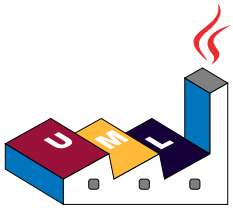
Dessiner de l'UML avec PlantUML



Guide de référence du langage PlantUML

(Version 1.2021.2)

PlantUML est un composant qui permet de dessiner rapidement des:

- diagrammes de séquence
- diagrammes de cas d'utilisation
- diagrammes de classes
- diagrammes d'objet
- diagrammes d'activité
- diagrammes de composant
- diagrammes de déploiement
- diagrammes d'état
- diagrammes de temps

Certains autres diagrammes (hors UML) sont aussi possibles:

- données au format JSON
- données au format YAML
- diagrammes de réseaux (nwdiag)
- maquettes d'interface graphique (salt)
- diagrammes Archimate
- diagrammes de langage de description et de spécification (SDL)
- diagrammes ditaa
- diagrammes de Gantt
- diagrammes d'idées (mindmap)
- organigramme (Work Breakdown Structure)
- notation mathématique avec AsciiMath ou JLaTeXMath
- diagrammes entité relation (ER/IE)

Les diagrammes sont définis à l'aide d'un langage simple et intuitif.

1 Diagramme de séquence

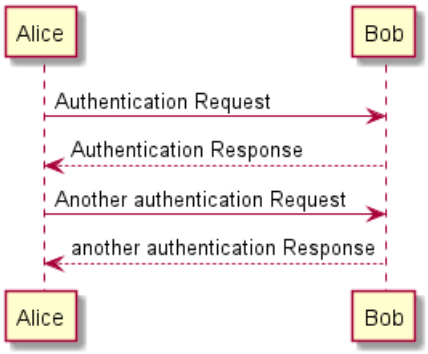
1.1 Exemples de base

Le symbole `->` est utilisé pour dessiner un message entre deux participants. Les participants n'ont pas besoin d'être explicitement déclarés.

Pour avoir une flèche en pointillée, il faut utiliser `-->`.

Il est aussi possible d'utiliser `<-` et `<--`. Cela ne change pas le dessin, mais cela peut améliorer la lisibilité du texte source. Ceci est uniquement vrai pour les diagrammes de séquences, les règles sont différentes pour les autres diagrammes.

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

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



1.2 Déclaration de participants

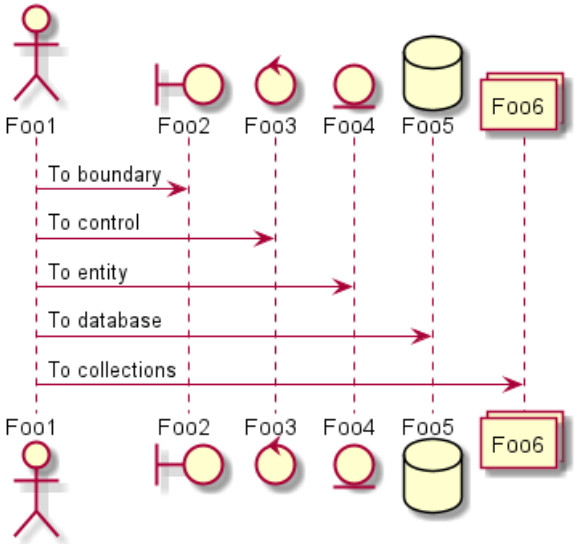
Il est possible de changer l'ordre des participants à l'aide du mot clé `participant`.

Il est aussi possible d'utiliser d'autres mot-clés pour déclarer un participant :

- `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
```

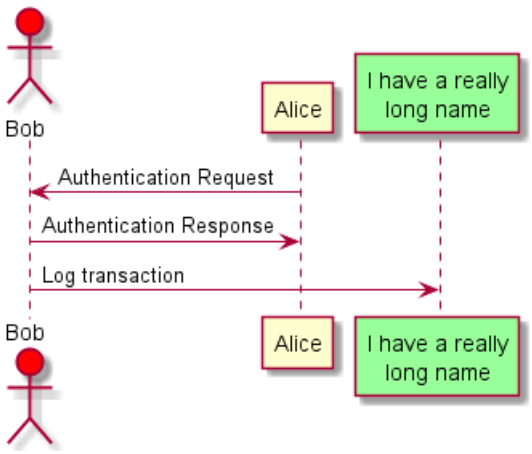
```
@enduml
```



On peut aussi utiliser un nom court à l'aide grâce au mot-clé `as`.

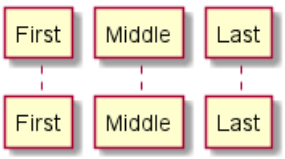
La couleur d'un acteur ou d'un participant peut être définie avec son code ou son nom HTML.

