnifying-glass	? question-mark	▣ vertical-align-bottom
○ arrow-circle-top	▼ caret-bottom	▷ eject	❖ grid-three-up	❖ map-marker	❖ rain	▣ vertical-align-center
← arrow-left	◀ caret-left	▷ envelope-closed	❖ grid-two-up	❖ map	✖ random	▣ vertical-align-top
→ arrow-right	▶ caret-right	▷ envelope-open	❖ hard-drive	▷ media-pause	▷ resize-both	▶ volume-high
↓ arrow-thick-bottom	▲ caret-top	▷ info	❖ header	▶ media-play	❖ resize-height	◀ volume-low
← arrow-thick-left	◀ cart	▷ italic	❖ heart	● media-record	❖ resize-width	◀ volume-off
→ arrow-thick-right	■ chat	▷ infinity	❖ headphones	◀ media-skip-backward	▶ rss-alt	⚠ warning
↑ arrow-thick-top	✓ check	▷ justfy-center	❖ home	▶ media-skip-forward	❖ rss	❖ wifi
↑ arrow-top	▼ chevron-bottom	▷ justfy-left	❖ image	◀ media-step-backward	❖ script	❖ wrench
❖ audio-spectrum	◀ chevron-left	▷ justfy-right	❖ inbox	▶ media-step-forward	❖ share-boxed	✖ x
“ audio	▶ chevron-right	” double-quote-sans-left	❖ infinity	❖ menu	❖ share	¥ yen
❖ badge	▲ chevron-top	” double-quote-sans-right	❖ info	❖ medical-cross	❖ shield	❖ zoom-in
○ ban	○ circle-check	” double-quote-serif-left	❖ italic	❖ minus	❖ signal	❖ zoom-out
■ bar-chart	○ circle-x	” double-quote-serif-right	❖ justfy-center	❖ monitor	▶ signpost	▶ sort-ascending
❖ basket	■ clipboard	” ellipses	❖ justfy-left	❖ moon	◀ sort-descending	◀ sort-descending
□ battery-empty	○ clock	” envelope-closed	❖ justfy-right	❖ move	■ spreadsheet	■ spreadsheet
■ battery-full	▲ cloud-download	” envelope-open	€ euro			
✿ beaker	▲ cloud-upload					

22.14 Appendix: Examples of "Creole List" on all diagrams

22.14.1 Activity

```
@startuml
start
:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item;
stop
@enduml
```





22.14.2 Class

TODO: FIXME

- *Sub item*
- *Sub sub item*

TODO: FIXME

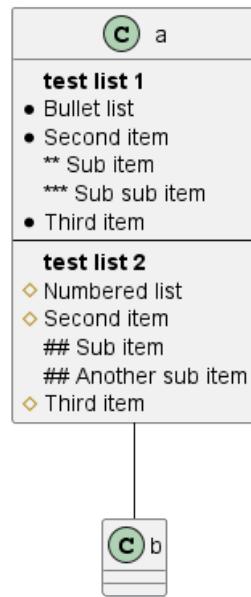
```
@startuml
```

```
class a {
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}
```

```
a -- b
```

```
@enduml
```





22.14.3 Component, Deployment, Use-Case

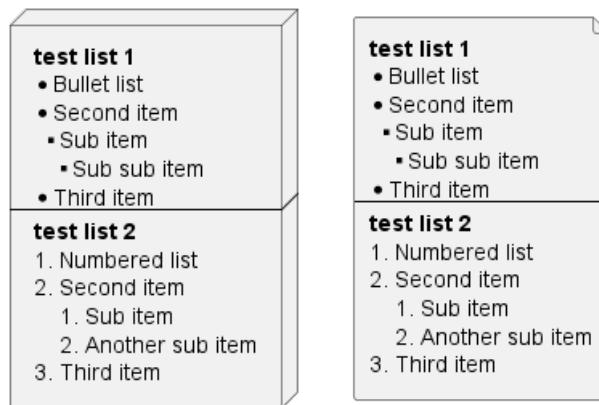
```

@startuml
node n [
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
]

file f as "
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
"
@enduml

```





TODO: DONE [Corrected in V1.2020.18]

22.14.4 Gantt project planning

N/A

22.14.5 Object

TODO: FIXME

- *Sub item*
- *Sub sub item*

TODO: FIXME

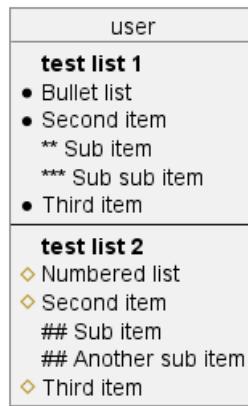
```

@startuml
object user {
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}

```

@enduml



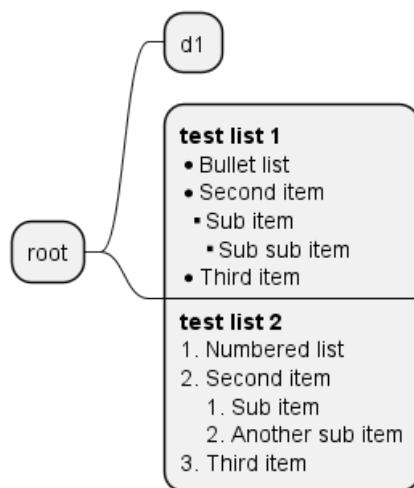


22.14.6 MindMap

```
@startmindmap
```

```
* root
** d1
**:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item;
```

```
@endmindmap
```

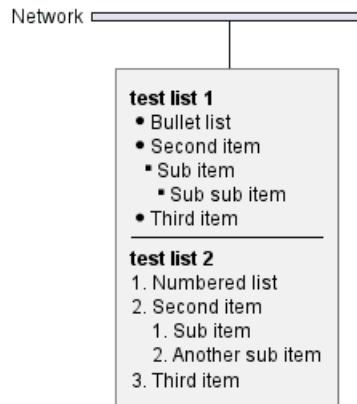


22.14.7 Network (nwdiag)

```
@startuml
nwdiag {
```

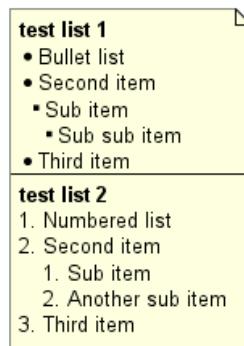


```
network Network {
    Server [description="**test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item"]
}
@enduml
```



22.14.8 Note

```
@startuml
note as n
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
end note
@enduml
```



22.14.9 Sequence

```
@startuml
<style>
participant {HorizontalAlignment left}
</style>
```



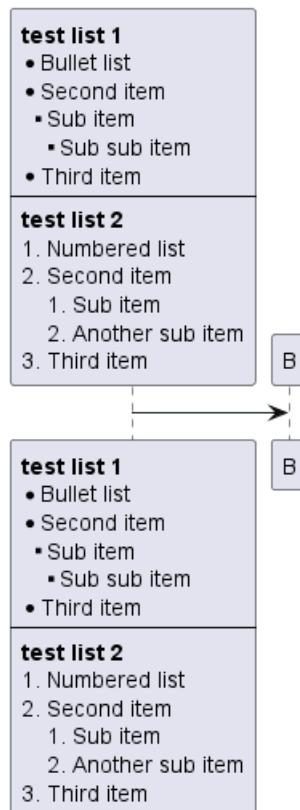
```

participant Participant [
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
]

participant B

```

Participant -> B
@enduml



[Ref. QA-15232]

22.14.10 State

```

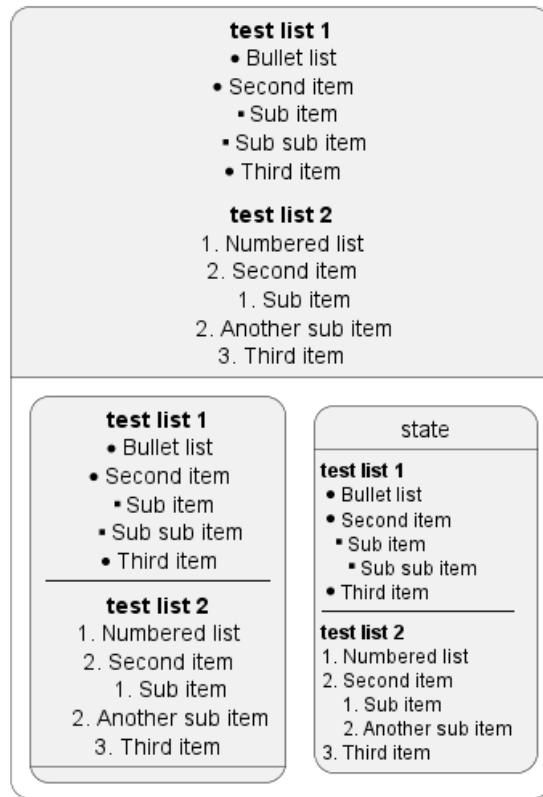
@startuml
<style>
stateDiagram {
title {HorizontalAlignment left}
}
</style>
state "***test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n--"
state "***test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n--"

```



```
state : **test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n}
```

@enduml



[Ref. QA-16978]

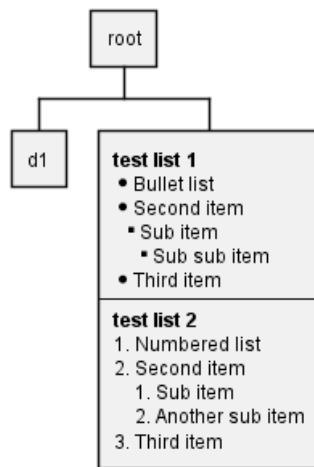
22.14.11 WBS

@startwbs

```
* root
** d1
**:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item;
```

@endwbs





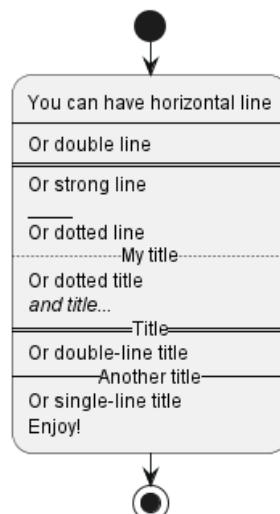
22.15 Appendix: Examples of "Creole horizontal lines" on all diagrams

22.15.1 Activity

TODO: FIXME strong line **----** **TODO:** FIXME

```

@startuml
start
:You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;
stop
@enduml
  
```



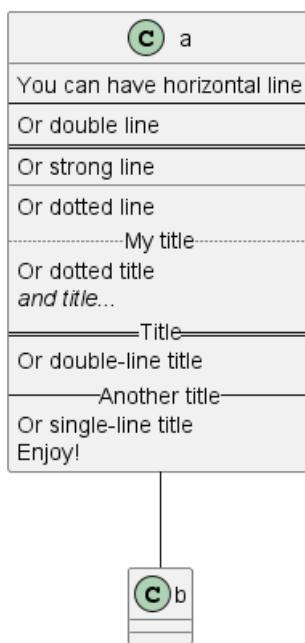
22.15.2 Class

@startuml

```
class a {
You can have horizontal line
-----
Or double line
-----
Or strong line
-----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!
}
```

a -- b

@enduml



22.15.3 Component, Deployment, Use-Case

@startuml

```
node n [
You can have horizontal line
-----
Or double line
-----
Or strong line
-----
Or dotted line
..My title..
```



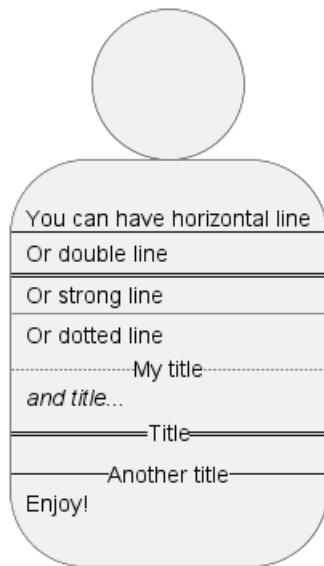
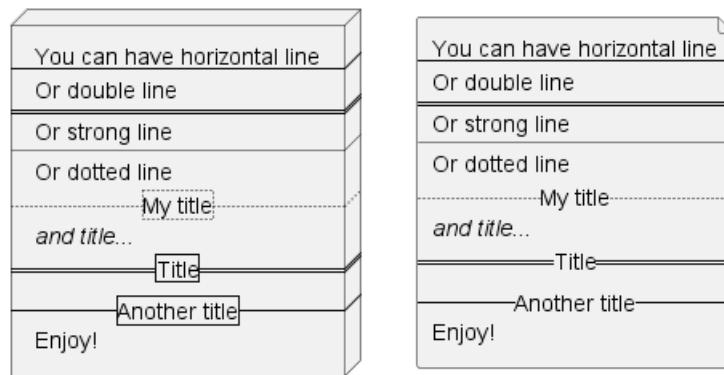
```
//and title... //
==Title==
--Another title--
Enjoy!
]

file f as "
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
"

person p [
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!

]
@enduml
```





22.15.4 Gantt project planning

N/A

22.15.5 Object

```
@startuml
object user {
You can have horizontal line
----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
}
```

```
@enduml
```



user
You can have horizontal line
Or double line
Or strong line
Or dotted line
----- My title -----
and title...
==== Title =====
--- Another title ---
Enjoy!

TODO: DONE [Corrected on V1.2020.18]

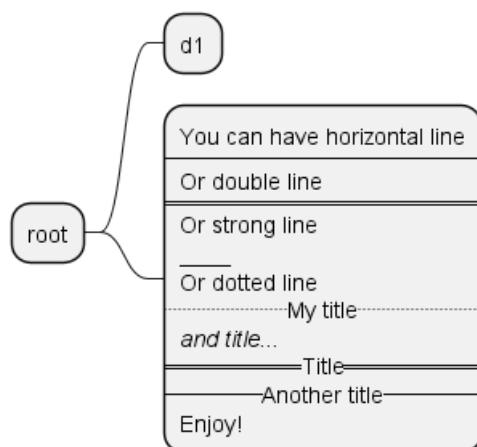
22.15.6 MindMap

TODO: FIXME strong line ---- **TODO:** FIXME

@startmindmap

```
* root
** d1
**:You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title...
//and title... //
==Title==
--Another title--
Enjoy!;
```

@endmindmap

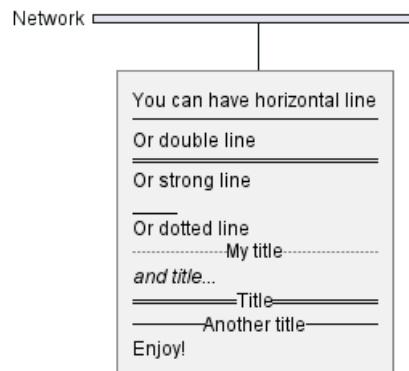


22.15.7 Network (nwdiag)

```
@startuml
nwdiag {
    network Network {
```

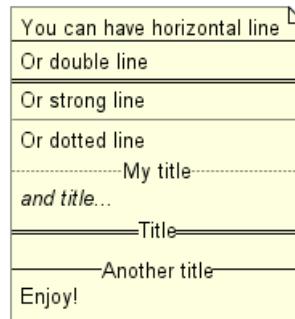


```
Server [description="You can have horizontal line\n----\nOr double line\n====\nOr strong line\n\n}@\n@enduml
```



22.15.8 Note

```
@startuml
note as n
You can have horizontal line
----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
end note
@enduml
```



22.15.9 Sequence

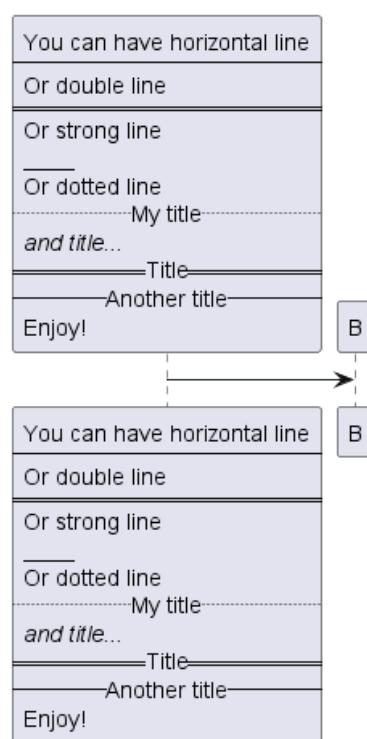
```
@startuml
<style>
participant {HorizontalAlignment left}
</style>
participant Participant [
You can have horizontal line
----
Or double line
```



```
=====
Or strong line
-----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
]

participant B

Participant -> B
@enduml
```

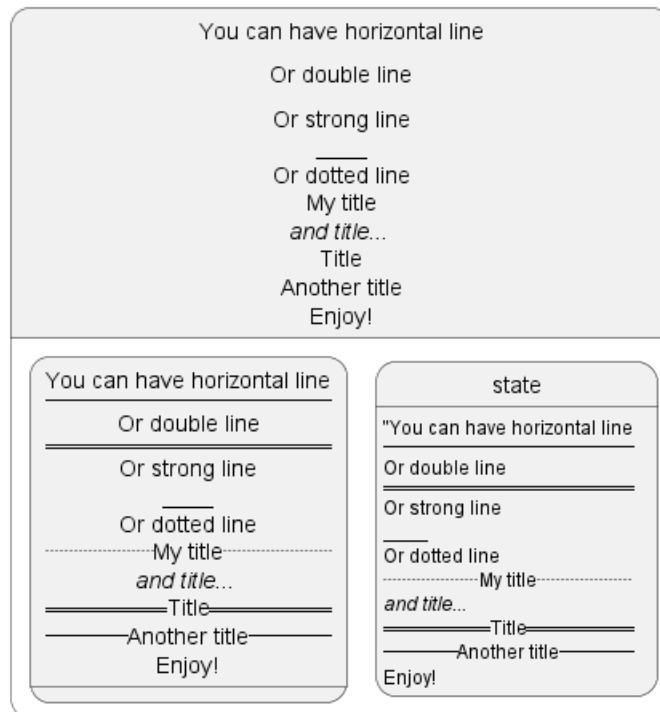


[Ref. QA-15232]

22.15.10 State

```
@startuml
<style>
stateDiagram {
title {HorizontalAlignment left}
}
</style>
state "You can have horizontal line\n----\nOr double line\n=====\\nOr strong line\\n____\\nOr dotted line"
state "You can have horizontal line\n----\nOr double line\n=====\\nOr strong line\\n____\\nOr dotted line"
state : "You can have horizontal line\n----\nOr double line\n=====\\nOr strong line\\n____\\nOr dotted line"
}
@enduml
```





[Ref. QA-16978]

22.15.11 WBS

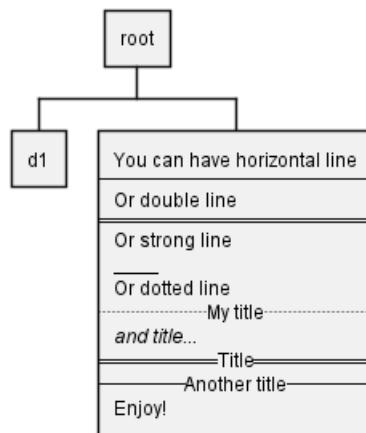
TODO: FIXME strong line ---- **TODO:** FIXME

@startwbs

```
* root
** d1
**:You can have horizontal line
----
Or double line
=====
Or strong line
-----
Or dotted line
..My title...
//and title... //
==Title==
--Another title--
Enjoy!;
```

@endwbs





22.16 Style equivalent (between Creole and HTML)

Style	Creole	Legacy HTML like
bold	This is **bold**	This is bold
<i>italics</i>	This is //italics//	This is <i>italics</i>
<code>monospaced</code>	This is ""monospaced""	This is <font:monospaced>monospaced
stroked	This is --stroked--	This is <s>stroked</s>
<u>underlined</u>	This is __underlined__	This is <u>underlined</u>
waved	This is ~~~	This is <w>waved</w>

```

@startmindmap
* Style equivalent\n(between Creole and HTML)
**:**Creole**
-----
<#silver>|= code|= output
| \n This is ""~**bold**"\n | \n This is **bold** |
| \n This is ""~//italics//"\n | \n This is //italics// |
| \n This is ""~"monospaced~"" "\n | \n This is ""monospaced"" |
| \n This is ""~~stroked~~"\n | \n This is --stroked-- |
| \n This is ""~__underlined__"\n | \n This is __underlined__ |
| \n This is ""~<U+007E><U+007E>waved<U+007E><U+007E>""\n | \n This is ~~waved~~ |;
**:<b>Legacy HTML like
-----
<#silver>|= code|= output
| \n This is ""~<b>bold</b>""\n | \n This is <b>bold</b> |
| \n This is ""~<i>italics</i>""\n | \n This is <i>italics</i> |
| \n This is ""~<font:monospaced>monospaced</font>""\n | \n This is <font:monospaced>monospaced</font> |
| \n This is ""~<s>stroked</s>""\n | \n This is <s>stroked</s> |
| \n This is ""~<u>underlined</u>""\n | \n This is <u>underlined</u> |
| \n This is ""~<w>waved</w>""\n | \n This is <w>waved</w> |

And color as a bonus...
<#silver>|= code|= output
| \n This is ""~<s:=""<color:green>"green"</color>"">stroked</s>""\n | \n This is <s:green>stroked</s> |
| \n This is ""~<u:=""<color:red>"red"</color>"">underlined</u>""\n | \n This is <u:red>underlined</u> |
| \n This is ""~<w:=""<color:#0000FF>"#0000FF"</color>"">waved</w>""\n | \n This is <w:#0000FF>waved</w> |
@endmindmap

```



Creole	
code	output
This is **bold**	This is bold
This is //italics//	This is <i>italics</i>
This is ""monospaced""	This is monospaced
This is --stroked--	This is stroked
This is __underlined__	This is <u>underlined</u>
This is ~~waved~~	This is <u>waved</u>

Legacy HTML like	
code	output
This is bold	This is bold
This is <i>italics</i>	This is <i>italics</i>
This is <font:monospaced>monospaced	This is monospaced
This is <s>stroked</s>	This is stroked
This is <u>underlined</u>	This is <u>underlined</u>
This is <w>waved</w>	This is <u>waved</u>

And color as a bonus...

code	output
This is <s:green>stroked</s>	This is stroked
This is <u:red>underlined</u>	This is <u>underlined</u>
This is <w:#0000FF>waved</w>	This is <u>waved</u>



23 Defining and using sprites

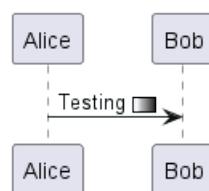
A *Sprite* is a small graphic element that can be used in diagrams.