```
@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
```



Vous pouvez utiliser le mot-clé `order` pour modifier l'ordre des participants

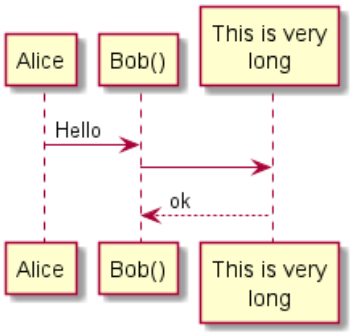
```
@startuml
participant Last order 30
participant Middle order 20
participant First order 10
@enduml
```



1.3 Caractères non alphanumérique dans les participants

Si vous voulez mettre des charactères non alphanumériques, il est possible d'utiliser des guillemets. Et on peut utiliser le mot clé `as` pour définir un alias pour ces participants.

```
@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
```

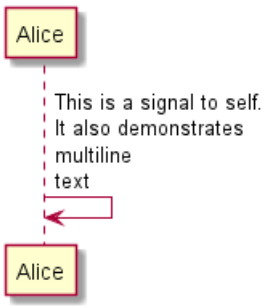


1.4 Message à soi-même

Un participant peut très bien s'envoyer un message.

Il est possible de mettre un message sur plusieurs lignes grâce à .

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



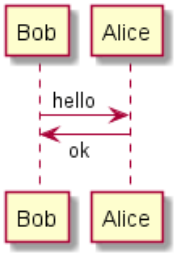
1.5 Text alignment

1.5.1 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
```



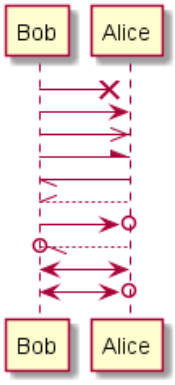
TODO: TODO Link to Text Alignment on skinparam page.

1.6 Autre style de flèches

Vous pouvez changer les flèches de plusieurs façons :

- Pour indiquer un message perdu, terminer la flèche avec `x`
- Utiliser `\` ou `/` à la place de `<` ou `>` pour avoir seulement la partie supérieure ou inférieure de la flèche.
- Doubler un des caractères (par exemple, `>>` ou `//`)pour avoir une flèche plus fine.
- Utiliser `--` à la place de `-` pour avoir des pointillés.
- Utiliser ”o” après la flèche
- Utiliser une flèche bi-directionnelle `<->`

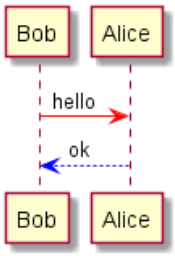
```
@startuml
Bob ->x Alice
Bob -> Alice
Bob ->> Alice
Bob -\ Alice
Bob \\- Alice
Bob //-- Alice

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

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



1.7 Changer la couleur des flèches

Changer la couleur d’une flèche ainsi:

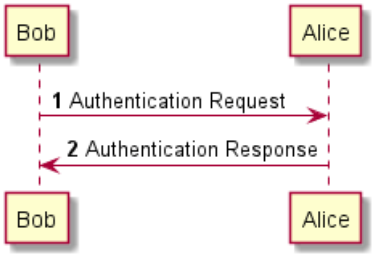
```
@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml
```



1.8 Numérotation automatique des messages

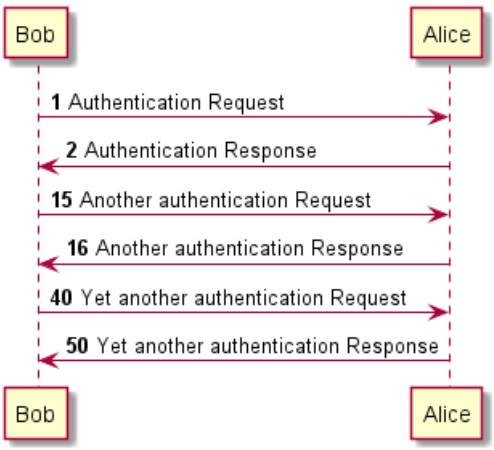
Le mot clé `autonumber` est utilisé pour ajouter automatiquement des numéros aux messages.

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



Spécifier le numéro de départ avec `autonumber //start//` , et l'incrément avec `autonumber //start// //increment//`.

```
@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
```

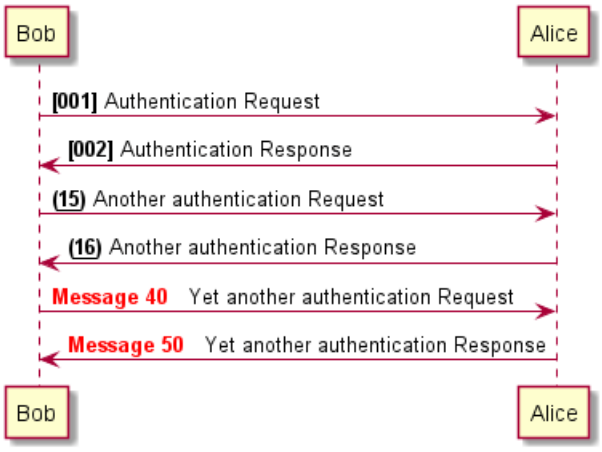


Spécifier le format d'un nombre entre guillemets anglais.

Le formatage est fait par la classe `DecimalFormat` (`0` signifie un chiffre, `#` signifie un chiffre ou zéro si absent).

Des balises HTML sont permises dans le format.

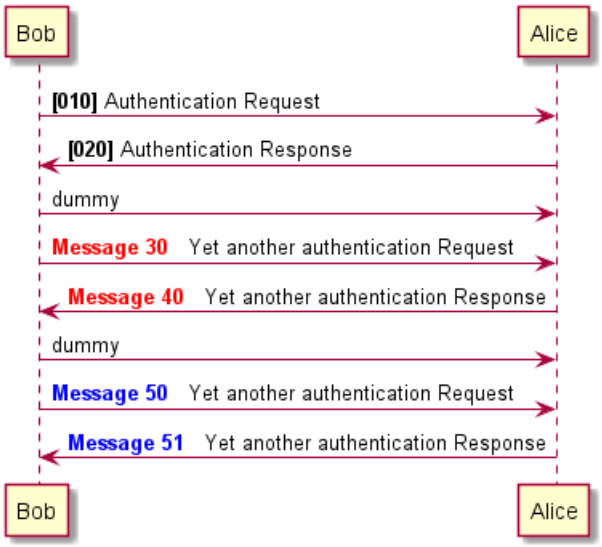
```
@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

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

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

@enduml
```



Vous pouvez utiliser `autonumber //stop//` et `autonumber resume //increment// //format//` pour respectivement arrêter et reprendre la numérotation automatique.

```
@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
```

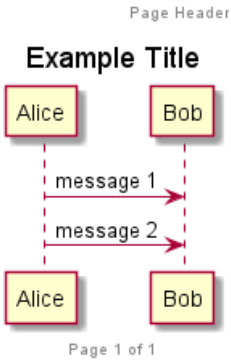


1.9 Page Title, Header and Footer

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
```



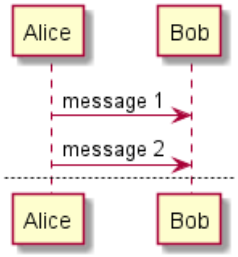
1.10 Découper un diagramme

Le mot clé `newpage` est utilisé pour découper un digramme en plusieurs images.

Vous pouvez mettre un titre pour la nouvelle page juste après le mot clé `newpage`.