In PlantUML, sprites are monochrome and can have either 4, 8 or 16 gray level.

To define a sprite, you have to use a hexadecimal digit between 0 and F per pixel.

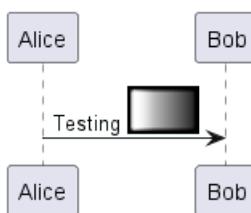
Then you can use the sprite using <\$XXX> where XXX is the name of the sprite.

```
@startuml
sprite $foo1 {
    FFFFFFFFFFFFFF
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    F0123456789ABC
    FFFFFFFFFFFFFF
}
Alice -> Bob : Te
@enduml
```



You can scale the sprite.

```
@startuml
sprite $foo1 {
    FFFFFFFFFFFFFF
    F0123456789ABCF
    FFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1{scale=3}>
@enduml
```

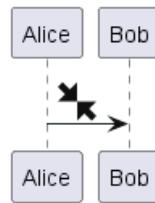


23.1 Inline SVG sprite

You can also use inlined SVG for sprites. Only a tiny subset of SVG directives is possible, so you probably have to compress existing SVG files using <https://vecta.io/nano>.

```
@startuml
sprite foo1 <svg width="8" height="8" viewBox="0 0 8 8"><path d="M1 0l-1 1 1.5 1.5-1.5 1.5h4v-4l-1.5
```

```
Alice->Bob : <$foo1*3>
@enduml
```

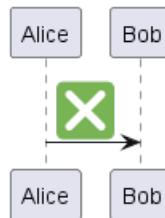


Another example:

```
@startuml
sprite foo1 <svg viewBox="0 0 36 36">
<path fill="#77B255" d="M36 32c0 2.209-1.791 4-4 4H4c-2.209 0-4-1.791-4-4V4c0-2.209 1.791-4 4-4h28c2
<path fill="#FFF" d="M21.529 18.00618.238-8.238c.977-.976.977-2.559 0-3.535-.977-.977-2.559-.977-3.559
</svg>
```

```
Alice->Bob : <$foo1>
```

```
@enduml
```

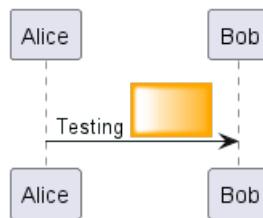


23.2 Changing colors

Although sprites are monochrome, it's possible to change their color.

```
@startuml
sprite $foo1 {
    FFFFFFFFFFFFFF
    F0123456789ABCF
    F0123456789ABCF
    F0123456789ABCF
    F0123456789ABCF
    F0123456789ABCF
    F0123456789ABCF
    F0123456789ABCF
    F0123456789ABCF
    FFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1,scale=3.4,color=orange>
@enduml
```





23.3 Encoding Sprite

To encode sprite, you can use the command line like:

```
java -jar plantuml.jar -encodesprite 16z foo.png
```

where `foo.png` is the image file you want to use (it will be converted to gray automatically).

After `-encodesprite`, you have to specify a format: `4`, `8`, `16`, `4z`, `8z` or `16z`.

The number indicates the gray level and the optional `z` is used to enable compression in sprite definition.

23.4 Importing Sprite

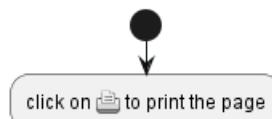
You can also launch the GUI to generate a sprite from an existing image.

Click in the menubar then on **File/Open Sprite Window**.

After copying an image into your clipboard, several possible definitions of the corresponding sprite will be displayed : you will just have to pickup the one you want.

23.5 Examples

```
@startuml
sprite $printer [15x15/8z] N0tH3W0W208HxFz_kMAhj71HWpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwv
start
:click on <$printer> to print the page;
@enduml
```



```
@startuml
sprite $bug [15x15/16z] PKzR2i0m2BFMi15p__FEjQEeqB1z27aeqCqixa8S40T7C53cKpsHpaYPDJY_12MHM-BLRyywPhrr
sprite $printer [15x15/8z] N0tH3W0W208HxFz_kMAhj71HWpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwv
sprite $disk {
    444445566677881
    436000000009991
    436000000000ACA1
    53700000001A7A1
    53700000012B8A1
    53800000123B8A1
    63800001233C9A1
    634999AABC99B1
    744566778899AB1
    7456AAAAA99AAB1
    8566AFC228AABB1
    8567AC8118BBBB1
    867BD4433BBBBB1
    39AAAAABBBBBBC1
}
```

}



```

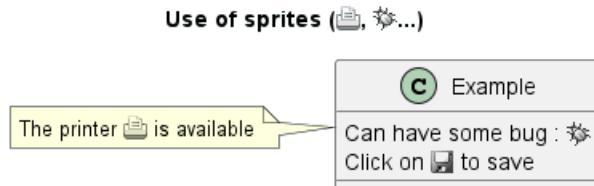
title Use of sprites (<$printer>, <$bug>...)

class Example {
Can have some bug : <$bug>
Click on <$disk> to save
}

note left : The printer <$printer> is available

@enduml

```



23.6 StdLib

The PlantUML StdLib includes a number of ready icons in various IT areas such as architecture, cloud services, logos etc. It including AWS, Azure, Kubernetes, C4, product Logos and many others. To explore these libraries:

- Browse the Github folders of PlantUML StdLib
- Browse the source repos of StdLib collections that interest you. Eg if you are interested in logos you can find that it came from gilbarbara-plantuml-sprites, and quickly find its sprites-list. (The next section shows how to list selected sprites but unfortunately that's in grayscale whereas this custom listing is in color.)
- Study the in-depth Hitchhiker's Guide to PlantUML, eg sections Standard Library Sprites and PlantUML Stdlib Overview

23.7 Listing Sprites

You can use the `listsprites` command to show available sprites:

- Used on its own, it just shows ArchiMate sprites
- If you include some sprite libraries in your diagram, the command shows all these sprites, as explained in View all the icons with `listsprites`.

(Example from Hitchhikers Guide to PlantUML)

```

@startuml
!define osaPuml https://raw.githubusercontent.com/Crashedmind/PlantUML-opensecurityarchitecture2-ico...
!include osaPuml/Common.puml
!include osaPuml/User/all.puml
!include osaPuml/Hardware/all.puml
!include osaPuml/Misc/all.puml
!include osaPuml/Server/all.puml
!include osaPuml/Site/all.puml

```

```
listsprites
```

```
' From The Hitchhiker's Guide to PlantUML
@enduml
```





Most collections have files called `all` that allow you to see a whole sub-collection at once. Else you need to find the sprites that interest you and include them one by one. Unfortunately, the version of a collection included in StdLib often does not have such `all` files, so as you see above we include the collection from github, not from StdLib.

All sprites are in grayscale, but most collections define specific macros that include appropriate (vendor-specific) colors.

24 Skinparam command

You can change colors and font of the drawing using the `skinparam` command.

Example:

```
skinparam backgroundColor transparent
```

24.1 Usage

You can use this command :

- In the diagram definition, like any other commands,
- In an included file,
- In a configuration file, provided in the command line or the ANT task.

24.2 Nested

To avoid repetition, it is possible to nest definition. So the following definition :

```
skinparam xxxxParam1 value1
skinparam xxxxParam2 value2
skinparam xxxxParam3 value3
skinparam xxxxParam4 value4
```

is strictly equivalent to:

```
skinparam xxxx {
    Param1 value1
    Param2 value2
    Param3 value3
    Param4 value4
}
```

24.3 Black and White

You can force the use of a black&white output using `skinparam monochrome true` command.

```
@startuml
```

```
skinparam monochrome true

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

User -> A: DoWork
activate A

A -> B: Create Request
activate B

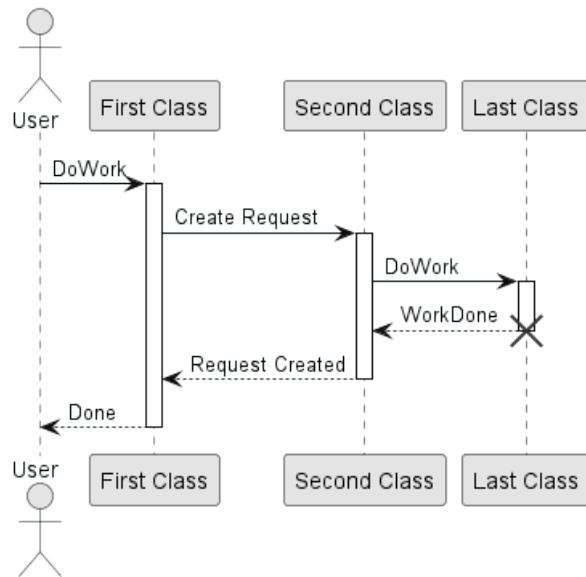
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B
```



```
A --> User: Done
deactivate A

@enduml
```



24.4 Shadowing

You can disable the shadowing using the `skinparam shadowing false` command.

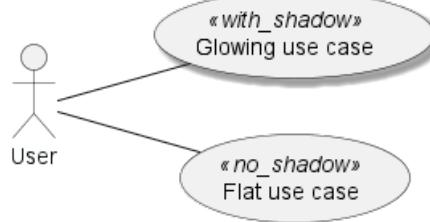
```
@startuml
```

left to right direction

```
skinparam shadowing<<no_shadow>> false
skinparam shadowing<<with_shadow>> true

actor User
(Glowing use case) <<with_shadow>> as guc
(Flat use case) <<no_shadow>> as fuc
User -- guc
User -- fuc
```

```
@enduml
```



24.5 Reverse colors

You can force the use of a black&white output using `skinparam monochrome reverse` command. This can be useful for black background environment.

```
@startuml
```

```
skinparam monochrome reverse
```



```
actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C
```

```
User -> A: DoWork
activate A
```

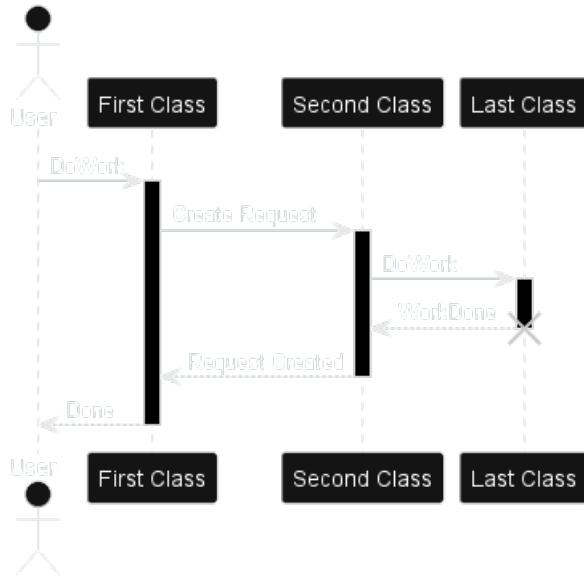
```
A -> B: Create Request
activate B
```

```
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C
```

```
B --> A: Request Created
deactivate B
```

```
A --> User: Done
deactivate A
```

```
@enduml
```



24.6 Colors

You can use either standard color name or RGB code.

```
@startuml
colors
@enduml
```



APPLICATION	Crimson	DeepPink	Indigo	LightYellow	Navy	RoyalBlue	Turquoise
AliceBlue	Cyan	DeepSkyBlue	Ivory	Lime	OldLace	STRATEGY	Violet
AntiqueWhite	DarkBlue	DimGray	Khaki	LimeGreen	Olive	SaddleBrown	Wheat
Aqua	DarkCyan	DimGrey	Lavender	Linen	OliveDrab	Salmon	White
Aquamarine	DarkGoldenRod	DodgerBlue	LavenderBlush	MOTIVATION	Orange	SandyBrown	WhiteSmoke
Azure	DarkGray	FireBrick	LawnGreen	Magenta	OrangeRed	SeaGreen	Yellow
BUSINESS	DarkGreen	FloralWhite	LemonChiffon	Maroon	Orchid	SeaShell	YellowGreen
Beige	DarkGrey	ForestGreen	LightBlue	MediumAquaMarine	PHYSICAL	Sienna	
Bisque	DarkKhaki	Fuchsia	LightCoral	MediumBlue	PaleGoldenRod	Silver	
Black	DarkMagenta	Gainsboro	LightCyan	MediumOrchid	PaleGreen	SkyBlue	
BlanchedAlmond	DarkOliveGreen	GhostWhite	LightGoldenRodYellow	MediumPurple	PaleTurquoise	SlateBlue	
Blue	DarkOrchid	Gold	LightGray	MediumSeaGreen	PaleVioletRed	SlateGray	
BlueViolet	DarkRed	GoldenRod	LightGreen	MediumSlateBlue	PapayaWhip	SlateGrey	
Brown	DarkSalmon	Gray	LightGrey	MediumSpringGreen	PeachPuff	Snow	
BurlyWood	DarkSeaGreen	Green	LightPink	MediumTurquoise	Peru	SpringGreen	
CadetBlue	DarkSlateBlue	GreenYellow	LightSalmon	MediumVioletRed	Pink	SteelBlue	
Chartreuse	DarkSlateGray	Grey	LightSeaGreen	MidnightBlue	Plum	TECHNOLOGY	
Chocolate	DarkSlateGrey	HoneyDew	LightSkyBlue	MintCream	PowderBlue	Tan	
Coral	DarkTurquoise	HotPink	LightSlateGray	MistyRose	Purple	Teal	
CornflowerBlue	DarkViolet	IMPLEMENTATION	LightSlateGrey	Moccasin	Red	Thistle	
Cornsilk	DarkOrange	IndianRed	LightSteelBlue	NavajoWhite	RosyBrown	Tomato	

transparent can only be used for background of the image.

24.7 Font color, name and size

You can change the font for the drawing using `xxxFontColor`, `xxxFontSize` and `xxxFontName` parameters.

Example:

```
skinparam classFontColor red
skinparam classFontSize 10
skinparam classFontName Aapex
```

You can also change the default font for all fonts using `skinparam defaultFontName`.

Example:

```
skinparam defaultFontName Aapex
```

Please note the fontname is highly system dependent, so do not over use it, if you look for portability. `Helvetica` and `Courier` should be available on all systems.

A lot of parameters are available. You can list them using the following command:

```
java -jar plantuml.jar -language
```

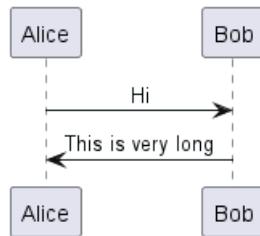
24.8 Text Alignment

Text alignment can be set to `left`, `right` or `center` in `skinparam sequenceMessageAlign`. You can also use `direction` or `reverseDirection` values to align text depending on arrow direction.

Param name	Default value	Comment
sequenceMessageAlign	left	Used for messages in sequence diagrams
sequenceReferenceAlign	center	Used for <code>ref</code> over in sequence diagrams

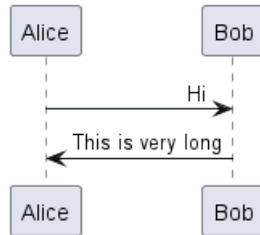
```
@startuml
skinparam sequenceMessageAlign center
Alice -> Bob : Hi
Bob -> Alice : This is very long
@enduml
```





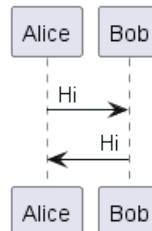
```

@startuml
skinparam sequenceMessageAlign right
Alice -> Bob : Hi
Bob -> Alice : This is very long
@enduml
  
```



```

@startuml
skinparam sequenceMessageAlign direction
Alice -> Bob : Hi
Bob -> Alice: Hi
@enduml
  
```



24.9 Examples

```

@startuml
skinparam backgroundColor #EEEBCD
skinparam handwritten true

skinparam sequence {
ArrowColor DeepSkyBlue
ActorBorderColor DeepSkyBlue
LifeLineBorderColor blue
LifeLineBackgroundColor #A9DCDF

ParticipantBorderColor DeepSkyBlue
ParticipantBackgroundColor DodgerBlue
ParticipantFontName Impact
ParticipantFontSize 17
ParticipantFontColor #A9DCDF

ActorBackgroundColor aqua
ActorFontColor DeepSkyBlue
ActorFontSize 17
ActorFontName Aapex
  
```



```

}

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

User -> A: DoWork
activate A

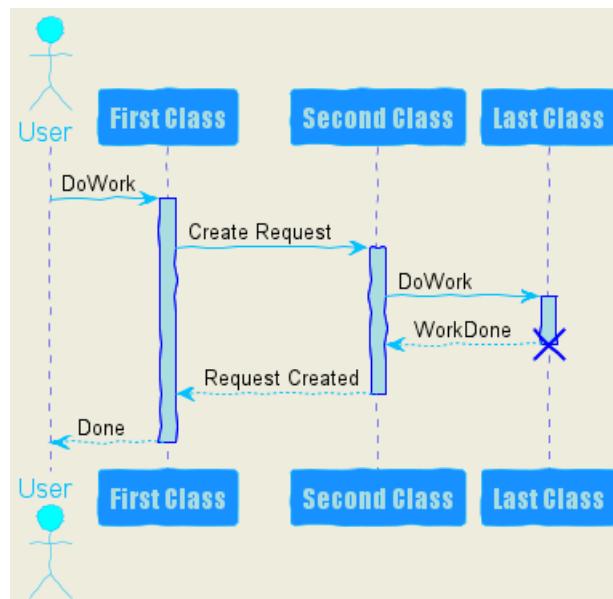
A -> B: Create Request
activate B

B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A
@enduml

```



```

@startuml
skinparam handwritten true

skinparam actor {
BorderColor black
FontName Courier
    BackgroundColor<< Human >> Gold
}

skinparam usecase {
BackgroundColor DarkSeaGreen
BorderColor DarkSlateGray

BackgroundColor<< Main >> YellowGreen

```



```

BorderColor<< Main >> YellowGreen

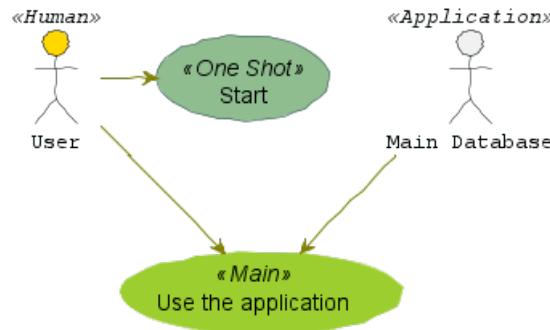
ArrowColor Olive
}

User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>

User -> (Start)
User --> (Use)

MySql --> (Use)
@enduml

```



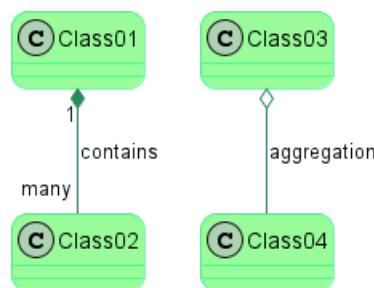
```

@startuml
skinparam roundcorner 20
skinparam class {
BackgroundColor PaleGreen
ArrowColor SeaGreen
BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen

Class01 "1" *-- "many" Class02 : contains

Class03 o-- Class04 : aggregation
@enduml

```



```

@startuml
skinparam interface {
  backgroundColor RosyBrown
  borderColor orange
}

skinparam component {
  FontSize 13

```



```

BackgroundColor<<Apache>> LightCoral
BorderColor<<Apache>> #FF6655
FontName Courier
BorderColor black
BackgroundColor gold
ArrowFontName Impact
ArrowColor #FF6655
ArrowFontColor #777777
}

```

```

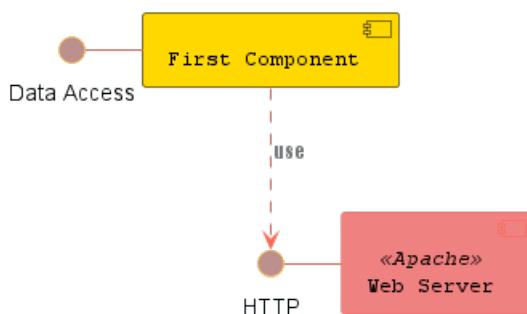
() "Data Access" as DA
[Web Server] << Apache >>

```

```

DA - [First Component]
[First Component] ..> () HTTP : use
HTTP - [Web Server]
@enduml

```



```

@startuml
[AA] <<static lib>>
[BB] <<shared lib>>
[CC] <<static lib>>

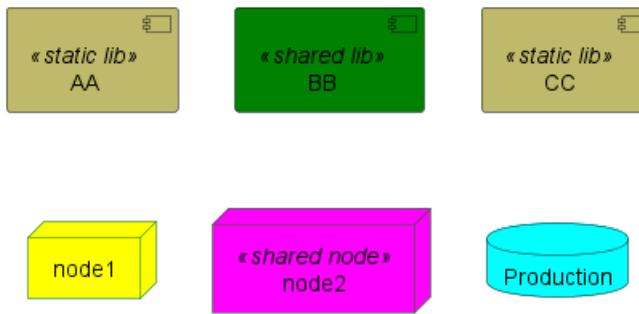
node node1
node node2 <<shared node>>
database Production

skinparam component {
    backgroundColor<<static lib>> DarkKhaki
    backgroundColor<<shared lib>> Green
}

skinparam node {
borderColor Green
backgroundColor Yellow
backgroundColor<<shared node>> Magenta
}
skinparam databaseBackgroundColor Aqua
@enduml

```





24.10 List of all skinparam parameters

You can use `-language` on the command line or generate a "diagram" with a list of all the skinparam parameters using :

- `help skinparams`
- `skinparameters`

24.10.1 Command Line: `-language` command

Since the documentation is not always up to date, you can have the complete list of parameters using this command:

```
java -jar plantuml.jar -language
```

24.10.2 Command: `help skinparams`

That will give you the following result, from this page (*code of this command*): `CommandHelpSkinparam.java`

```
@startuml
help skinparams
@enduml
```

Welcome to PlantUML!