Ceci est très pratique pour mettre de très longs digrammes sur plusieurs pages.

```
@startuml

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

newpage

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

newpage A title for the\nlast page

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



1.11 Regrouper les messages (cadres UML)

Il est possible de regrouper les messages dans un cadre UML à l'aide d'un des mot clés suivants:

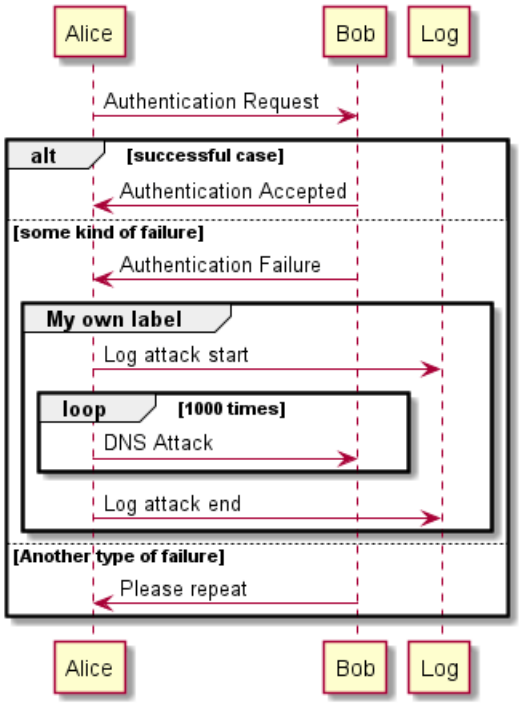
- `alt/else`
- `opt`
- `loop`
- `par`
- `break`
- `critical`
- `group`, suivi par le texte à afficher

Il est aussi possible de mettre un texte à afficher dans l'entête. Le mot-clé `end` est utiliser pour fermer le groupe. Il est aussi possible d'imbriquer les groupes.

Terminer le cadre avec le mot-clé `end`.

Il est possible d'imbriquer les cadres.

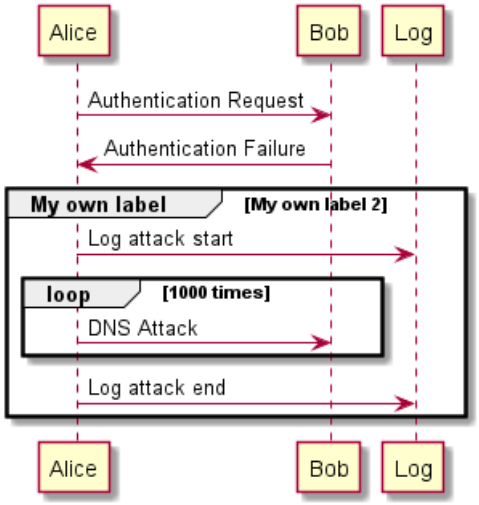
```
@startuml
Alice -> Bob: Authentication Request

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
```



1.12 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
```



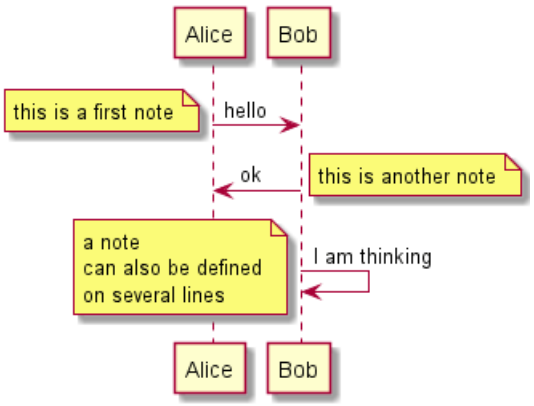
[Ref. QA-2503]

1.13 Note sur les messages

Pour attacher une note à un message, utiliser les mots-clés `note left` (pour une note à gauche) ou `note right` (pour une note à droite) *juste après le message*.

Il est possible d'avoir une note sur plusieurs lignes avec le mot clé `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
```



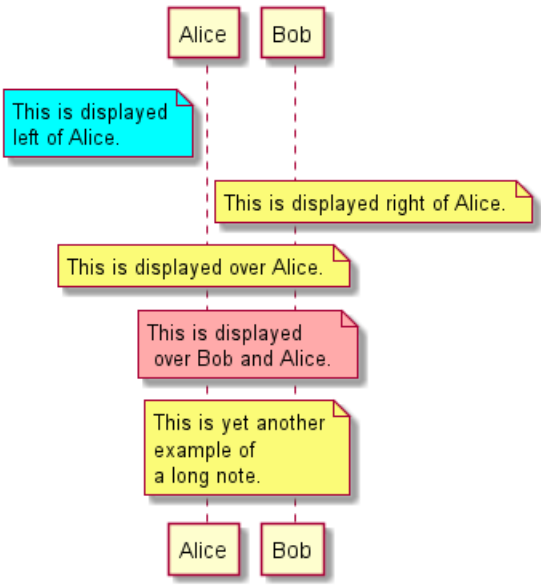
1.14 Encore plus de notes

Il est aussi possible de mettre des notes placées par rapport aux participants.

Il est aussi possible de faire ressortir une note en changeant sa couleur de fond.

On peut aussi avoir des notes sur plusieurs lignes à l'aide du mot clé `end note`.

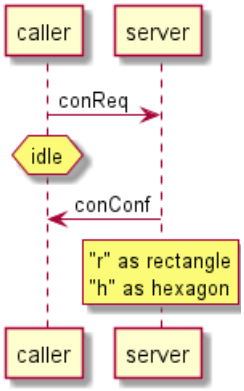
```
@startuml
participant Alice
participant Bob
note left of Alice #aqua
This is displayed
left of Alice.
end note