You can start with a simple UML Diagram like:



`Bob->Alice: Hello`

Or

`class Example`

You will find more information about PlantUML syntax on <https://plantuml.com>

(Details by typing `license` keyword)

```
PlantUML 1.2023.11
[From string (line 2) ]

@startuml
help skinparams
Syntax Error?
```

24.10.3 Command: `skinparameters`

```
@startuml
skinparameters
@enduml
```



ActivityBackgroundColor	ClassFontStyle	FolderStereotypeFontSize	NoteFontStyle	SequenceDelayFontName
ActivityBorderColor	ClassStereotypeFontColor	FolderStereotypeFontStyle	NoteShadowing	SequenceDelayFontSize
ActivityBorderThickness	ClassStereotypeFontName	FooterFontColor	NoteTextAlignment	SequenceDelayFontStyle
ActivityDiamondFontColor	ClassStereotypeFontSize	FooterFontName	ObjectAttributeFontColor	SequenceDividerBorderThickness
ActivityDiamondFontName	ClassStereotypeFontStyle	FooterFontSize	ObjectAttributeFontName	SequenceDividerFontColor
ActivityDiamondFontSize	CloudFontColor	FooterFontStyle	ObjectAttributeFontSize	SequenceDividerFontName
ActivityDiamondFontStyle	CloudFontName	FrameFontColor	ObjectAttributeFontStyle	SequenceDividerFontSize
ActivityFontColor	CloudFontSize	FrameFontName	ObjectBorderThickness	SequenceDividerFontStyle
ActivityFontName	CloudFontStyle	FrameFontSize	ObjectFontColor	SequenceGroupBodyBackgroundColor
ActivityFontSize	CloudStereotypeFontColor	FrameFontStyle	ObjectFontName	SequenceGroupBorderThickness
ActivityFontStyle	CloudStereotypeFontName	FrameStereotypeFontColor	ObjectFontSize	SequenceGroupFontColor
ActorBackgroundColor	CloudStereotypeFontSize	FrameStereotypeFontName	ObjectFontStyle	SequenceGroupFontName
ActorBorderColor	CloudStereotypeFontStyle	FrameStereotypeFontSize	ObjectStereotypeFontColor	SequenceGroupFontSize
ActorFontColor	ColorArrowSeparationSpace	FrameStereotypeFontStyle	ObjectStereotypeFontName	SequenceGroupFontStyle
ActorFontName	ComponentBorderThickness	GenericDisplay	ObjectStereotypeFontSize	SequenceGroupHeaderFontColor
ActorFontSize	ComponentFontColor	Guillemet	ObjectStereotypeFontStyle	SequenceGroupHeaderFontName
ActorFontStyle	ComponentFontName	Handwritten	PackageBorderThickness	SequenceGroupHeaderFontSize
ActorStereotypeFontColor	ComponentFontSize	HeaderFontColor	PackageFontColor	SequenceGroupHeaderFontStyle
ActorStereotypeFontName	ComponentFontStyle	HeaderFontName	PackageFontName	SequenceHeaderLineBorderColor
ActorStereotypeFontSize	ComponentStereotypeFontColor	HeaderFontSize	PackageFontSize	SequenceHeaderLineBorderThickness
ActorStereotypeFontStyle	ComponentStereotypeFontName	HeaderFontStyle	PackageFontStyle	SequenceMessageAlignment
AgentBorderThickness	ComponentStereotypeFontSize	HexagonBorderThickness	PackageStereotypeFontColor	SequenceNewpageSeparatorColor
AgentFontColor	ComponentStereotypeFontStyle	HexagonFontColor	PackageStereotypeFontSize	SequenceParticipant
AgentFontName	ComponentStyle	HexagonFontName	PackageStereotypeFontStyle	SequenceParticipantBorderThickness
AgentFontSize	ConditionEndStyle	HexagonFontSize	PackageStyle	SequenceReferenceAlignment
AgentFontStyle	ConditionStyle	HexagonFontStyle	PackageTitleAlignment	SequenceReferenceBackgroundColor
AgentStereotypeFontColor	ControlFontColor	HexagonStereotypeFontColor	Padding	SequenceReferenceBorderThickness
AgentStereotypeFontName	ControlFontName	HexagonStereotypeFontName	PageBorderColor	SequenceReferenceFontColor
AgentStereotypeFontSize	ControlFontSize	HexagonStereotypeFontSize	PageExternalColor	SequenceReferenceFontName
AgentStereotypeFontStyle	ControlFontStyle	HexagonStereotypeFontStyle	PageMargin	SequenceReferenceFontSize
ArchimateBorderThickness	ControlStereotypeFontColor	HyperlinkColor	ParticipantFontColor	SequenceReferenceFontStyle
ArchimateFontColor	ControlStereotypeFontName	HyperlinkUnderline	ParticipantFontName	SequenceReferenceHeaderBackgroundColor
ArchimateFontName	ControlStereotypeFontSize	IconEMandatoryColor	ParticipantFontSize	SequenceStereotypeFontColor
ArchimateFontSize	ControlStereotypeFontStyle	IconPackageBackgroundColor	ParticipantFontStyle	SequenceStereotypeFontName
ArchimateFontStyle	DatabaseFontColor	IconPackageColor	ParticipantPadding	SequenceStereotypeFontSize
ArchimateStereotypeFontColor	DatabaseFontName	IconPrivateBackgroundColor	ParticipantStereotypeFontColor	Shadowing
ArchimateStereotypeFontName	DatabaseFontSize	IconPrivateColor	ParticipantStereotypeFontName	StackFontColor
ArchimateStereotypeFontSize	DatabaseFontStyle	IconProtectedBackgroundColor	ParticipantStereotypeFontSize	StackFontName
ArchimateStereotypeFontStyle	DatabaseStereotypeFontColor	IconProtectedColor	ParticipantStereotypeFontStyle	StackFontPointSize
ArrowFontColor	DatabaseStereotypeFontName	IconPublicBackgroundColor	PartitionBorderThickness	StackFontSize
ArrowFontName	DatabaseStereotypeFontSize	IconPublicColor	PartitionFontColor	StackFontStyle
ArrowFontSize	DatabaseStereotypeFontStyle	InterfaceFontColor	PartitionFontName	StackStereotypeFontColor
ArrowFontStyle	DefaultFontColor	InterfaceFontName	PartitionFontSize	StackStereotypeFontName
ArrowHeadColor	DefaultFontName	InterfaceFontSize	PartitionFontStyle	StackStereotypeFontSize
ArrowLollipopColor	DefaultFontSize	InterfaceFontStyle	PathHoverColor	StackStereotypeFontStyle
ArrowMessageAlignment	DefaultFontStyle	InterfaceStereotypeFontColor	PersonBorderThickness	StateAttributeFontColor
ArrowThickness	DefaultMonospacedFontName	InterfaceStereotypeFontName	PersonFontColor	StateAttributeFontName
ArtifactFontColor	DefaultTextAlignment	InterfaceStereotypeFontSize	PersonFontName	StateAttributeFontSize
ArtifactFontName	DesignedBackgroundColor	InterfaceStereotypeFontStyle	PersonFontSize	StateAttributeFontStyle
ArtifactFontSize	DesignedBorderColor	LabelFontColor	PersonFontStyle	StateBorderColor
ArtifactFontStyle	DesignedDomainBorderThickness	LabelFontName	PersonStereotypeFontColor	StateFontColor
ArtifactStereotypeFontColor	DesignedDomainFontColor	LabelFontSize	PersonStereotypeFontName	StateFontName
ArtifactStereotypeFontName	DesignedDomainFontName	LabelFontStyle	PersonStereotypeFontSize	StateFontSize
ArtifactStereotypeFontSize	DesignedDomainFontSize	LabelStereotypeFontColor	PersonStereotypeFontStyle	StateFontStyle
ArtifactStereotypeFontStyle	DesignedDomainFontStyle	LabelStereotypeFontName	QueueBorderThickness	StateMessageAlignment
BackgroundColor	DesignedDomainStereotypeFontColor	LabelStereotypeFontSize	QueueColor	StereotypePosition
BiddableBackgroundColor	DesignedDomainStereotypeFontName	LabelStereotypeFontStyle	QueueFontColor	StorageFontColor
BiddableBorderColor	DesignedDomainStereotypeFontSize	LegendBorderThickness	QueueFontName	StorageFontName
BoundaryFontColor	DesignedDomainStereotypeFontStyle	LegendFontColor	QueueFontSize	StorageFontSize
BoundaryFontName	DiagramBorderColor	LegendFontName	QueueFontStyle	StorageFontStyle
BoundaryFontSize	DiagramBorderThickness	LegendFontSize	QueueStereotypeFontColor	StorageStereotypeFontColor
BoundaryFontStyle	DomainBackgroundColor	LegendFontStyle	QueueStereotypeFontName	StorageStereotypeFontName
BoundaryStereotypeFontColor	DomainBorderColor	LexicalBackgroundColor	QueueStereotypeFontSize	StorageStereotypeFontSize
BoundaryStereotypeFontName	DomainBorderThickness	LexicalBorderColor	QueueStereotypeFontStyle	StorageStereotypeFontStyle
BoundaryStereotypeFontSize	DomainFontColor	LifelineStrategy	Ranksep	StorageStereotypeFontSize
BoundaryStereotypeFontStyle	DomainFontName	Linetype	RectangleBorderThickness	Style
BoxPadding	DomainFontSize	MachineBackgroundColor	RectangleFontColor	SvglinkTarget
CaptionFontColor	DomainFontStyle	MachineBorderColor	RectangleFontName	SwimlaneBorderThickness
CaptionFontName	DomainStereotypeFontColor	MachineBorderThickness	RectangleFontSize	SwimlaneTitleFontColor
CaptionFontSize	DomainStereotypeFontName	MachineFontColor	RectangleFontStyle	SwimlaneTitleFontName
CaptionFontStyle	DomainStereotypeFontSize	MachineFontName	RectangleStereotypeFontColor	SwimlaneTitleFontSize
CardBorderThickness	DomainStereotypeFontStyle	MachineFontSize	RectangleStereotypeFontName	SwimlaneTitleFontStyle
CardFontColor	Dpi	MachineFontStyle	RectangleStereotypeFontSize	SwimlaneWidth
CardFontName	EntityFontColor	MachineStereotypeFontColor	RequirementBackgroundColor	TabSize
CardFontSize	EntityFontName	MachineStereotypeFontName	RequirementBorderColor	TimingFontColor
CardFontStyle	EntityFontSize	MachineStereotypeFontSize	RequirementBorderThickness	TimingFontSize
CardStereotypeFontColor	EntityFontStyle	MachineStereotypeFontStyle	RequirementFontColor	TimingFontStyle
CardStereotypeFontName	EntityStereotypeFontColor	MaxAsciiMessageLength	RequirementFontName	TitleBorderRoundCorner
CardStereotypeFontSize	EntityStereotypeFontName	MaxMessageSize	RequirementFontSize	TitleBorderThickness
CardStereotypeFontStyle	EntityStereotypeFontSize	MinClassWidth	RequirementFontStyle	TitleFontColor
CircledCharacterFontColor	EntityStereotypeFontStyle	Monochrome	RequirementStereotypeFontColor	TitleFontName
CircledCharacterFontName	FileFontColor	NodeFontColor	RequirementStereotypeFontName	TitleFontSize
CircledCharacterFontSize	FileFontName	NodeFontName	RequirementStereotypeFontSize	TitleFontStyle
CircledCharacterFontStyle	FileFontSize	NodeFontSize	RequirementStereotypeFontStyle	UsecaseBorderThickness
CircledCharacterRadius	FileFontStyle	NodeFontStyle	RequirementStereotypeFontSize	UsecaseFontColor
ClassAttributeFontColor	FileStereotypeFontColor	NodeStereotypeFontColor	ResponseMessageBelowArrow	UsecaseFontName
ClassAttributeFontName	FileStereotypeFontName	NodeStereotypeFontName	RoundCorner	UsecaseFontSize
ClassAttributeFontSize	FileStereotypeFontSize	NodeStereotypeFontSize	SameClassWidth	UsecaseFontStyle
ClassAttributeIconSize	FileStereotypeFontStyle	NodeStereotypeFontStyle	SequenceActorBorderThickness	UsecaseStereotypeFontColor
ClassBackgroundColor	FixCircleLabelOverlapping	Nodesep	SequenceArrowThickness	UsecaseStereotypeFontName
ClassBorderColor	FolderFontColor	NoteBackgroundColor	SequenceBoxBorderColor	UsecaseStereotypeFontSize
ClassBorderThickness	FolderFontName	NoteBorderColor	SequenceBoxFontColor	UsecaseStereotypeFontStyle
ClassFontColor	FolderFontSize	NoteBorderThickness	SequenceBoxFontName	UsecaseStereotypeFontSize
ClassFontName	FolderFontStyle	NoteFontColor	SequenceBoxFontSize	UsecaseStereotypeFontStyle
ClassFontSize	FolderStereotypeFontColor	NoteFontName	SequenceBoxFontStyle	WrapWidth
ClassFontSize	FolderStereotypeFontName	NoteFontSize	SequenceDelayFontColor	



24.10.4 All Skin Parameters on the Ashley's PlantUML Doc

You can also view each skinparam parameters with its results displayed at the page [All Skin Parameters of Ashley's PlantUML Doc](#):

- <https://plantuml-documentation.readthedocs.io/en/latest/formatting/all-skin-params.html>.



25 Preprocessing

Some preprocessing capabilities are included in **PlantUML**, and available for *all* diagrams.

Those functionalities are very similar to the C language preprocessor, except that the special character # has been changed to the exclamation mark !.

25.1 Variable definition [=, ?=]

Although this is not mandatory, we highly suggest that variable names start with a \$.

There are three types of data:

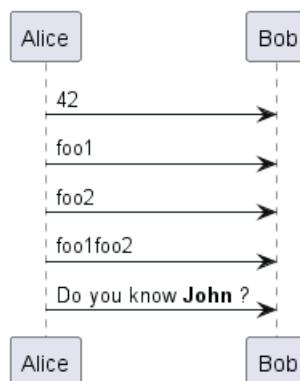
- **Integer number** (*int*);
- **String** (*str*) - these must be surrounded by single quote or double quote;
- **JSON** (*JSON*) - these must be surrounded by curly brackets.

(*for JSON variable definition and usage, see more details on Preprocessing-JSON page*)

Variables created outside function are **global**, that is you can access them from everywhere (including from functions). You can emphasize this by using the optional **global** keyword when defining a variable.

```
@startuml
!$a = 42
!$ab = "foo1"
!$cd = "foo2"
!$ef = $ab + $cd
!$foo = { "name": "John", "age" : 30 }
```

```
Alice -> Bob : $a
Alice -> Bob : $ab
Alice -> Bob : $cd
Alice -> Bob : $ef
Alice -> Bob : Do you know **$foo.name** ?
@enduml
```



You can also assign a value to a variable, only if it is not already defined, with the syntax: !\$a ?= "foo"

```
@startuml
Alice -> Bob : 1. **$name** should be empty

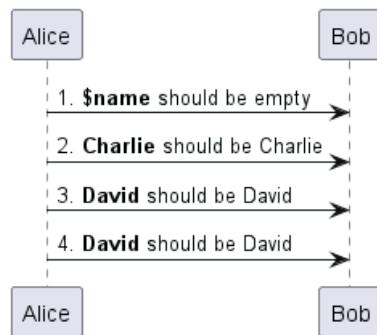
$name ?= "Charlie"
Alice -> Bob : 2. **$name** should be Charlie

$name = "David"
Alice -> Bob : 3. **$name** should be David

$name ?= "Ethan"
Alice -> Bob : 4. **$name** should be David
```



```
@enduml
```



25.2 Boolean expression

25.2.1 Boolean representation [0 is false]

There is not real boolean type, but PlantUML use this integer convention:

- Integer 0 means **false**
- and any non-null number (as 1) or any string (as "1", or even "0") means **true**.

[Ref. QA-9702]

25.2.2 Boolean operation and operator [&&, ||, ()]

You can use boolean expression, in the test, with :

- *parenthesis ()*;
- *and operator &&*;
- *or operator ||*.

(See next example, within *if* test.)

25.2.3 Boolean builtin functions [%false(), %true(), %not(<exp>)]

For convenience, you can use those boolean builtin functions:

- `%false()`
- `%true()`
- `%not(<exp>)`

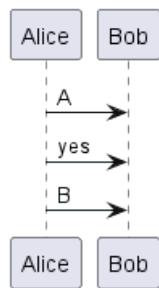
[See also *Builtin functions*]

25.3 Conditions [*!if, !else, !elseif, !endif*]

- You can use expression in condition.
- *else* and *elseif* are also implemented

```
@startuml
!$a = 10
!$ijk = "foo"
Alice -> Bob : A
!if ($ijk == "foo") && ($a+10>=4)
Alice -> Bob : yes
!else
Alice -> Bob : This should not appear
!endif
Alice -> Bob : B
@enduml
```





25.4 While loop [!while, !endwhile]

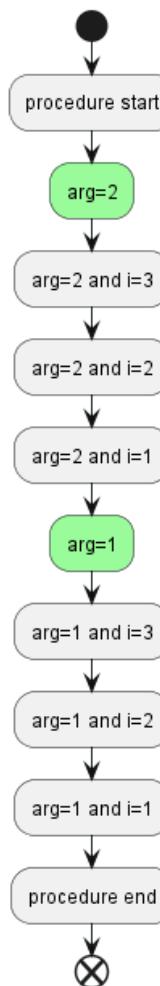
You can use `!while` and `!endwhile` keywords to have repeat loops.

25.4.1 While loop (on Activity diagram)

```
@startuml
!procedure $foo($arg)
:procedure start;
!while $arg!=0
    !$i=3
    #palegreen:arg=$arg;
    !while $i!=0
        :arg=$arg and i=$i;
        !$i = $i - 1
    !endwhile
    !$arg = $arg - 1
    !endwhile
:procedure end;
!endprocedure

start
$foo(2)
end
@enduml
```





[Adapted from QA-10838]

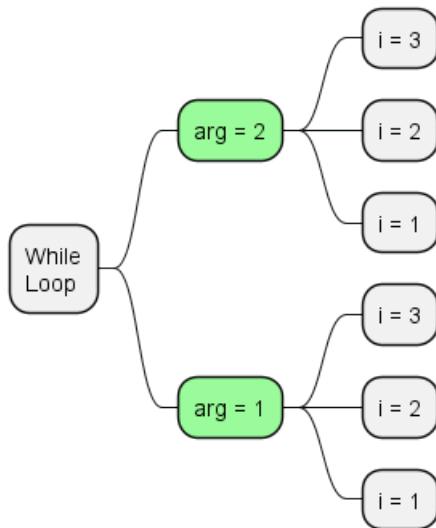
25.4.2 While loop (on Mindmap diagram)

```

@startmindmap
!procedure $foo($arg)
  !$arg!=0
  !$i=3
  **[#palegreen] arg = $arg
  !while $i!=0
    *** i = $i
    !$i = $i - 1
  !endwhile
  !$arg = $arg - 1
!endwhile
!endprocedure

*:While
Loop;
$foo(2)
@endmindmap
  
```





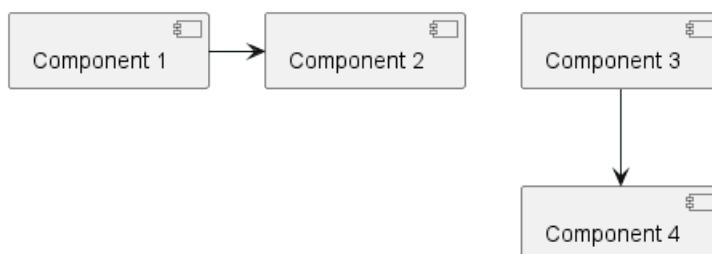
25.4.3 While loop (on Component/Deployment diagram)

```

@startuml
!procedure $foo($arg)
  !while $arg!=0
    [Component $arg] as $arg
    !$arg = $arg - 1
  !endwhile
!endprocedure

$foo(4)

1->2
3-->4
@enduml
  
```



[Ref. QA-14088]

25.5 Procedure [!procedure, !endprocedure]

- Procedure names *should* start with a \$
- Argument names *should* start with a \$
- Procedures can call other procedures

Example:

```

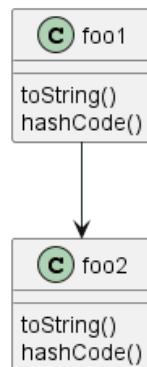
@startuml
!procedure $msg($source, $destination)
  $source --> $destination
!endprocedure
  
```



```
!procedure $init_class($name)
  class $name {
    $addCommonMethod()
  }
!endprocedure
```

```
!procedure $addCommonMethod()
  toString()
  hashCode()
!endprocedure
```

```
$init_class("foo1")
$init_class("foo2")
$msg("foo1", "foo2")
@enduml
```



Variables defined in procedures are **local**. It means that the variable is destroyed when the procedure ends.

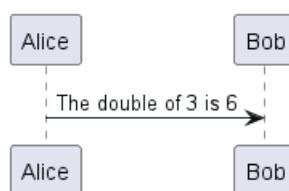
25.6 Return function [!function, !endfunction]

A return function does not output any text. It just define a function that you can call:

- directly in variable definition or in diagram text
- from other return functions
- from procedures
- Function name *should* start with a \$
- Argument names *should* start with a \$

```
@startuml
!function $double($a)
!return $a + $a
!endfunction
```

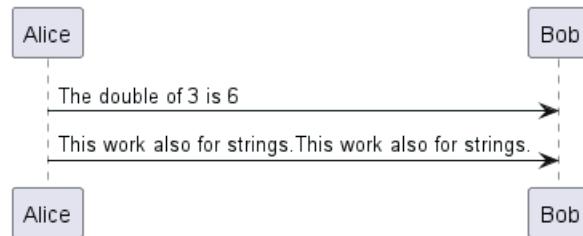
```
Alice -> Bob : The double of 3 is $double(3)
@enduml
```



It is possible to shorten simple function definition in one line:

```
@startuml
!function $double($a) !return $a + $a

Alice -> Bob : The double of 3 is $double(3)
Alice -> Bob : $double("This work also for strings.")
@enduml
```

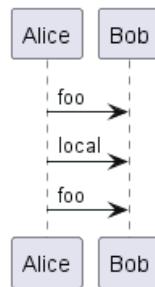


As in procedure (void function), variable are local by default (they are destroyed when the function is exited). However, you can access to global variables from function. However, you can use the `local` keyword to create a local variable if ever a global variable exists with the same name.

```
@startuml
!function $dummy()
!local $ijk = "local"
!return "Alice -> Bob : " + $ijk
!endfunction

!global $ijk = "foo"

Alice -> Bob : $ijk
$dummy()
Alice -> Bob : $ijk
@enduml
```



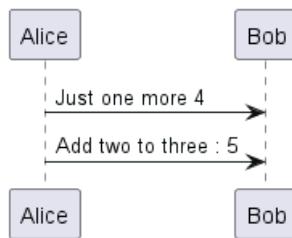
25.7 Default argument value

In both procedure and return functions, you can define default values for arguments.

```
@startuml
!function $inc($value, $step=1)
!return $value + $step
!endfunction

Alice -> Bob : Just one more $inc(3)
Alice -> Bob : Add two to three : $inc(3, 2)
@enduml
```



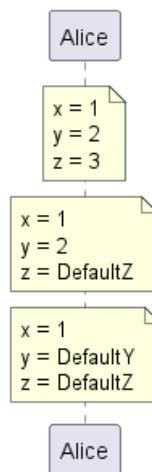


Only arguments at the end of the parameter list can have default values.

```

@startuml
!procedure defaulttest($x, $y="DefaultY", $z="DefaultZ")
note over Alice
  x = $x
  y = $y
  z = $z
end note
!endprocedure

defaulttest(1, 2, 3)
defaulttest(1, 2)
defaulttest(1)
@enduml
  
```



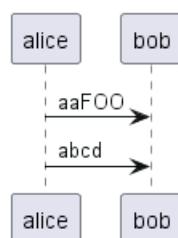
25.8 Unquoted procedure or function [!unquoted]

By default, you have to put quotes when you call a function or a procedure. It is possible to use the `unquoted` keyword to indicate that a function or a procedure does not require quotes for its arguments.

```

@startuml
!unquoted function id($text1, $text2="FOO") !return $text1 + $text2

alice -> bob : id(aa)
alice -> bob : id(ab,cd)
@enduml
  
```



25.9 Keywords arguments

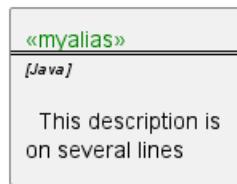
Like in Python, you can use keywords arguments :

```
@startuml
```

```
!unquoted procedure $element($alias, $description="", $label="", $technology="", $size=12, $colour="")
rectangle $alias as "
<color:$colour><$alias></color>
==$label==
//<size:$size>[$technology]</size>//

$description"
!endprocedure

$element(myalias, "This description is %newline()on several lines", $size=10, $technology="Java")
@enduml
```



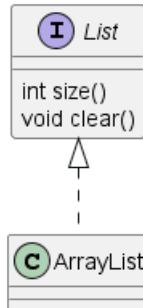
25.10 Including files or URL [!include, !include_many, !include_once]

Use the `!include` directive to include file in your diagram. Using URL, you can also include file from Internet/Intranet. Protected Internet resources can also be accessed, this is described in URL authentication.

Imagine you have the very same class that appears in many diagrams. Instead of duplicating the description of this class, you can define a file that contains the description.

```
@startuml
```

```
interface List
List : int size()
List : void clear()
List <|.. ArrayList
@enduml
```



File List.iuml

```
interface List
List : int size()
List : void clear()
```

The file `List.iuml` can be included in many diagrams, and any modification in this file will change all diagrams that include it.



You can also put several @startuml/@enduml text block in an included file and then specify which block you want to include adding !0 where 0 is the block number. The !0 notation denotes the first diagram.

For example, if you use !include foo.txt!1, the second @startuml/@enduml block within foo.txt will be included.

You can also put an id to some @startuml/@enduml text block in an included file using @startuml(id=MY OWN_ID) syntax and then include the block adding !MY OWN_ID when including the file, so using something like !include foo.txt!MY OWN_ID.

By default, a file can only be included once. You can use !include_many instead of !include if you want to include some file several times. Note that there is also a !include_once directive that raises an error if a file is included several times.

25.11 Including Subpart [!startsub, !endsub, !includesub]

You can also use !startsub NAME and !endsub to indicate sections of text to include from other files using !includesub. For example:

file1.puml:

```
@startuml

A -> A : stuff1
!startsub BASIC
B -> B : stuff2
!endsub
C -> C : stuff3
!startsub BASIC
D -> D : stuff4
!endsub
@enduml
```

file1.puml would be rendered exactly as if it were:

@startuml

```
A -> A : stuff1
B -> B : stuff2
C -> C : stuff3
D -> D : stuff4
@enduml
```

However, this would also allow you to have another file2.puml like this:

file2.puml

@startuml

```
title this contains only B and D
!includesub file1.puml!BASIC
@enduml
```

This file would be rendered exactly as if:

@startuml

```
title this contains only B and D
B -> B : stuff2
D -> D : stuff4
@enduml
```

25.12 Builtin functions [%]

Some functions are defined by default. Their name starts by %



Name	Description
%chr	Return a character from a give Unicode value
%darken	Return a darken color of a given color with some ratio
%date	Retrieve current date. You can provide an optional format for the date You can provide another optional time (on epoch format)
%dec2hex	Return the hexadecimal string (String) of a decimal value (Int)
%dirpath	Retrieve current dirpath
%feature	Check if some feature is available in the current PlantUML running version
%false	Return always false
%file_exists	Check if a file exists on the local filesystem
%filename	Retrieve current filename
%function_exists	Check if a function exists
%get_variable_value	Retrieve some variable value
%getenv	Retrieve environment variable value
%hex2dec	Return the decimal value (Int) of a hexadecimal string (String)
%hsl_color	Return the RGBa color from a HSL color %hsl_color(h, s, l) or %hsl_color(h, s, l, a)
%intval	Convert a String to Int
%is_dark	Check if a color is a dark one
%is_light	Check if a color is a light one
%lighten	Return a lighten color of a given color with some ratio
%load_json	Load JSON data from local file or external URL
%lower	Return a lowercase string
%newline	Return a newline
%not	Return the logical negation of an expression
%now	Return the current epoch time
%ord	Return a Unicode value from a given character
%lighten	Return a lighten color of a given color with some ratio
%reverse_color	Reverse a color using RGB
%reverse_hsluv_color	Reverse a color using HSLuv
%set_variable_value	Set a global variable
%size	Return the size of any string or JSON structure
%string	Convert an expression to String
%strlen	Calculate the length of a String
%strpos	Search a substring in a string
%substr	Extract a substring. Takes 2 or 3 arguments
%true	Return always true
%upper	Return an uppercase string
%variable_exists	Check if a variable exists
%version	Return PlantUML current version

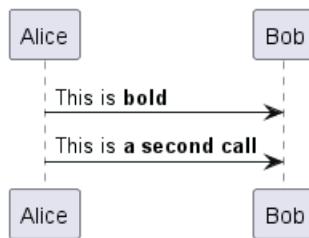
25.13 Logging [!log]

You can use `!log` to add some log output when generating the diagram. This has no impact at all on the diagram itself. However, those logs are printed in the command line's output stream. This could be useful for debug purpose.

```
@startuml
!function bold($text)
 !$result = "<b>" + $text +"</b>"
 !log Calling bold function with $text. The result is $result
 !return $result
!endfunction

Alice -> Bob : This is bold("bold")
Alice -> Bob : This is bold("a second call")
@enduml
```





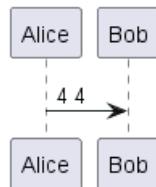
25.14 Memory dump [!dump_memory]

You can use `!dump_memory` to dump the full content of the memory when generating the diagram. An optional string can be put after `!dump_memory`. This has no impact at all on the diagram itself. This could be useful for debug purpose.

```

@startuml
!function $inc($string)
!$val = %intval($string)
!log value is $val
!dump_memory
!return $val+1
!endfunction

Alice -> Bob : 4 $inc("3")
!unused = "foo"
!dump_memory EOF
@enduml
  
```



25.15 Assertion [!assert]

You can put assertions in your diagram.

```

@startuml
Alice -> Bob : Hello
!assert %strpos("abcdef", "cd")==3 : "This always fails"
@enduml
  
```



Welcome to PlantUML!

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on <https://plantuml.com>

(Details by typing `license` keyword)



```
PlantUML 1.2023.11
[From string (line 3) ]

@startuml
Alice -> Bob : Hello
!assert %strpos("abcdef", "cd")==3 : "This always fails"
Assertion error : This always fails
```

25.16 Building custom library [`!import`, `!include`]

It's possible to package a set of included files into a single .zip or .jar archive. This single zip/jar can then be imported into your diagram using `!import` directive.

Once the library has been imported, you can `!include` file from this single zip/jar.

Example:

```
@startuml
```

```
!import /path/to/customLibrary.zip
' This just adds "customLibrary.zip" in the search path

!include myFolder/myFile.iuml
' Assuming that myFolder/myFile.iuml is located somewhere
' either inside "customLibrary.zip" or on the local filesystem
```

```
...
```

25.17 Search path

You can specify the java property `plantuml.include.path` in the command line.

For example:

```
java -Dplantuml.include.path="c:/mydir" -jar plantuml.jar atest1.txt
```

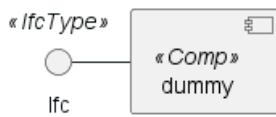
Note the this `-D` option has to put before the `-jar` option. `-D` options after the `-jar` option will be used to define constants within plantuml preprocessor.

25.18 Argument concatenation [`##`]

It is possible to append text to a macro argument using the `##` syntax.

```
@startuml
!unquoted procedure COMP_TEXTGENCOMP(name)
[name] << Comp >>
interface Ifc << IfcType >> AS name##Ifc
name##Ifc - [name]
!endprocedure
COMP_TEXTGENCOMP(dummy)
@enduml
```





25.19 Dynamic invocation [%invoke_procedure(), %call_user_func()]

You can dynamically invoke a procedure using the special `%invoke_procedure()` procedure. This procedure takes as first argument the name of the actual procedure to be called. The optional following arguments are copied to the called procedure.

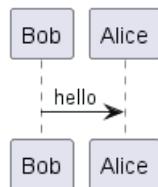
For example, you can have:

```

@startuml
!procedure $go()
  Bob -> Alice : hello
!endprocedure

!$wrapper = "$go"

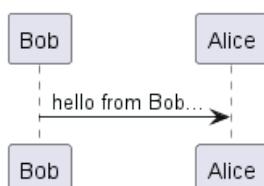
%invoke_procedure($wrapper)
@enduml
  
```



```

@startuml
!procedure $go($txt)
  Bob -> Alice : $txt
!endprocedure

%invoke_procedure("$go", "hello from Bob...")
@enduml
  
```

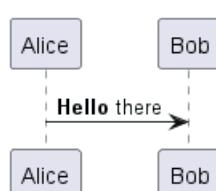


For return functions, you can use the corresponding special function `%call_user_func()`:

```

@startuml
!function bold($text)
!return "<b>" + $text + "</b>"
!endfunction

Alice -> Bob : %call_user_func("bold", "Hello") there
@enduml
  
```



25.20 Evaluation of addition depending of data types [+]

Evaluation of \$a + \$b depending of type of \$a or \$b

```
@startuml
title
<#LightBlue>|= |= $a |= $b |= <U+0025>string($a + $b) |
<#LightGray>| type | str | str | str (concatenation) |
| example |= "a" |= "b" |= %string("a" + "b") |
<#LightGray>| type | str | int | str (concatenation) |
| ex.|= "a" |= 2 |= %string("a" + 2) |
<#LightGray>| type | str | int | str (concatenation) |
| ex.|= 1 |= "b" |= %string(1 + "b") |
<#LightGray>| type | bool | str | str (concatenation) |
| ex.|= <U+0025>true() |= "b" |= %string(%true() + "b") |
<#LightGray>| type | str | bool | str (concatenation) |
| ex.|= "a" |= <U+0025>false() |= %string("a" + %false()) |
<#LightGray>| type | int | int | int (addition of int) |
| ex.|= 1 |= 2 |= %string(1 + 2) |
<#LightGray>| type | bool | int | int (addition) |
| ex.|= <U+0025>true() |= 2 |= %string(%true() + 2) |
<#LightGray>| type | int | bool | int (addition) |
| ex.|= 1 |= <U+0025>false() |= %string(1 + %false()) |
<#LightGray>| type | int | int | int (addition) |
| ex.|= 1 |= <U+0025>intval("2") |= %string(1 + %intval("2")) |
end title
@enduml
```

	\$a	\$b	%string(\$a + \$b)
type	str	str	str (concatenation)
example	"a"	"b"	ab
type	str	int	str (concatenation)
ex.	"a"	2	a2
type	str	int	str (concatenation)
ex.	1	"b"	1b
type	bool	str	str (concatenation)
ex.	%true()	"b"	1b
type	str	bool	str (concatenation)
ex.	"a"	%false()	a0
type	int	int	int (addition of int)
ex.	1	2	3
type	bool	int	int (addition)
ex.	%true()	2	3
type	int	bool	int (addition)
ex.	1	%false()	1
type	int	int	int (addition)
ex.	1	%intval("2")	3

25.21 Preprocessing JSON

You can extend the functionality of the current Preprocessing with JSON Preprocessing features:

- JSON Variable definition
- Access to JSON data
- Loop over JSON array

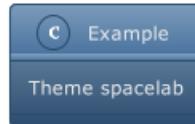
(See more details on [Preprocessing-JSON page](#))

25.22 Including theme [!theme]

Use the !theme directive to change the default theme of your diagram.



```
@startuml
!theme spacelab
class Example {
    Theme spacelab
}
@enduml
```



You will find more information on the dedicated page.

25.23 Migration notes

The current preprocessor is an update from some legacy preprocessor.

Even if some legacy features are still supported with the actual preprocessor, you should not use them any more (they might be removed in some long term future).

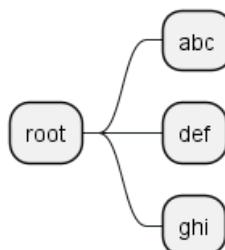
- You should not use `!define` and `!definelong` anymore. Use `!function`, `!procedure` or variable definition instead.
 - `!define` should be replaced by `return !function`
 - `!definelong` should be replaced by `!procedure`.
- `!include` now allows multiple inclusions : you don't have to use `!include_many` anymore
- `!include` now accepts a URL, so you don't need `!includeurl`
- Some features (like `%date%`) have been replaced by builtin functions (for example `%date()`)
- When calling a legacy `!definelong` macro with no arguments, you do have to use parenthesis. You have to use `my_own_definelong()` because `my_own_definelong` without parenthesis is not recognized by the new preprocessor.

Please contact us if you have any issues.

25.24 %Splitstr builtin function

```
@startmindmap
!$list = %splitstr("abc~def~ghi", "~")

* root
!foreach $item in $list
  ** $item
!endfor
@endmindmap
```



[Ref. QA-15374]



26 Unicode

The PlantUML language use *letters* to define actor, usecase and so on.

But *letters* are not only A-Z latin characters, it could be *any kind of letter from any language*.