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
```



1.15 Changer l'aspect des notes

Vous pouvez préciser la forme géométrique des notes. (`rnote` : rectangulaire, ou `hnote` : hexagonale)

```
@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)]

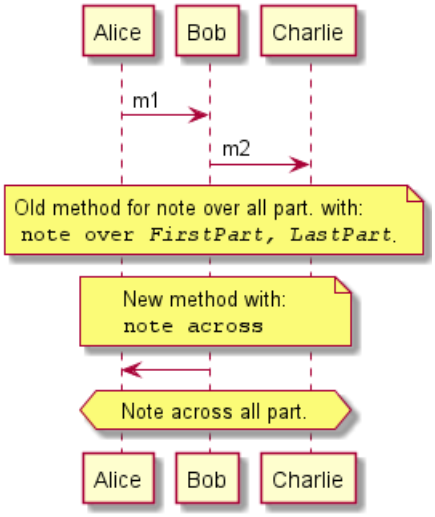
1.16 Note over all participants [across]

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
```



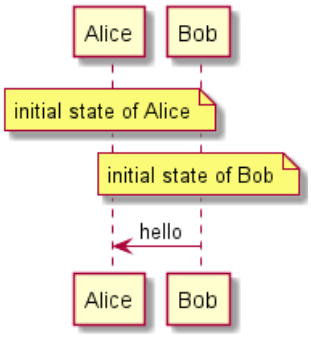
[Ref. QA-9738]

1.17 Several notes aligned at the same level [/]

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

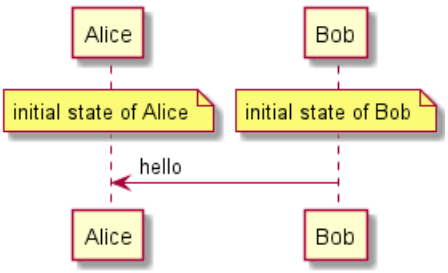
- without `/` *(by default, the notes are not aligned)*

```
@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
```

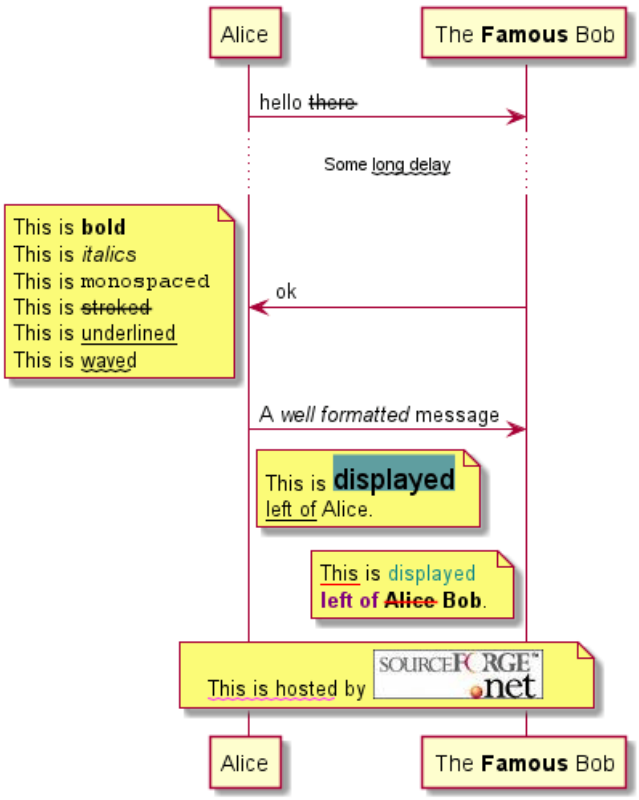


[Ref. QA-354]

1.18 Créole (langage de balisage léger) et HTML

Il est également possible d'utiliser le formatage créole (langage de balisage léger):

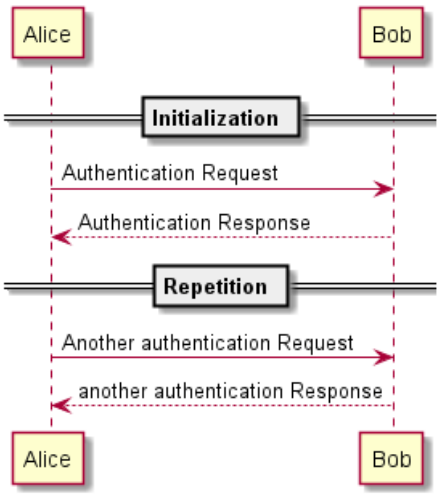
```
@startuml
participant Alice
participant "The **Famous** Bob" as Bob