26.1 Examples

```
@startuml
skinparam handwritten true
skinparam backgroundColor #EEEBDC

actor 使用者
participant "頭等艙" as A
participant "第二類" as B
participant "最後一堂課" as 別的東西

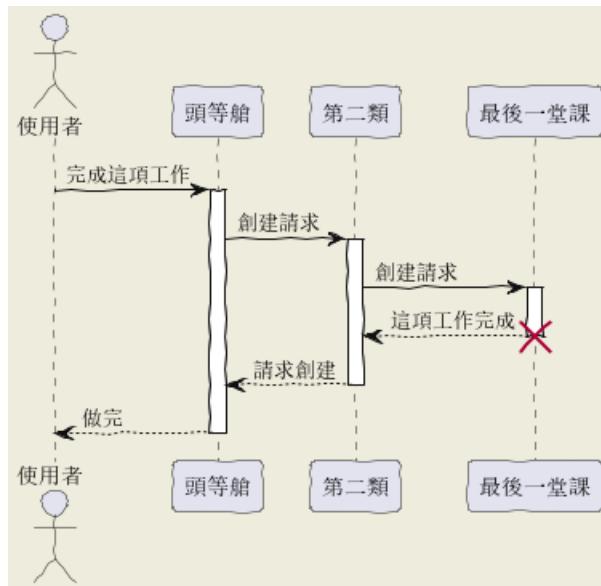
使用者 -> A: 完成這項工作
activate A

A -> B: 創建請求
activate B

B -> 別的東西: 創建請求
activate 別的東西
別的東西 --> B: 這項工作完成
destroy 別的東西

B --> A: 請求創建
deactivate B

A --> 使用者: 做完
deactivate A
@enduml
```



```
@startuml
(*) --> "膩平台"
--> === S1 ===
--> 鞠躬向公眾
--> === S2 ===
```



--> 這傢伙波武器
 --> (*)

```
skinparam backgroundColor #AFFFFF
skinparam activityStartColor red
skinparam activityBarColor SaddleBrown
skinparam activityEndColor Silver
skinparam activityBackgroundColor Peru
skinparam activityBorderColor Peru
@enduml
```



@startuml

```
skinparam usecaseBackgroundColor DarkSeaGreen
skinparam usecaseArrowColor Olive
skinparam actorBorderColor black
skinparam usecaseBorderColor DarkSlateGray
```

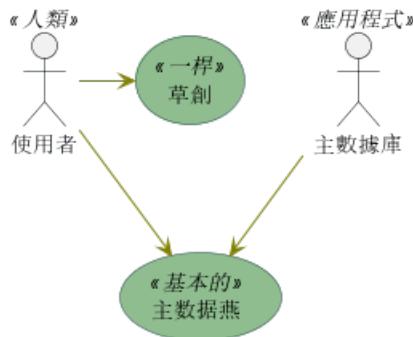
使用者 << 人類 >>
 "主數據庫" as 數據庫 << 應用程式 >>
 (草創) << 一桿 >>
 "主数据燕" as (贏余) << 基本的 >>

使用者 -> (草創)
 使用者 --> (贏余)

數據庫 --> (贏余)

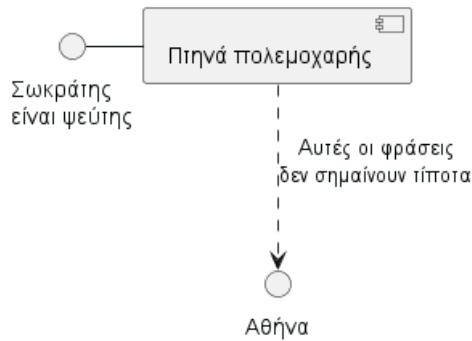
@enduml





```

@startuml
() "Σ" as Σ
Σ - [Π]
[Π] ..> () A : A
@enduml
  
```



26.2 Charset

The default charset used when *reading* the text files containing the UML text description is system dependent.

Normally, it should just be fine, but in some case, you may want to use another charset. For example, with the command line:

```
java -jar plantuml.jar -charset UTF-8 files.txt
```

Or, with the ant task:

```
<!-- Put images in c:/images directory -->
<target name="main">
<plantuml dir=".src" charset="UTF-8" />
```

Depending of your Java installation, the following charset should be available: ISO-8859-1, UTF-8, UTF-16BE, UTF-16LE, UTF-16.

26.3 Using Unicode Character on PlantUML

On PlantUML diagram, you can integrate:

- Special characters using `&#XXXX;` or `<U+XXXX>` form;
- Emoji using `<:XXXXX:>` or `<:NameOfEmoji:>` form.



27 PlantUML Standard Library

Welcome to the guide on PlantUML's official **Standard Library (stdlib)**. Here, we delve into this integral resource that is now included in all official releases of PlantUML, facilitating a richer diagram creation experience. The library borrows its file inclusion convention from the "C standard library", a well-established protocol in the programming world.

27.0.1 Standard Library Overview

The Standard Library is a repository of files and resources, constantly updated to enhance your PlantUML experience. It forms the backbone of PlantUML, offering a range of functionalities and features to explore.

27.0.2 Contribution from the Community

A significant portion of the library's contents are generously provided by third-party contributors. We extend our heartfelt gratitude to them for their invaluable contributions that have played a pivotal role in enriching the library.

We encourage users to delve into the abundant resources the Standard Library offers, to not only enhance their diagram crafting experience but also possibly contribute and be a part of this collaborative endeavor.

27.1 List of Standard Library

You can list standard library folders using the special diagram:

```
@startuml  
stdlib  
@enduml
```



archimate

Version 1.1.0

Delivered by <https://github.com/plantuml-stdlib/Archimate-PlantUML>**aws**

Version 18.02.22

Delivered by <https://github.com/milo-minderbinder/AWS-PlantUML>**awslib**

Version 14.0.0

Delivered by <https://github.com/aws-labs/aws-icons-for-plantuml>**azure**

Version 2.2.0

Delivered by <https://github.com/plantuml-stdlib/Azure-PlantUML>**c4**

Version 2.8.0

Delivered by <https://github.com/plantuml-stdlib/C4-PlantUML>**classy**

Version 1.0.0

Delivered by <https://github.com/james-gadrow-kr/classy-plantuml>**classy-c4**

Version 1.0.0

Delivered by <https://github.com/james-gadrow-kr/classy-c4>**cloudinsight**

Version 1.0.0

Delivered by <https://github.com/plantuml-stdlib/cicon-plantuml-sprites>**cloudogu**

Version 1.0.2

Delivered by <https://github.com/cloudogu/plantuml-cloudogu-sprites>**domainstory**

Version 0.3

Delivered by <https://github.com/johthor/DomainStory-PlantUML>**elastic**

Version 0.0.1

Delivered by <https://github.com/Crashedmind/PlantUML-Elastic-icons>**kubernetes**

Version 5.3.45

Delivered by <https://github.com/plantuml-stdlib/plantuml-kubernetes-sprites>**logos**

Version 1.1.0

Delivered by <https://github.com/plantuml-stdlib/gilbarbara-plantuml-sprites>**material**

Version 0.0.1

Delivered by <https://github.com/Templarian/MaterialDesign>**office**

Version 1.0.0

Delivered by <https://github.com/Roemer/plantuml-office>**osa**

Version 0.0.1

Delivered by <https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons>**tupadr3**

Version 2.4.0

Delivered by <https://github.com/tupadr3/plantuml-icon-font-sprites>

It is also possible to use the command line `java -jar plantuml.jar -stdlib` to display the same list.

Finally, you can extract the full standard library sources using `java -jar plantuml.jar -extractstdlib`. All files will be extracted in the folder `stdlib`.

Sources used to build official PlantUML releases are hosted here <https://github.com/plantuml/plantuml>



stdlib. You can create Pull Request to update or add some library if you find it relevant.

27.2 ArchiMate [archimate]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/archimate
src	https://github.com/ebbypeter/Archimate-PlantUML
orig	https://en.wikipedia.org/wiki/ArchiMate

This repository contains ArchiMate PlantUML macros and other includes for creating Archimate Diagrams easily and consistently.

```
@startuml
!include <archimate/Archimate>

title Archimate Sample - Internet Browser

' Elements
Business_Object(businessObject, "A Business Object")
Business_Process(someBusinessProcess, "Some Business Process")
Business_Service(itSupportService, "IT Support for Business (Application Service)")

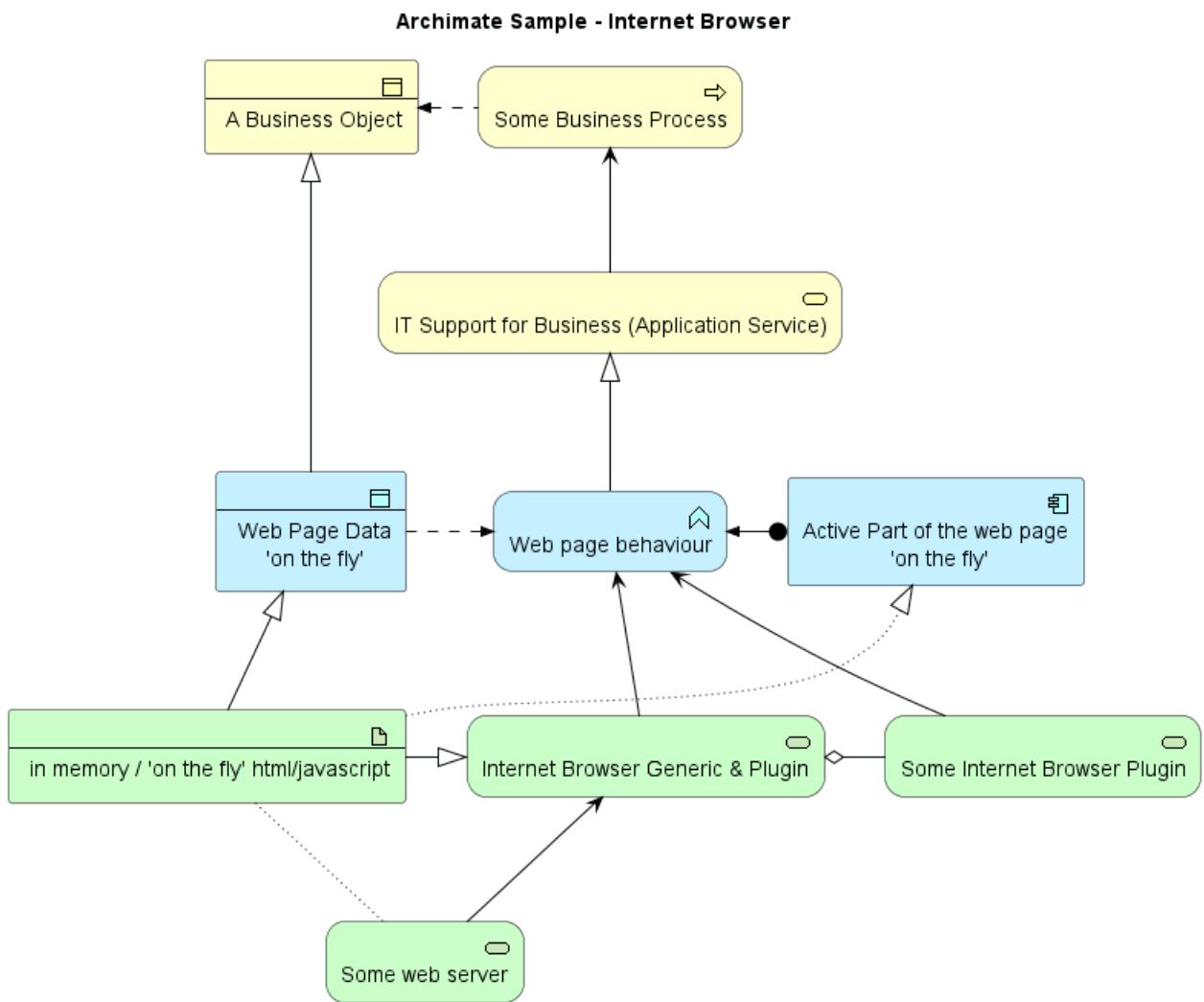
Application_DataObject(dataObject, "Web Page Data \n 'on the fly'")
Application_Function(webpageBehaviour, "Web page behaviour")
Application_Component(ActivePartWebPage, "Active Part of the web page \n 'on the fly')

Technology_Artifact(inMemoryItem, "in memory / 'on the fly' html/javascript")
Technology_Service(internetBrowser, "Internet Browser Generic & Plugin")
Technology_Service(internetBrowserPlugin, "Some Internet Browser Plugin")
Technology_Service(webServer, "Some web server")

'Relationships
Rel_Flow_Left(someBusinessProcess, businessObject, "")
Rel_Serving_Up(itSupportService, someBusinessProcess, "")
Rel_Specialization_Up(webpageBehaviour, itSupportService, "")
Rel_Flow_Right(dataObject, webpageBehaviour, "")
Rel_Specialization_Up(dataObject, businessObject, "")
Rel_Assignment_Left(ActivePartWebPage, webpageBehaviour, "")
Rel_Specialization_Up(inMemoryItem, dataObject, "")
Rel_Realization_Up(inMemoryItem, ActivePartWebPage, "")
Rel_Specialization_Right(inMemoryItem, internetBrowser, "")
Rel_Serving_Up(internetBrowser, webpageBehaviour, "")
Rel_Serving_Up(internetBrowserPlugin, webpageBehaviour, "")
Rel_Aggregation_Right(internetBrowser, internetBrowserPlugin, "")
Rel_Access_Up(webServer, inMemoryItem, "")
Rel_Serving_Up(webServer, internetBrowser, "")

@enduml
```





27.2.1 List possible sprites

You can list all possible sprites for Archimate using the following diagram:

```
@startuml
listsprite
@enduml
```





27.3 Amazon Labs AWS Library [awslib]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/awslib
src	https://github.com/awslabs/aws-icons-for-plantuml
orig	https://aws.amazon.com/en/architecture/icons/

The Amazon Labs AWS library provides PlantUML sprites, macros, and other includes for Amazon Web Services (AWS) services and resources.

Used to create PlantUML diagrams with AWS components. All elements are generated from the official AWS Architecture Icons and when combined with PlantUML and the C4 model, are a great way to communicate your design, deployment, and topology as code.

```

@startuml
!include <awslib/AWSCommon>
!include <awslib/InternetOfThings/IoTRule>
!include <awslib/Analytics/KinesisDataStreams>
!include <awslib/ApplicationIntegration/SimpleQueueService>

left to right direction

agent "Published Event" as event #fff

IoTRule(iotRule, "Action Error Rule", "error if Kinesis fails")
KinesisDataStreams(eventStream, "IoT Events", "2 shards")
SimpleQueueService(errorQueue, "Rule Error Queue", "failed Rule actions")

event --> iotRule : JSON message
iotRule --> eventStream : messages
iotRule --> errorQueue : Failed action message
@enduml

```



27.4 Azure library [azure]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/azure
src	https://github.com/RicardoNiepel/Azure-PlantUML/
orig	Microsoft Azure

The Azure library consists of Microsoft Azure icons.

Use it by including the file that contains the sprite, eg: `!include <azure/Analytics/AzureEventHub>`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

You may also include the `AzureCommon.puml` file, eg: `!include <azure/AzureCommon>`, which contains helper macros defined. With the `AzureCommon.puml` imported, you can use the `NAME_OF_SPRITE(parameters...)` macro.

Example of usage:

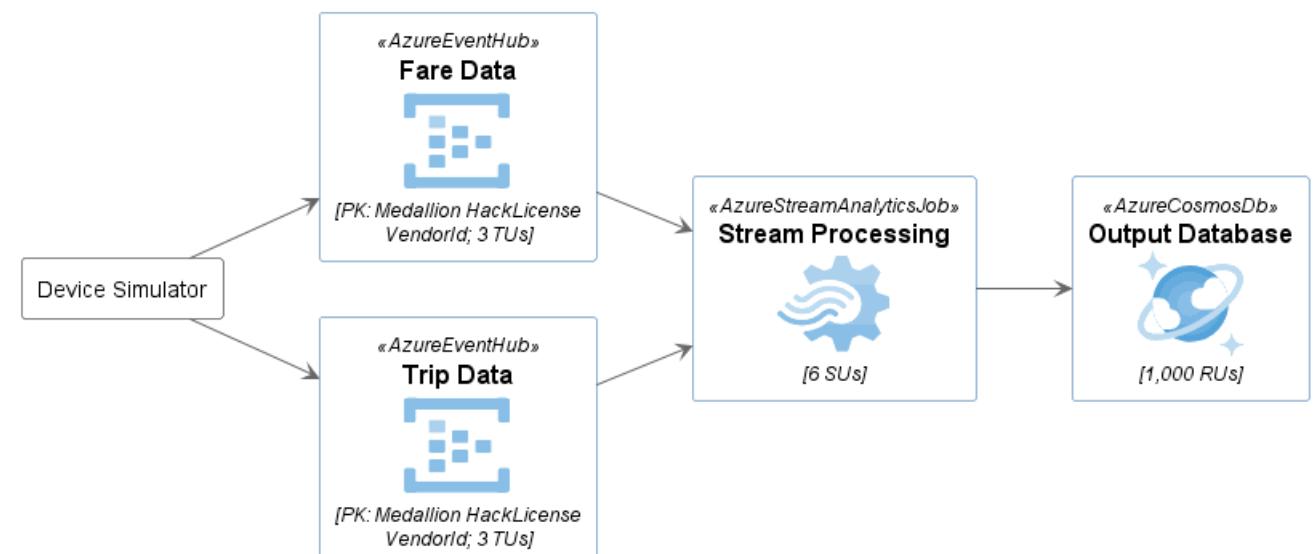
```
@startuml
!include <azure/AzureCommon>
!include <azure/Analytics/AzureEventHub>
!include <azure/Analytics/AzureStreamAnalyticsJob>
!include <azure/Databases/AzureCosmosDb>
```

left to right direction

```
agent "Device Simulator" as devices #fff
```

```
AzureEventHub(fareDataEventHub, "Fare Data", "PK: Medallion HackLicense VendorId; 3 TUs")
AzureEventHub(tripDataEventHub, "Trip Data", "PK: Medallion HackLicense VendorId; 3 TUs")
AzureStreamAnalyticsJob(streamAnalytics, "Stream Processing", "6 SUs")
AzureCosmosDb(outputCosmosDb, "Output Database", "1,000 RUs")
```

```
devices --> fareDataEventHub
devices --> tripDataEventHub
fareDataEventHub --> streamAnalytics
tripDataEventHub --> streamAnalytics
streamAnalytics --> outputCosmosDb
@enduml
```



27.5 C4 Library [C4]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/C4
src	https://github.com/plantuml-stdlib/C4-PlantUML
orig	https://en.wikipedia.org/wiki/C4_model

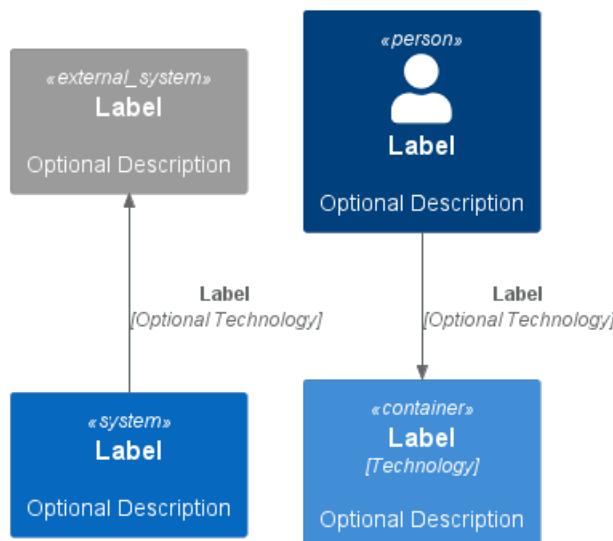
```
@startuml
!include <C4/C4_Container>

Person(personAlias, "Label", "Optional Description")
Container(containerAlias, "Label", "Technology", "Optional Description")
System(systemAlias, "Label", "Optional Description")

System_Ext(extSystemAlias, "Label", "Optional Description")

Rel(personAlias, containerAlias, "Label", "Optional Technology")

Rel_U(systemAlias, extSystemAlias, "Label", "Optional Technology")
@enduml
```



27.6 Cloud Insight [cloudinsight]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/cloudinsight
src	https://github.com/rabelenda/cicon-plantuml-sprites
orig	Cloudinsight icons

This repository contains PlantUML sprites generated from Cloudinsight icons, which can easily be used in PlantUML diagrams for nice visual representation of popular technologies.

```
@startuml
!include <cloudinsight/tomcat>
!include <cloudinsight/kafka>
!include <cloudinsight/java>
!include <cloudinsight/cassandra>

title Cloudinsight sprites example

skinparam monochrome true

rectangle "<$tomcat>\nwebapp" as webapp
```



```

queue "<$kafka>" as kafka
rectangle "<$java>\ndaemon" as daemon
database "<$cassandra>" as cassandra

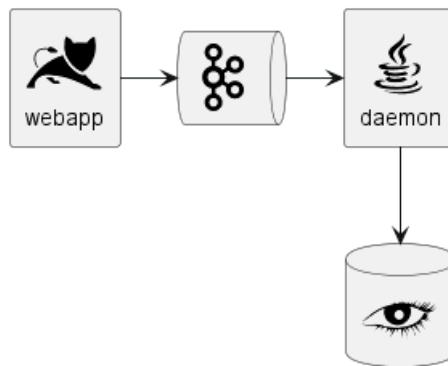
```

```

webapp -> kafka
kafka -> daemon
daemon --> cassandra
@enduml

```

Cloudinsight sprites example



27.7 Cloudogu [cloudogu]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/cloudogu
src	https://github.com/cloudogu/plantuml-cloudogu-sprites
orig	https://cloudogu.com