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
```



1.19 Séparation

Si vous voulez, vous pouvez séparer le diagramme avec l'aide de "==" en étapes logiques.

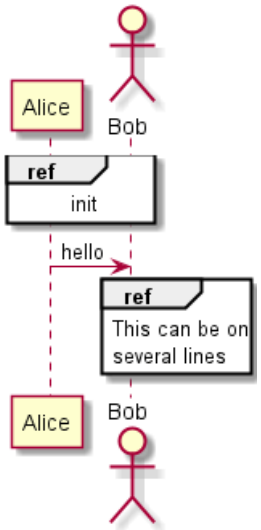
```
@startuml

== Initialization ==

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

== Repetition ==

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

@enduml
```



1.20 Référence

Vous pouvez ajouter des références dans un diagramme, en utilisant le mot-clé `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
```

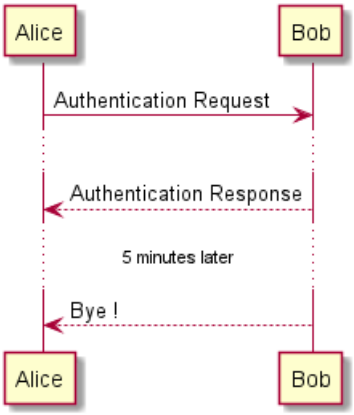


1.21 Retard

Utiliser `...` pour indiquer le passage de temps arbitraire dans le diagramme. Un message peut être associé à un retard.

```
@startuml
```

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

@enduml
```

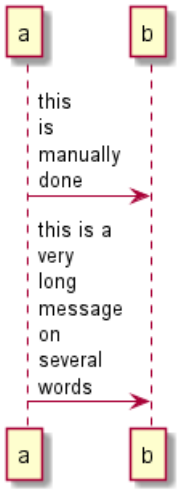


1.22 Text wrapping

To break long messages, you can manually add 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
```



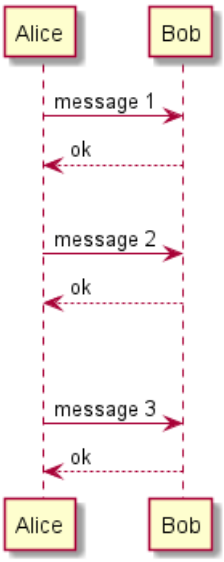
1.23 Séparation verticale

Utiliser `|||` pour créer un espace vertical dans le diagramme.

Il est également possible de spécifier un nombre de pixels pour la séparation verticale.

```
@startuml
```