The Cloudogu library provides PlantUML sprites, macros, and other includes for Cloudogu services and resources.

```

@startuml
!include <cloudogu/common>
!include <cloudogu/dogus/jenkins>
!include <cloudogu/dogus/cloudogu>
!include <cloudogu/dogus/scm>
!include <cloudogu/dogus/smeagol>
!include <cloudogu/dogus/nexus>
!include <cloudogu/tools/k8s>

node "Cloudogu Ecosystem" <<$cloudogu>> {
    DOGU_JENKINS(jenkins, Jenkins) #ffffff
    DOGU_SCM(scm, SCM-Manager) #ffffff
    DOGU_SMEAGOL(smeagol, Smeagol) #ffffff
    DOGU_NEXUS(nexus, Nexus) #ffffff
}

TOOL_K8S(k8s, Kubernetes) #ffffff

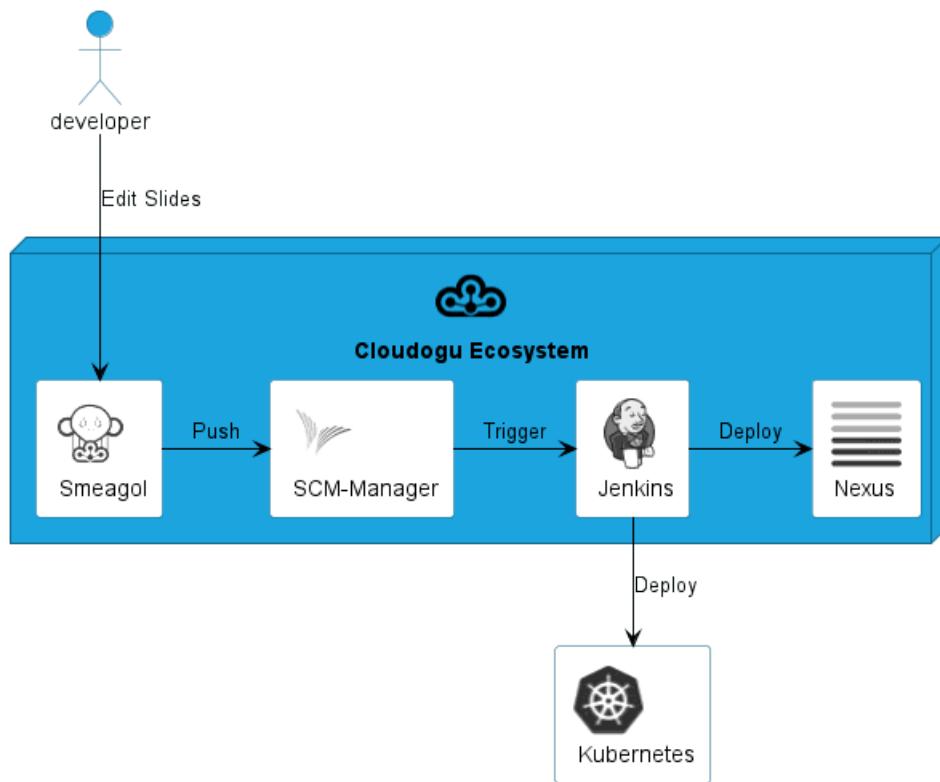
actor developer

developer --> smeagol : "Edit Slides"
smeagol -> scm : Push
scm -> jenkins : Trigger
jenkins -> nexus : Deploy
jenkins --> k8s : Deploy

```



@enduml

**All cloudogu sprites**

See all possible cloudogu sprites on plantuml-cloudogu-sprites.

27.8 Elastic library [elastic]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/elastic
src	https://github.com/Crashedmind/PlantUML-Elastic-icons
orig	Elastic

The Elastic library consists of Elastic icons. It is similar in use to the AWS and Azure libraries (it used the same tool to create them).

Use it by including the file that contains the sprite, eg: `!include elastic/elasticsearch/elasticsearch`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

You may also include the `common.puml` file, eg: `!include <elastic/common>`, which contains helper macros defined. With the `common.puml` imported, you can use the `NAME//OF//SPRITE(parameters...)` macro.

Example of usage:

```

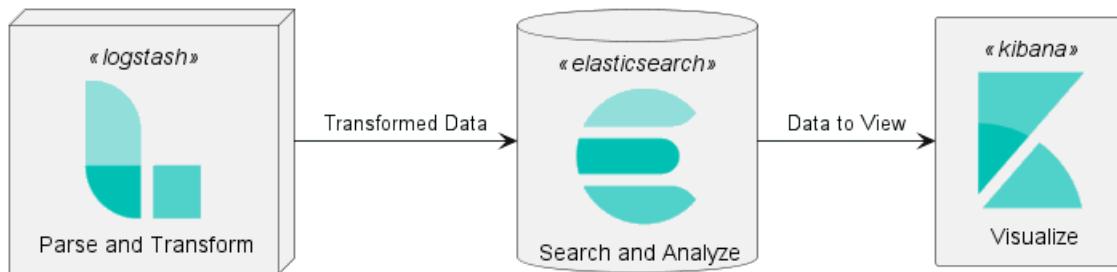
@startuml
!include <elastic/common>
!include <elastic/elasticsearch/elasticsearch>
!include <elastic/logstash/logstash>
!include <elastic/kibana/kibana>

ELASTICSEARCH(ElasticSearch, "Search and Analyze", database)
LOGSTASH(Logstash, "Parse and Transform", node)
KIBANA(Kibana, "Visualize", agent)

Logstash -right-> Elasticsearch: Transformed Data
    
```



ElasticSearch -right-> Kibana: Data to View
 @enduml



All Elastic Sprite Set

```

@startuml
'Adapted from https://github.com/Crashedmind/PlantUML-Elastic-icons/blob/master/All.puml

'Elastic stuff here
'=====

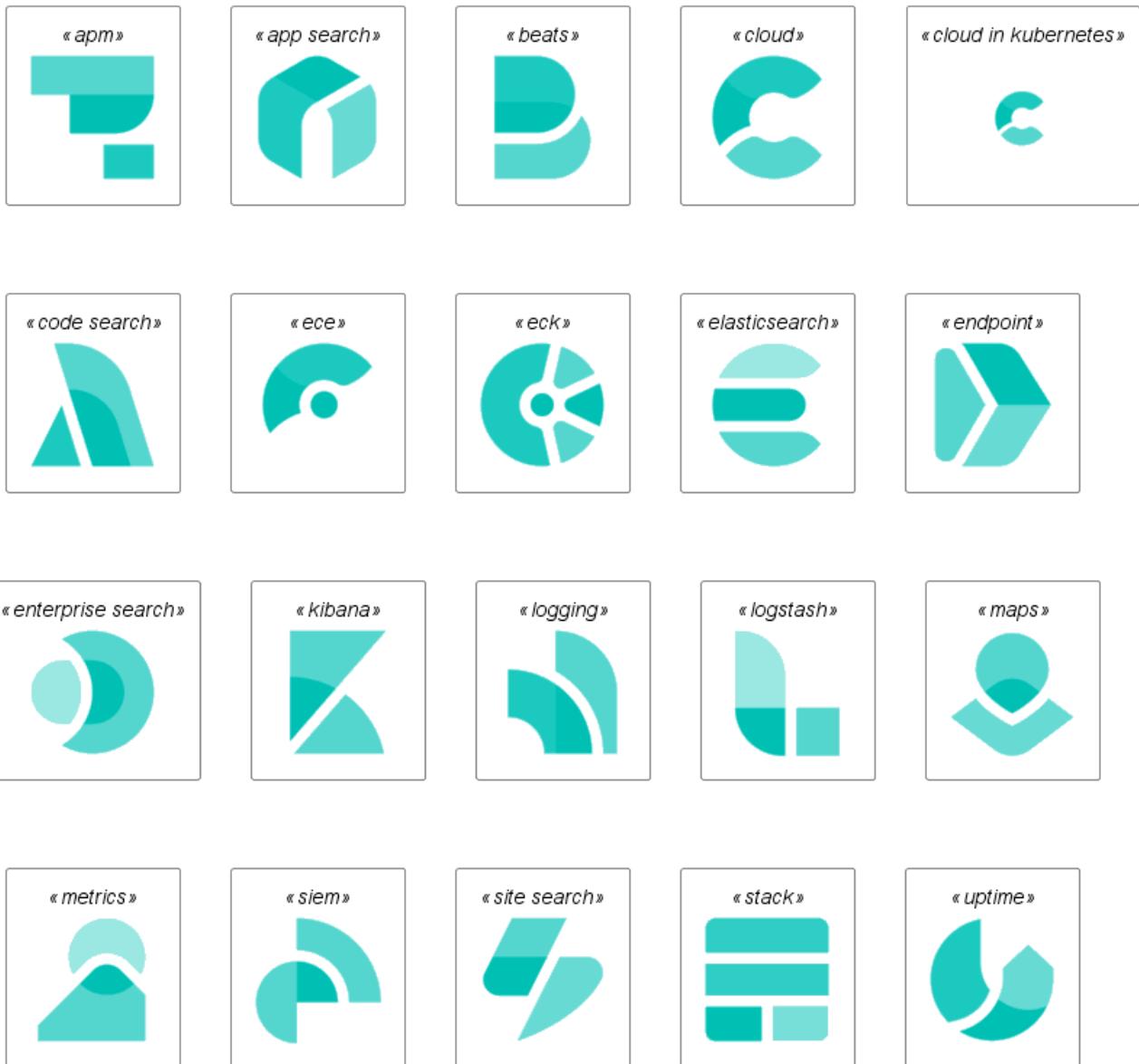
!include <elastic/common>
!include <elastic/apm/apm>
!include <elastic/app_search/app_search>
!include <elastic/beats/beats>
!include <elastic/cloud/cloud>
!include <elastic/cloud_in_kubernetes/cloud_in_kubernetes>
!include <elastic/code_search/code_search>
!include <elastic/ece/ece>
!include <elastic/eck/eck>
' Beware of the difference between Crashedmind and plantuml-stdlib version: with '_' usage!
!include <elastic/elasticsearch/elasticsearch>
!include <elastic/endpoint/endpoint>
!include <elastic/enterprise_search/enterprise_search>
!include <elastic/kibana/kibana>
!include <elastic/logging/logging>
!include <elastic/logstash/logstash>
!include <elastic/maps/maps>
!include <elastic/metrics/metrics>
!include <elastic/siem/siem>
!include <elastic/site_search/site_search>
!include <elastic/stack/stack>
!include <elastic/uptime/uptime>

skinparam agentBackgroundColor White

APM(apm)
APP_SEARCH(app_search)
BEATS(beats)
CLOUD(cloud)
CLOUD_IN_KUBERNETES(cloud_in_kubernetes)
CODE_SEARCH(code_search)
ECE(ece)
ECK(eck)
ELASTICSEARCH(elastic_search)
ENDPOINT(endpoint)
ENTERPRISE_SEARCH(enterprise_search)
KIBANA(kibana)
LOGGING(logging)
  
```



```
LOGSTASH(logstash)
MAPS.maps
METRICS(metrics)
SIEM(siem)
SITE_SEARCH(site_search)
STACK(stack)
UPTIME(uptime)
@enduml
```



27.9 Google Material Icons [material]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/material
src	https://github.com/Templarian/MaterialDesign
orig	Material Design Icons

This library consists of a free Material style icons from Google and other artists.

Use it by including the file that contains the sprite, eg: `!include <material/ma_folder_move>`. When imported, you can use the sprite as normally you would, using `<$ma_sprite_name>`. Notice that this



library requires an `ma_` prefix on sprites names, this is to avoid clash of names if multiple sprites have the same name on different libraries.

You may also include the `common.puml` file, eg: `!include <material/common>`, which contains helper macros defined. With the `common.puml` imported, you can use the `MA_NAME_OF_SPRITE(parameters...)` macro, note again the use of the prefix `MA_`.

Example of usage:

```
@startuml
!include <material/common>
' To import the sprite file you DON'T need to place a prefix!
!include <material/folder_move>

MA_FOLDER_MOVE(Red, 1, dir, rectangle, "A label")
@enduml
```



Notes:

When mixing sprites macros with other elements you may get a syntax error if, for example, trying to add a rectangle along with classes. In those cases, add `{` and `}` after the macro to create the empty rectangle.

Example of usage:

```
@startuml
!include <material/common>
' To import the sprite file you DON'T need to place a prefix!
!include <material/folder_move>

MA_FOLDER_MOVE(Red, 1, dir, rectangle, "A label") {
}

class foo {
    bar
}
@enduml
```



27.10 Kubernetes [kubernetes]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/kubernetes
src	https://github.com/michiel/plantuml-kubernetes-sprites
orig	Kubernetes

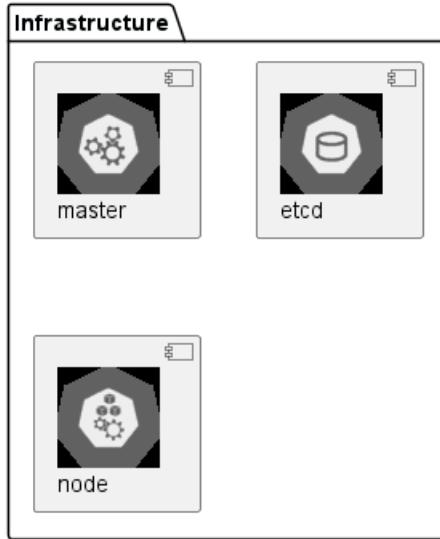
```
@startuml
!include <kubernetes/k8s-sprites-unlabeled-25pct>
package "Infrastructure" {
    component "<$master>\nmaster" as master
```



```

component "<$etcd>\netcd" as etcd
component "<$node>\nnode" as node
}
@enduml

```



27.11 Logos [logos]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/logos
src	https://github.com/plantuml-stdlib/gilbarbara-plantuml-sprites
orig	Gil Barbara's logos

This repository contains PlantUML sprites generated from Gil Barbara's logos, which can easily be used in PlantUML diagrams for nice visual aid.

```

@startuml
!include <logos/flask>
!include <logos/kafka>
!include <logos/kotlin>
!include <logos/cassandra>

title Gil Barbara's logos example

skinparam monochrome true

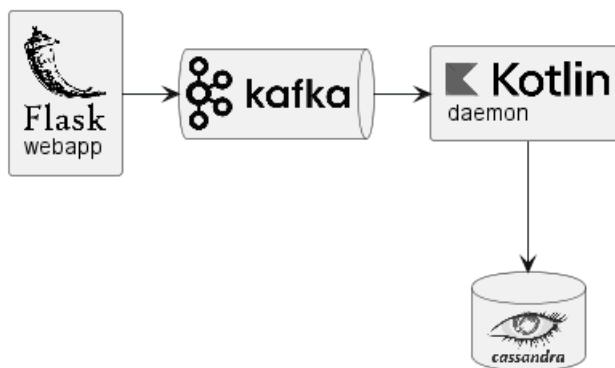
rectangle "<$flask>\nwebapp" as webapp
queue "<$kafka>" as kafka
rectangle "<$kotlin>\ndaemon" as daemon
database "<$cassandra>" as cassandra

webapp -> kafka
kafka -> daemon
daemon --> cassandra
@enduml

```



Gil Barbara's logos example



```

@startuml
scale 0.7
!include <logos/apple-pay>
!include <logos/dinersclub>
!include <logos/discover>
!include <logos/google-pay>
!include <logos/jcb>
!include <logos/maestro>
!include <logos/mastercard>
!include <logos/paypal>
!include <logos/unionpay>
!include <logos/visaelectron>
!include <logos/visa>
' ...

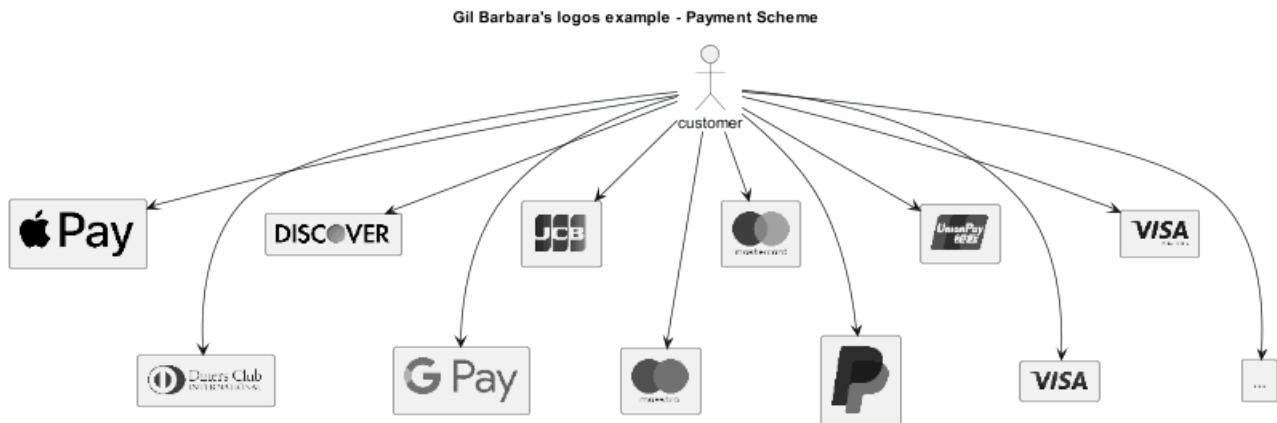
title Gil Barbara's logos example - **Payment Scheme**

actor customer
rectangle "<$apple-pay>" as ap
rectangle "<$dinersclub>" as dc
rectangle "<$discover>" as d
rectangle "<$google-pay>" as gp
rectangle "<$jcb>" as j
rectangle "<$maestro>" as ma
rectangle "<$mastercard>" as m
rectangle "<$paypal>" as p
rectangle "<$unionpay>" as up
rectangle "<$visa>" as v
rectangle "<$visaelectron>" as ve
rectangle "..." as etc

customer --> ap
customer ---> dc
customer --> d
customer ---> gp
customer --> j
customer ---> ma
customer --> m
customer ---> p
customer --> up
customer ---> v
customer --> ve
customer ---> etc
  
```



```
@enduml
```



27.12 Office [office]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/office
src	https://github.com/Roemer/plantuml-office
orig	

There are sprites (*.puml) and colored png icons available. Be aware that the sprites are all only monochrome even if they have a color in their name (due to automatically generating the files). You can either color the sprites with the macro (see examples below) or directly use the fully colored pngs. See the following examples on how to use the sprites, the pngs and the macros.

Example of usage:

```

@startuml
!include <tupadr3/common>

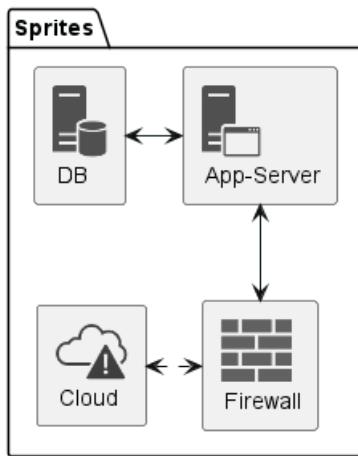
!include <office/Servers/database_server>
!include <office/Servers/application_server>
!include <office/Concepts/firewall_orange>
!include <office/Clouds/cloud_disaster_red>

title Office Icons Example

package "Sprites" {
    OFF_DATABASE_SERVER(db,DB)
    OFF_APPLICATION_SERVER(app,App-Server)
    OFF_FIREWALL_ORANGE(fw,Firewall)
    OFF_CLOUD_DISASTER_RED(cloud,Cloud)
    db <-> app
    app <--> fw
    fw <.left.> cloud
}
@enduml

```



Office Icons Example

```

@startuml
!include <tupadr3/common>

!include <office/servers/database_server>
!include <office/servers/application_server>
!include <office/Concepts/firewall_orange>
!include <office/Clouds/cloud_disaster_red>

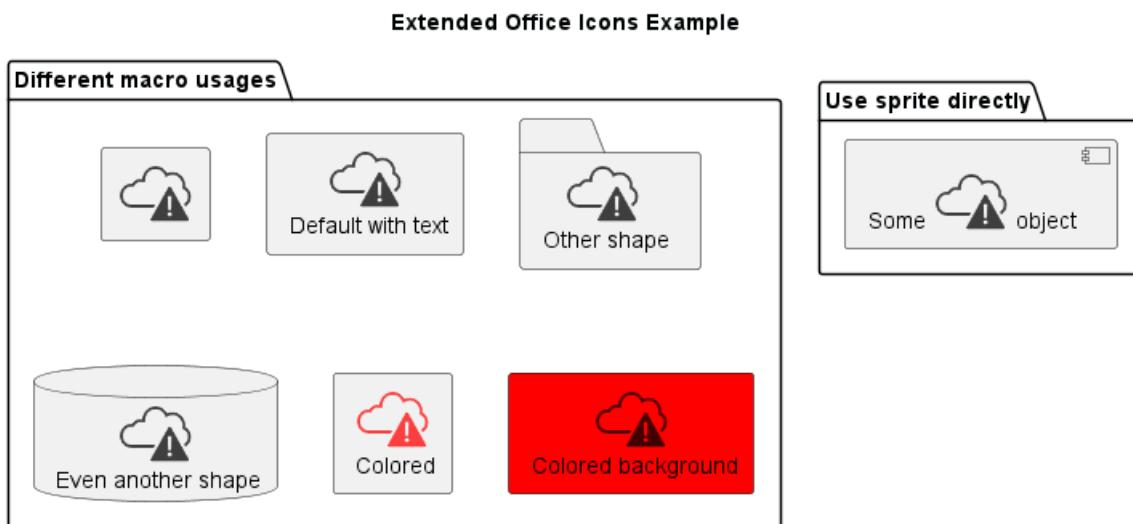
' Used to center the label under the images
skinparam defaultTextAlignment center

title Extended Office Icons Example

package "Use sprite directly" {
    [Some <$cloud_disaster_red> object]
}

package "Different macro usages" {
    OFF_CLOUD_DISASTER_RED(cloud1)
    OFF_CLOUD_DISASTER_RED(cloud2,Default with text)
    OFF_CLOUD_DISASTER_RED(cloud3,Other shape,Folder)
    OFF_CLOUD_DISASTER_RED(cloud4,Even another shape,Database)
    OFF_CLOUD_DISASTER_RED(cloud5,Colored,Rectangle, red)
    OFF_CLOUD_DISASTER_RED(cloud6,Colored background) #red
}
@enduml
  
```





27.13 Open Security Architecture (OSA) [osa]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/osa
src	https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons
orig	https://www.opensecurityarchitecture.org

```
@startuml
```

```
'Adapted from https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons/blob/master/all
scale .5
!include <osa/arrow/green/left/left>
!include <osa/arrow/yellow/right/right>
!include <osa/awareness/awareness>
!include <osa/contract/contract>
!include <osa/database/database>
!include <osa/desktop/desktop>
!include <osa/desktop/imac/imac>
!include <osa/device_music/device_music>
!include <osa/device_scanner/device_scanner>
!include <osa/device_usb/device_usb>
!include <osa/device_wireless_router/device_wireless_router>
!include <osa/disposal/disposal>
!include <osa/drive_optical/drive_optical>
!include <osa/firewall/firewall>
!include <osa/hub/hub>
!include <osa/ics/drive/drive>
!include <osa/ics/plc/plc>
!include <osa/ics/thermometer/thermometer>
!include <osa/id/card/card>
!include <osa/laptop/laptop>
!include <osa/lifecycle/lifecycle>
!include <osa/lightning/lightning>
!include <osa/media_flash/media_flash>
!include <osa/media_optical/media_optical>
!include <osa/media_tape/media_tape>
!include <osa/mobile/pda/pda>
!include <osa/padlock/padlock>
!include <osa/printer/printer>
!include <osa/site_branch/site_branch>
!include <osa/site_factory/site_factory>
!include <osa/vpn/vpn>
```



```
!include <osa/wireless/network/network>

rectangle "OSA" {
rectangle "Left:\n <$left>" 
rectangle "Right:\n <$right>" 
rectangle "Awareness:\n <$awareness>" 
rectangle "Contract:\n <$contract>" 
rectangle "Database:\n <$database>" 
rectangle "Desktop:\n <$desktop>" 
rectangle "Imac:\n <$imac>" 
rectangle "Device_music:\n <$device_music>" 
rectangle "Device_scanner:\n <$device_scanner>" 
rectangle "Device_usb:\n <$device_usb>" 
rectangle "Device_wireless_router:\n <$device_wireless_router>" 
rectangle "Disposal:\n <$disposal>" 
rectangle "Drive_optical:\n <$drive_optical>" 
rectangle "Firewall:\n <$firewall>" 
rectangle "Hub:\n <$hub>" 
rectangle "Drive:\n <$drive>" 
rectangle "Plc:\n <$plc>" 
rectangle "Thermometer:\n <$thermometer>" 
rectangle "Card:\n <$card>" 
rectangle "Laptop:\n <$laptop>" 
rectangle "Lifecycle:\n <$lifecycle>" 
rectangle "Lightning:\n <$lightning>" 
rectangle "Media_flash:\n <$media_flash>" 
rectangle "Media_optical:\n <$media_optical>" 
rectangle "Media_tape:\n <$media_tape>" 
rectangle "Pda:\n <$pda>" 
rectangle "Padlock:\n <$padlock>" 
rectangle "Printer:\n <$printer>" 
rectangle "Site_branch:\n <$site_branch>" 
rectangle "Site_factory:\n <$site_factory>" 
rectangle "Vpn:\n <$vpn>" 
rectangle "Network:\n <$network>" 
}
@enduml
```



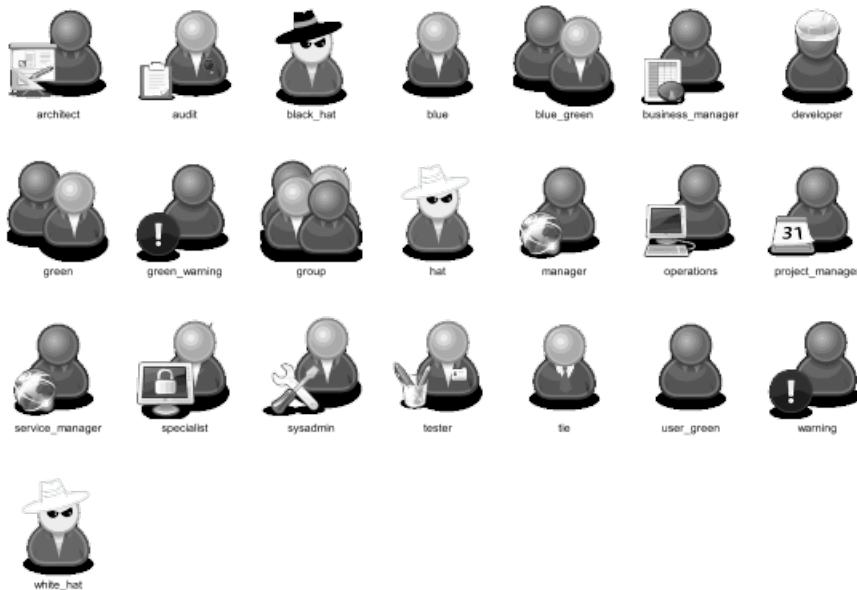


```
@startuml
scale .5
!include <osa/user/audit/audit>
'beware of 'hat-sprite'
!include <osa/user/black/hat/hat-sprite>
!include <osa/user/blue/blue>
!include <osa/user/blue/security/specialist/specialist>
!include <osa/user/blue/sysadmin/sysadmin>
!include <osa/user/blue/tester/tester>
!include <osa/user/blue/tie/tie>
!include <osa/user/green/architect/architect>
!include <osa/user/green/business/manager/manager>
!include <osa/user/green/developer/developer>
!include <osa/user/green/green>
!include <osa/user/green/operations/operations>
!include <osa/user/green/project/manager/manager>
!include <osa/user/green/service/manager/manager>
!include <osa/user/green/warning/warning>
!include <osa/user/large/group/group>
!include <osa/users/blue/green/green>
!include <osa/user/white/hat/hat>
```

listsprites



```
@enduml
```



27.14 Tupadr3 library [tupadr3]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/tupadr3
src	https://github.com/tupadr3/plantuml-icon-font-sprites
orig	https://github.com/tupadr3/plantuml-icon-font-sprites#icon-sets

This library contains several libraries of icons (including Devicons and Font Awesome).

Use it by including the file that contains the sprite, eg: `!include <font-awesome/common>`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

You may also include the `common.puml` file, eg: `!include <font-awesome/common>`, which contains helper macros defined. With the `common.puml` imported, you can use the `NAME_OF_SPRITE(parameters...)` macro.

Example of usage:

```
@startuml
!include <tupadr3/common>
!include <tupadr3/font-awesome/server>
!include <tupadr3/font-awesome/database>
```

```
title Styling example
```

```
FA_SERVER(web1,web1) #Green
FA_SERVER(web2,web2) #Yellow
FA_SERVER(web3,web3) #Blue
FA_SERVER(web4,web4) #YellowGreen

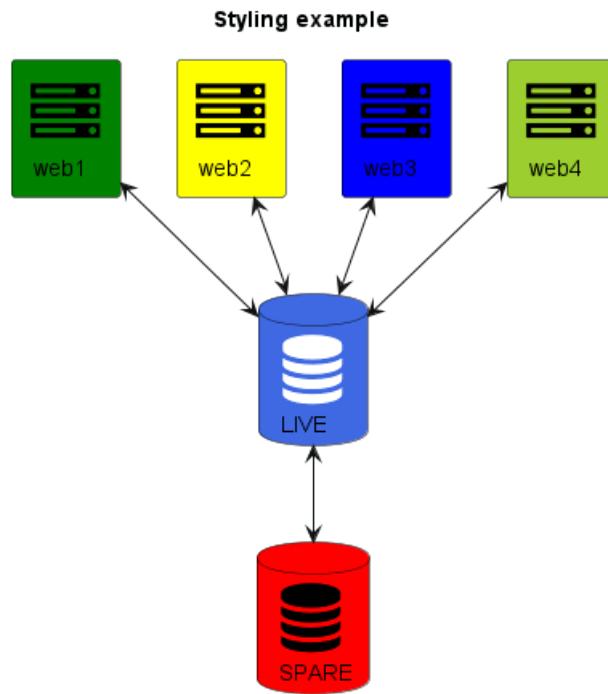
FA_DATABASE(db1,LIVE,database,white) #RoyalBlue
FA_DATABASE(db2,SPARE,database) #Red
```

```
db1 <--> db2
```

```
web1 <--> db1
web2 <--> db1
web3 <--> db1
web4 <--> db1
```



@enduml



@startuml

```

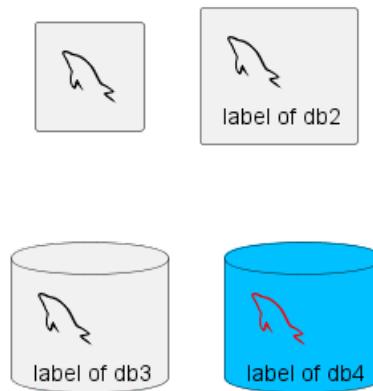
!include <tupadr3/common>
!include <tupadr3/devicons/mysql>

```

```

DEV_MYSQL(db1)
DEV_MYSQL(db2,label of db2)
DEV_MYSQL(db3,label of db3,database)
DEV_MYSQL(db4,label of db4,database,red) #DeepSkyBlue
@enduml

```



27.15 AWS library [aws]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/aws
src	https://github.com/milo-minderbinder/AWS-PlantUML
orig	https://aws.amazon.com/en/architecture/icons/

Warning: We are thinking about deprecating this library.

So you should probably use <awslib> instead (see above).



hr

The AWS library consists of Amazon AWS icons, it provides icons of two different sizes (normal and large).

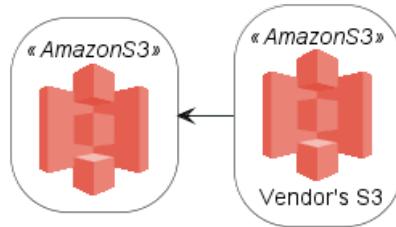
Use it by including the file that contains the sprite, eg: !include <aws/Storage/AmazonS3/AmazonS3>. When imported, you can use the sprite as normally you would, using <\$sprite_name>.

You may also include the common.puml file, eg: !include <aws/common>, which contains helper macros defined. With the common.puml imported, you can use the NAME_OF_SPRITE(parameters...) macro.

Example of usage:

```
@startuml  
!include <aws/common>  
!include <aws/Storage/AmazonS3/AmazonS3>
```

```
AMAZONS3(s3_internal)  
AMAZONS3(s3_partner, "Vendor's S3")  
s3_internal <- s3_partner  
@enduml
```



Contents

1 Diagrama de secuencia	1
1.1 Ejemplo básico	1
1.2 Declarando participantes	2
1.3 Declaring participant on multiline	3
1.4 Sin usar letras en participantes	4
1.5 Mensaje a sí mismo	4
1.6 Text alignment	5
1.6.1 Text of response message below the arrow	5
1.7 Cambiar estilo de la flecha	5
1.8 Cambiar el color de la flecha	6
1.9 Numeración de la secuencia de mensajes	6
1.10 Page Title, Header and Footer	9
1.11 Dividiendo diagramas	9
1.12 Agrupando mensajes	10
1.13 Secondary group label	11
1.14 Notas en mensajes	12
1.15 Algunas otras notas	12
1.16 Cambiando el aspecto de las notas	13
1.17 Note over all participants [across]	13
1.18 Several notes aligned at the same level [/]	14
1.19 Creole y HTML	15
1.20 Divisor	16
1.21 Referencia	17
1.22 Retardo	17
1.23 Text wrapping	18
1.24 Espaciado	18
1.25 Activación y Destrucción de la Línea de vida	19
1.26 Return	20
1.27 Creación de participante	21
1.28 Shortcut syntax for activation, deactivation, creation	21
1.29 Mensajes entrantes y salientes	23
1.30 Short arrows for incoming and outgoing messages	24
1.31 Anchors and Duration	25
1.32 Estereotipos y marcas	25
1.33 Mayor información en los títulos	26
1.34 Entorno de participante	27
1.35 Removiendo pie de página	28
1.36 Personalización (Skinparam)	28
1.37 Cambiando el relleno	30
1.38 Appendix: Examples of all arrow type	31
1.38.1 Normal arrow	31
1.38.2 Itself arrow	32
1.38.3 Incoming and outgoing messages (with '[, ']')	33
1.38.4 Incoming messages (with '[')	33
1.38.5 Outgoing messages (with ']')	35
1.38.6 Short incoming and outgoing messages (with '?')	36
1.38.7 Short incoming (with '?')	36
1.38.8 Short outgoing (with '?')	37
1.39 Specific SkinParameter	39
1.39.1 By default	39
1.39.2 LifelineStrategy	39
1.39.3 style strictuml	39
1.40 Hide unlinked participant	40
1.41 Color a group message	40
1.42 Mainframe	41
1.43 Slanted or odd arrows	41



2 Diagrama de Casos de Uso	44
2.1 Casos de uso	44
2.2 Actores	44
2.3 Cambiar el estilo del actor	45
2.3.1 Hombre palo (<i>por defecto</i>)	45
2.4 Descripción de Casos de uso	46
2.5 Utilice el paquete	46
2.6 Ejemplo básico	48
2.7 Extensión	48
2.8 Usando notas	49
2.9 Estereotipos	49
2.10 Cambio de dirección de las flechas	50
2.11 Dividiendo los diagramas	51
2.12 Dirección: de izquierda a derecha	52
2.13 Personalización (Skinparam)	52
2.14 Un ejemplo completo	53
2.15 Business Use Case	54
2.15.1 Business Usecase	54
2.15.2 Business Actor	54
2.16 Change arrow color and style (inline style)	55
2.17 Change element color and style (inline style)	55
2.18 Display JSON Data on Usecase diagram	56
2.18.1 Simple example	56
3 Diagrama de clase	57
3.1 Elemento declarante	57
3.2 Relación entre clases	58
3.3 Etiquetas en las relaciones	59
3.4 Sin usar letras	59
3.5 Añadir métodos	60
3.6 Definiendo la visibilidad	61
3.7 Abstracto y Estático	62
3.8 Cuerpo avanzado de las clases	62
3.9 Notas y estereotipos	63
3.10 Más acerca de notas	64
3.11 Note on field (field, attribute, member) or method	65
3.11.1 Constraint	65
3.11.2 Note on field or method	65
3.11.3 Note on method with the same name	65
3.12 Notas en enlaces	66
3.13 Clases abstractas e interfaces	66
3.14 Atributos, métodos... ocultos	67
3.15 Clases ocultas	68
3.16 Remove classes	69
3.17 Hide, Remove or Restore tagged element or wildcard	69
3.18 Hide or Remove unlinked class	70
3.19 Uso de clases genéricas	72
3.20 Círculo enmarcador específico	72
3.21 Paquetes	72
3.22 Estilos de paquetes	73
3.23 Espacios de nombre	74
3.24 Creación automática del espacio de nombre	75
3.25 Interface Lollipop	76
3.26 Cambiando la dirección de las flechas	76
3.27 Asociación de clases	77
3.28 Association on same class	78
3.29 Personalización (Skinparam)	79
3.30 Estereotipos personalizados	79



3.31 Degrado de colores	80
3.32 Ayudar en el diseño	81
3.33 Dividiendo archivos grandes	81
3.34 Extends and implements	82
3.35 Bracketed relations (linking or arrow) style	83
3.35.1 Line style	83
3.35.2 Line color	84
3.35.3 Line thickness	84
3.35.4 Mix	85
3.36 Change relation (linking or arrow) color and style (inline style)	85
3.37 Change class color and style (inline style)	86
3.38 Arrows from/to class members	87
3.39 Grouping inheritance arrow heads	88
3.39.1 GroupInheritance 1 (no grouping)	88
3.39.2 GroupInheritance 2 (grouping from 2)	89
3.39.3 GroupInheritance 3 (grouping only from 3)	89
3.39.4 GroupInheritance 4 (grouping only from 4)	90
3.40 Display JSON Data on Class or Object diagram	90
3.40.1 Simple example	90
3.41 Packages and Namespaces Enhancement	91
4 Diagrama de Objetos	93
4.1 Definición de objetos	93
4.2 Relaciones entre objetos	93
4.3 Asociaciones de objetos	94
4.4 Agregando campos	94
4.5 Características comunes en diagramas de clases	95
4.6 Tabla o arreglo asociativo	95
4.7 Program (or project) evaluation and review technique (PERT) with map	97
4.8 Display JSON Data on Class or Object diagram	98
4.8.1 Simple example	98
5 Diagrama de Actividades	100
5.1 Actividades simples	100
5.2 Etiquetas en las flechas	100
5.3 Cambiando la dirección de la flecha	100
5.4 Ramas (bifurcaciones)	101
5.5 Más acerca de las Ramas	102
5.6 Sincronización	103
5.7 Descripción de actividades de gran contenido	104
5.8 Notas	104
5.9 Partición	105
5.10 Personalización (Skinparam)	106
5.11 Octágono	107
5.12 Ejemplo completo	107
6 Diagrama de actividad (nueva sintaxis)	110
6.0.1 Ventajas de la nueva sintaxis	110
6.0.2 Transición a la Nueva Sintaxis	110
6.1 Una Actividad simple	110
6.2 Start/Stop	110
6.3 Condicionales	111
6.4 Switch and case [switch, case, endswitch]	112
6.5 Conditional with stop on an action [kill, detach]	113
6.6 El ciclo Repeat	114
6.7 Break on a repeat loop [break]	115
6.8 Goto and Label Processing [label, goto]	116
6.9 El ciclo While	117
6.10 Procesamiento paralelo	118



6.11	Split processing	119
6.11.1	Split	119
6.11.2	Input split (multi-start)	119
6.11.3	Output split (multi-end)	120
6.12	Notas	121
6.13	Colores	122
6.14	Lines without arrows	122
6.15	Flechas	123
6.16	Connector	124
6.17	Color on connector	124
6.18	Agrupación	125
6.19	Carreles	126
6.20	Desacoplar y remover	126
6.21	Otras formas de representación de actividades	127
6.22	Un ejemplo completo	128
6.23	Condition Style	130
6.23.1	Inside style (by default)	130
6.23.2	Diamond style	131
6.23.3	InsideDiamond (or <i>Foo1</i>) style	132
6.24	Condition End Style	133
6.24.1	Diamond style (by default)	133
6.24.2	Horizontal line (hline) style	134
6.25	Using (global) style	135
6.25.1	Without style (by default)	135
6.25.2	With style	135
7	Diagrama de Componentes	138
7.1	Componentes	138
7.2	Interfaces	138
7.3	Ejemplos basicos	139
7.4	Usando notas	139
7.5	Agrupando componentes	140
7.6	Cambiando la dirección de las flechas	141
7.7	Use UML2 notation	142
7.8	Utiliza la notación UML1	143
7.9	Use rectangle notation (remove UML notation)	143
7.10	Long description	144
7.11	Colores individuales	144
7.12	Using Sprite in Stereotype	144
7.13	Personalización (Skinparam)	145
7.14	Specific SkinParameter	146
7.14.1	componentStyle	146
7.15	Hide or Remove unlinked component	147
7.16	Hide, Remove or Restore tagged component or wildcard	148
7.17	Display JSON Data on Component diagram	150
7.17.1	Simple example	150
7.18	Port [port, portIn, portOut]	151
7.18.1	Port	151
7.18.2	PortIn	151
7.18.3	PortOut	152
7.18.4	Mixing PortIn & PortOut	152
8	Diagrama de despliegue	154
8.1	Declaring element	154
8.2	Declaring element (using short form)	156
8.2.1	Actor	156
8.2.2	Component	157
8.2.3	Interface	157
8.2.4	Usecase	157



8.3	Linking or arrow	157
8.4	Bracketed arrow style	160
8.4.1	Line style	160
8.4.2	Line color	161
8.4.3	Line thickness	161
8.4.4	Mix	162
8.5	Change arrow color and style (inline style)	162
8.6	Change element color and style (inline style)	163
8.7	Nestable elements	164
8.8	Packages and nested elements	164
8.8.1	Example with one level	164
8.8.2	Other example	165
8.8.3	Full nesting	166
8.9	Alias	170
8.9.1	Simple alias with as	170
8.9.2	Examples of long alias	171
8.10	Round corner	173
8.11	Specific SkinParameter	173
8.11.1	roundCorner	173
8.12	Appendix: All type of arrow line	174
8.13	Appendix: All type of arrow head or '0' arrow	175
8.13.1	Type of arrow head	175
8.13.2	Type of '0' arrow or circle arrow	176
8.14	Appendix: Test of inline style on all element	177
8.14.1	Simple element	177
8.14.2	Nested element	178
8.14.3	Without sub-element	178
8.14.4	With sub-element	179
8.15	Appendix: Test of style on all element	180
8.15.1	Simple element	180
8.15.2	Global style (on componentDiagram)	180
8.15.3	Style for each element	181
8.15.4	Nested element (without level)	185
8.15.5	Global style (on componentDiagram)	185
8.15.6	Style for each nested element	185
8.15.7	Nested element (with one level)	187
8.15.8	Global style (on componentDiagram)	187
8.15.9	Style for each nested element	188
8.16	Appendix: Test of stereotype with style on all element	191
8.16.1	Simple element	191
8.17	Display JSON Data on Deployment diagram	192
8.17.1	Simple example	192
8.18	Mixing Deployment (Usecase, Component, Deployment) element within a Class or Object diagram	193
8.18.1	Mixing all elements	193
8.19	Port [port, portIn, portOut]	195
8.19.1	Port	195
8.19.2	PortIn	196
8.19.3	PortOut	196
8.19.4	Mixing PortIn & PortOut	197
9	Diagrama de estado	199
9.1	Estado simple	199
9.2	Cambiar la representación del estado	199
9.3	Estados compuestos	200
9.3.1	Sub-estado a sub-estado	201
9.4	Nombres largos	202
9.5	Histórico [[H], [H*]]	203



9.6	Bifurcación	203
9.7	Estados concurrentes [-,]	204
9.7.1	Separador horizontal --	204
9.7.2	Separador vertical 	205
9.8	Condicional [choice]	206
9.9	Ejemplo completo de etereotipos [start, choice, fork, join, end]	206
9.10	Punto [entryPoint, exitPoint]	208
9.11	Pin [inputPin, outputPin]	208
9.12	Expansión [expansionInput, expansionOutput]	209
9.13	Dirección de flecha	210
9.14	Cambiar el color y estilo de la línea	211
9.15	Notas	211
9.16	Notas sobre conexiones	212
9.17	Más sobre notas	212
9.18	Color entre-línea	213
9.19	Personalización (Skinparam)	214
9.19.1	Prueba de todos los skinparam específicos de los diagramas de estado	215
9.20	Cambiando estilo	215
9.21	Change state color and style (inline style)	216
9.22	Alias	218
9.23	Display JSON Data on State diagram	219
9.23.1	Simple example	219
10	Timing Diagram	220
10.1	Declaring element or participant	220
10.2	Binary and Clock	221
10.3	Adding message	221
10.4	Relative time	222
10.5	Anchor Points	223
10.6	Participant oriented	223
10.7	Setting scale	224
10.8	Initial state	225
10.9	Intricated state	225
10.9.1	Intricated or undefined robust state	225
10.9.2	Intricated or undefined binary state	226
10.10	Hidden state	226
10.11	Hide time axis	228
10.12	Using Time and Date	228
10.13	Adding constraint	229
10.14	Highlighted period	229
10.15	Using notes	230
10.16	Adding texts	231
10.17	Complete example	232
10.18	Digital Example	233
10.19	Adding color	234
10.20	Using (global) style	235
10.20.1	Without style (<i>by default</i>)	235
10.20.2	With style	235
10.21	Applying Colors to specific lines	236
10.22	Compact mode	237
10.22.1	By default	237
10.22.2	Global mode with mode compact	238
10.22.3	Local mode with only compact on element	238
11	Display JSON Data	240
11.1	Complex example	240
11.2	Highlight parts	241
11.3	Using different styles for highlight	241
11.4	JSON basic element	242



11.4.1	Synthesis of all JSON basic element	242
11.5	JSON array or table	243
11.5.1	Array type	243
11.5.2	Minimal array or table	244
11.5.3	Number array	244
11.5.4	String array	244
11.5.5	Boolean array	244
11.6	JSON numbers	244
11.7	JSON strings	245
11.7.1	JSON Unicode	245
11.7.2	JSON two-character escape sequence	245
11.8	Minimal JSON examples	246
11.9	Empty table or list	247
11.10	Using (global) style	247
11.10.1	Without style (<i>by default</i>)	247
11.10.2	With style	248
11.11	Display JSON Data on Class or Object diagram	249
11.11.1	Simple example	249
11.11.2	Complex example: with all JSON basic element	249
11.12	Display JSON Data on Deployment (Usecase, Component, Deployment) diagram	250
11.12.1	Simple example	250
11.13	Display JSON Data on State diagram	251
11.13.1	Simple example	251
12	Display YAML Data	253
12.1	Complex example	253
12.2	Specific key (with symbols or unicode)	254
12.3	Highlight parts	254
12.3.1	Normal style	254
12.3.2	Customised style	255
12.4	Using different styles for highlight	255
12.5	Using (global) style	256
12.5.1	Without style (<i>by default</i>)	256
12.5.2	With style	257
13	Network diagram (nwdiag)	259
13.1	Simple diagram	259
13.1.1	Define a network	259
13.1.2	Define some elements or servers on a network	259
13.1.3	Full example	259
13.2	Define multiple addresses	260
13.3	Grouping nodes	261
13.3.1	Define group inside network definitions	261
13.3.2	Define group outside of network definitions	261
13.3.3	Define several groups on same network	262
13.3.4	Example with 2 group	262
13.3.5	Example with 3 groups	263
13.4	Extended Syntax (for network or group)	264
13.4.1	Network	264
13.4.2	Group	265
13.5	Using Sprites	266
13.6	Using OpenIconic	267
13.7	Same nodes on more than two networks	268
13.8	Peer networks	269
13.9	Peer networks and group	269
13.9.1	Without group	269
13.9.2	Group on first	270
13.9.3	Group on second	271
13.9.4	Group on third	272



13.10 Add title, caption, header, footer or legend on network diagram	273
13.11 With or without shadow	274
13.11.1 With shadow (by default)	274
13.11.2 Without shadow	274
13.12 Change width of the networks	275
13.13 Other internal networks	278
13.14 Using (global) style	279
13.14.1 Without style (<i>by default</i>)	279
13.14.2 With style	280
13.15 Appendix: Test of all shapes on Network diagram (nwdiag)	281
14 Salt (Wireframe)	284
14.1 Basic widgets	284
14.2 Text area	284
14.3 Open, close dropdownlist	285
14.4 Using grid [and #, !, -, +]	285
14.5 Group box [^]	286
14.6 Using separator [., ==, ~~, -]	286
14.7 Tree widget [T]	287
14.8 Tree table [T]	287
14.9 Enclosing brackets [{, }]	289
14.10 Adding tabs [/]	289
14.11 Using menu [*]	290
14.12 Advanced table	291
14.13 Scroll Bars [S, SI, S-]	292
14.14 Colors	292
14.15 Creole on Salt	293
14.16 Pseudo sprite [«, »]	295
14.17 OpenIconic	295
14.18 Add title, header, footer, caption or legend	296
14.19 Zoom, DPI	297
14.19.1 Whitout zoom (by default)	297
14.19.2 Scale	297
14.19.3 DPI	297
14.20 Include Salt "on activity diagram"	298
14.21 Include salt "on while condition of activity diagram"	300
14.22 Include salt "on repeat while condition of activity diagram"	301
14.23 Skinparam	302
14.24 Style	303
15 Archimate Diagram	304
15.1 Archimate keyword	304
15.2 Defining Junctions	304
15.3 Example 1	305
15.4 Example 2	306
15.5 List possible sprites	307
15.6 ArchiMate Macros	307
15.6.1 Archimate Macros and Library	307
15.6.2 Archimate elements	307
15.6.3 Archimate relationships	308
15.6.4 Appendix: Examples of all Archimate RelationTypes	309
16 Gantt Diagram	313
16.1 Declaring tasks	313
16.1.1 Duration	313
16.1.2 Start	313
16.1.3 End	314
16.1.4 Start/End	314
16.2 One-line declaration (with the and conjunction)	315



16.3 Adding constraints	315
16.4 Short names	315
16.5 Customize colors	316
16.6 Completion status	316
16.6.1 Adding completion depending percentage	316
16.6.2 Change colour of completion (by style)	316
16.7 Milestone	317
16.7.1 Relative milestone (use of constraints)	318
16.7.2 Absolute milestone (use of fixed date)	318
16.7.3 Milestone of maximum end of tasks	318
16.8 Hyperlinks	318
16.9 Calendar	319
16.10 Coloring days	319
16.11 Changing scale	319
16.11.1 Daily (<i>by default</i>)	320
16.11.2 Weekly	320
16.11.3 Monthly	321
16.11.4 Quarterly	321
16.11.5 Yearly	322
16.12 Zoom (example for all scale)	322
16.12.1 Zoom on weekly scale	322
16.12.2 Without zoom	322
16.12.3 With zoom	322
16.12.4 Zoom on weekly scale	323
16.12.5 Without zoom	323
16.12.6 With zoom	323
16.12.7 Zoom on monthly scale	324
16.12.8 Without zoom	324
16.12.9 With zoom	324
16.12.10 Zoom on quarterly scale	324
16.12.11 Without zoom	324
16.12.12 With zoom	325
16.12.13 Zoom on yearly scale	325
16.12.14 Without zoom	325
16.12.15 With zoom	325
16.13 Weekscale with Weeknumbers or Calendar Date	326
16.13.1 With Weeknumbers (<i>by default</i>)	326
16.13.2 With Calendar Date	326
16.14 Close day	326
16.15 Definition of a week depending of closed days	327
16.16 Working days	328
16.17 Simplified task succession	328
16.18 Working with resources	329
16.19 Hide resources	329
16.19.1 Without any hiding (<i>by default</i>)	329
16.19.2 Hide resources names	330
16.19.3 Hide resources footbox	330
16.19.4 Hide the both (resources names and resources footbox)	330
16.20 Horizontal Separator	331
16.21 Vertical Separator	331
16.22 Complex example	331
16.23 Comments	332
16.24 Using style	332
16.24.1 Without style (<i>by default</i>)	332
16.24.2 With style	333
16.24.3 With style (full example)	334
16.24.4 Clean style	336
16.25 Add notes	337



16.26	Pause tasks	339
16.27	Change link colors	340
16.28	Tasks or Milestones on the same line	341
16.29	Highlight today	341
16.30	Task between two milestones	341
16.31	Grammar and verbal form	342
16.32	Add title, header, footer, caption or legend	342
16.33	Removing Foot Boxes (example for all scale)	342
16.34	Language of the calendar	344
16.34.1	English (<i>en, by default</i>)	344
16.34.2	Deutsch (de)	345
16.34.3	Japanese (ja)	345
16.34.4	Chinese (zh)	345
16.34.5	Korean (ko)	346
16.35	Delete Tasks or Milestones	346
16.36	Start a project, a task or a milestone a number of days before or after today	347
16.37	Change Label position	347
16.37.1	The labels are near elements (<i>by default</i>)	347
16.37.2	Label on first column	348
16.37.3	Label on last column	349
17	MindMap	351
17.1	Sintaxis OrgMode	351
17.2	Markdown syntax	351
17.3	Notación aritmética	352
17.4	Multilines	352
17.5	Multiroot Mindmap	354
17.6	Colors	354
17.6.1	With inline color	354
17.6.2	With style color	355
17.7	Removing box	357
17.8	Changing diagram direction	358
17.9	Complete example	359
17.10	Changing style	360
17.10.1	node, depth	360
17.10.2	boxless	361
17.11	Word Wrap	361
17.12	Creole on Mindmap diagram	362
18	Work Breakdown Structure (WBS)	365
18.1	OrgMode syntax	365
18.2	Change direction	365
18.3	Arithmetic notation	366
18.4	Multilines	366
18.5	Removing box	367
18.5.1	Boxless on Arithmetic notation	367
18.5.2	Several boxless node	367
18.5.3	All boxless node	368
18.5.4	Boxless on OrgMode syntax	368
18.5.5	Several boxless node	368
18.5.6	All boxless node	369
18.6	Colors (with inline or style color)	369
18.7	Using style	371
18.8	Word Wrap	372
18.9	Add arrows between WBS elements	373
18.10	Creole on WBS diagram	374
19	Maths	377
19.1	Standalone diagram	378



19.2 How is this working?	378
20 Diagrama de Relación de Entidades	379
20.1 Information Engineering Relations	379
20.2 Entities	379
20.3 Ejemplo completo	380
21 Comandos comunes en PlantUML	382
21.0.1 Elementos globales	382
21.0.2 Descripción de la sintaxis criolla	382
21.0.3 Comando de Control de Estilo	382
21.1 Comments	382
21.1.1 Simple comment	382
21.1.2 Block comment	382
21.1.3 Full example	383
21.2 Zoom	383
21.3 Title	384
21.4 Caption	385
21.5 Footer and header	385
21.6 Legend the diagram	386
21.7 Appendix: Examples on all diagram	386
21.7.1 Activity	386
21.7.2 Archimate	387
21.7.3 Class	388
21.7.4 Component, Deployment, Use-Case	388
21.7.5 Gantt project planning	389
21.7.6 Object	389
21.7.7 MindMap	390
21.7.8 Network (nwdiag)	391
21.7.9 Sequence	391
21.7.10 State	392
21.7.11 Timing	393
21.7.12 Work Breakdown Structure (WBS)	393
21.7.13 Wireframe (SALT)	394
21.8 Appendix: Examples on all diagram with style	395
21.8.1 Activity	395
21.8.2 Archimate	397
21.8.3 Class	398
21.8.4 Component, Deployment, Use-Case	400
21.8.5 Gantt project planning	401
21.8.6 Object	403
21.8.7 MindMap	404
21.8.8 Network (nwdiag)	405
21.8.9 Sequence	407
21.8.10 State	408
21.8.11 Timing	410
21.8.12 Work Breakdown Structure (WBS)	411
21.8.13 Wireframe (SALT)	412
21.9 Mainframe	413
21.10 Appendix: Examples of Mainframe on all diagram	414
21.10.1 Activity	414
21.10.2 Archimate	414
21.10.3 Class	415
21.10.4 Component, Deployment, Use-Case	415
21.10.5 Gantt project planning	415
21.10.6 Object	416
21.10.7 MindMap	416
21.10.8 Network (nwdiag)	416
21.10.9 Sequence	417



21.10.1\$state	417
21.10.1TIming	417
21.10.1Work Breakdown Structure (WBS)	418
21.10.1Wireframe (SALT)	418
21.11Appendix: Examples of title, header, footer, caption, legend and mainframe on all diagram	419
21.11.1 Activity	419
21.11.2 Archimate	419
21.11.3 Class	420
21.11.4 Component, Deployment, Use-Case	421
21.11.5 Gantt project planning	421
21.11.6 Object	422
21.11.7 MindMap	423
21.11.8 Network (nwdiag)	423
21.11.9 Sequence	424
21.11.1\$state	425
21.11.1TIming	425
21.11.1Work Breakdown Structure (WBS)	426
21.11.1Wireframe (SALT)	427
22 Creole	429
22.1 Emphasized text	429
22.2 Lists	429
22.3 Escape character	430
22.4 Headings	430
22.5 Emoji	431
22.5.1 Unicode block 26	431
22.6 Horizontal lines	432
22.7 Links	433
22.8 Code	433
22.9 Table	434
22.9.1 Create a table	434
22.9.2 Add color on rows or cells	435
22.9.3 Add color on border and text	435
22.9.4 No border or same color as the background	435
22.9.5 Bold header or not	435
22.10Tree	436
22.11Special characters	438
22.12Legacy HTML	439
22.12.1 Common HTML element	440
22.12.2 Subscript and Superscript element [sub, sup]	441
22.13OpenIconic	441
22.14Appendix: Examples of "Creole List" on all diagrams	442
22.14.1 Activity	442
22.14.2 Class	443
22.14.3 Component, Deployment, Use-Case	444
22.14.4 Gantt project planning	445
22.14.5 Object	445
22.14.6 MindMap	446
22.14.7 Network (nwdiag)	446
22.14.8 Note	447
22.14.9 Sequence	447
22.14.1\$state	448
22.14.1WBS	449
22.15Appendix: Examples of "Creole horizontal lines" on all diagrams	450
22.15.1 Activity	450
22.15.2 Class	451
22.15.3 Component, Deployment, Use-Case	451
22.15.4 Gantt project planning	453



22.15.5 Object	453
22.15.6 MindMap	454
22.15.7 Network (nwdiag)	454
22.15.8 Note	455
22.15.9 Sequence	455
22.15.10 State	456
22.15.11 WBS	457
22.16 Style equivalent (between Creole and HTML)	458
23 Defining and using sprites	460
23.1 Inline SVG sprite	461
23.2 Changing colors	461
23.3 Encoding Sprite	462
23.4 Importing Sprite	462
23.5 Examples	462
23.6 StdLib	463
23.7 Listing Sprites	463
24 Skinparam command	465
24.1 Usage	465
24.2 Nested	465
24.3 Black and White	465
24.4 Shadowing	466
24.5 Reverse colors	466
24.6 Colors	467
24.7 Font color, name and size	468
24.8 Text Alignment	468
24.9 Examples	469
24.10 List of all skinparam parameters	473
24.10.1 Command Line: -language command	473
24.10.2 Command: help skinparams	473
24.10.3 Command: skinparameters	473
24.10.4 All Skin Parameters on the Ashley's PlantUML Doc	475
25 Preprocessing	476
25.1 Variable definition [=, ?=]	476
25.2 Boolean expression	477
25.2.1 Boolean representation [0 is false]	477
25.2.2 Boolean operation and operator [&&, , ()]	477
25.2.3 Boolean builtin functions [%false(), %true(), %not(<exp>)]	477
25.3 Conditions [!if, !else, !elseif, !endif]	477
25.4 While loop [!while, !endwhile]	478
25.4.1 While loop (on Activity diagram)	478
25.4.2 While loop (on Mindmap diagram)	479
25.4.3 While loop (on Component/Deployment diagram)	480
25.5 Procedure [!procedure, !endprocedure]	480
25.6 Return function [!function, !endfunction]	481
25.7 Default argument value	482
25.8 Unquoted procedure or function [!unquoted]	483
25.9 Keywords arguments	484
25.10 Including files or URL [!include, !include_many, !include_once]	484
25.11 Including Subpart [!startsub, !endsub, !includesub]	485
25.12 Builtin functions [%]	485
25.13 Logging [!log]	486
25.14 Memory dump [!dump_memory]	487
25.15 Assertion [!assert]	487
25.16 Building custom library [!import, !include]	488
25.17 Search path	488
25.18 Argument concatenation [##]	488



25.19	Dynamic invocation [%invoke_procedure(), %call_user_func()]	489
25.20	Evaluation of addition depending of data types [+]	490
25.21	Preprocessing JSON	490
25.22	Including theme [!theme]	490
25.23	Migration notes	491
25.24	%Splitstr builtin function	491
26	Unicode	492
26.1	Examples	492
26.2	Charset	494
26.3	Using Unicode Character on PlantUML	494
27	PlantUML Standard Library	495
27.0.1	Standard Library Overview	495
27.0.2	Contribution from the Community	495
27.1	List of Standard Library	495
27.2	ArchiMate [archimate]	497
27.2.1	List possible sprites	498
27.3	Amazon Labs AWS Library [awslib]	499
27.4	Azure library [azure]	500
27.5	C4 Library [C4]	501
27.6	Cloud Insight [cloudinsight]	501
27.7	Cloudogu [cloudogu]	502
27.8	Elastic library [elastic]	503
27.9	Google Material Icons [material]	505
27.10	Kubernetes [kubernetes]	506
27.11	Logos [logos]	507
27.12	Office [office]	509
27.13	Open Security Architecture (OSA) [osa]	511
27.14	Tupadr3 library [tupadr3]	514
27.15	AWS library [aws]	515