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

@enduml
```



1.24 Lignes de vie

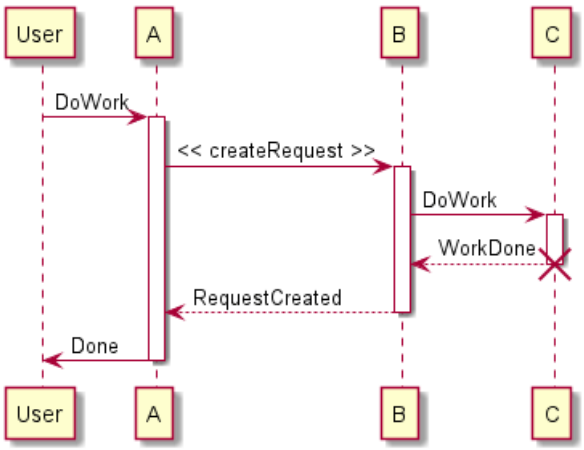
Vous pouvez utiliser `activate` et `deactivate` pour marquer l'activation des participants.

Une fois qu'un participant est activé, sa ligne de vie apparaît.

Les ordres `activate` et `deactivate` s'applique sur le message situé juste avant.

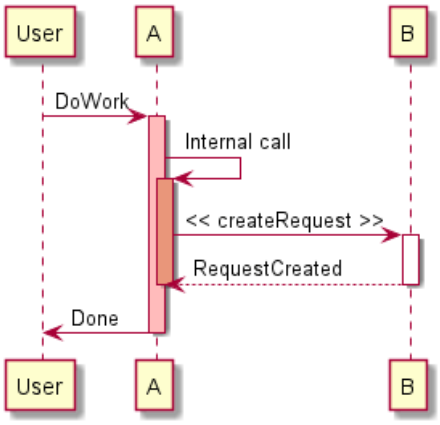
Le mot clé `destroy` sert à montrer la fin de vie d'un participant.

```
@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
```



Les lignes de vie peuvent être imbriquées, et il est possible de les colorer.

```
@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
```

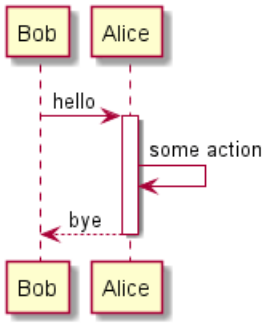


1.25 Return

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
```

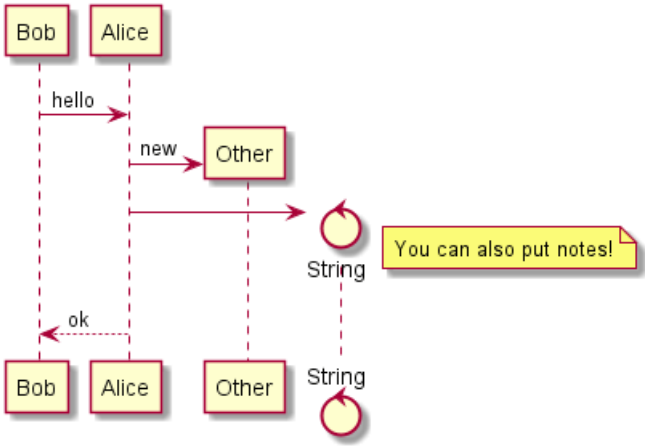
```
activate Alice
Alice -> Alice : some action
return bye
@enduml
```



1.26 Création de participants.

Vous pouvez utiliser le mot clé `create` juste avant la première réception d'un message pour montrer que le message en question est une *création* d'un nouvelle objet.

```
@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
```



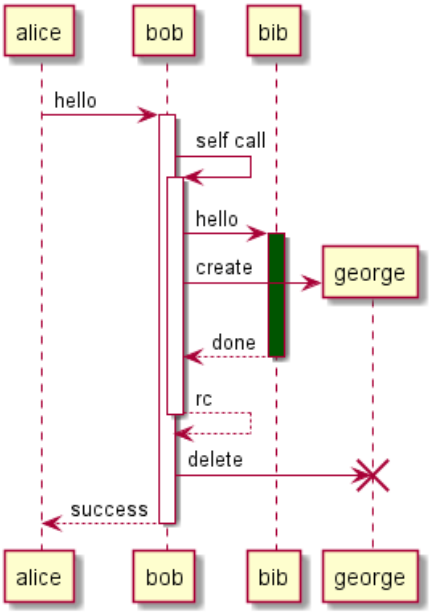
1.27 Shortcut syntax for activation, deactivation, creation

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

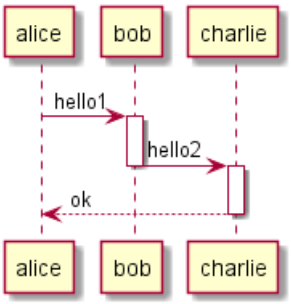
- `!!` Destroy an instance of the target

```
@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
```

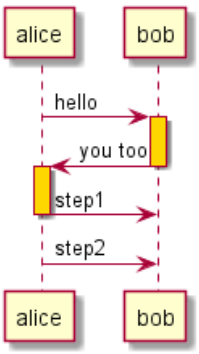


Then you can mix activation and deactivation, on same line:

```
@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
```



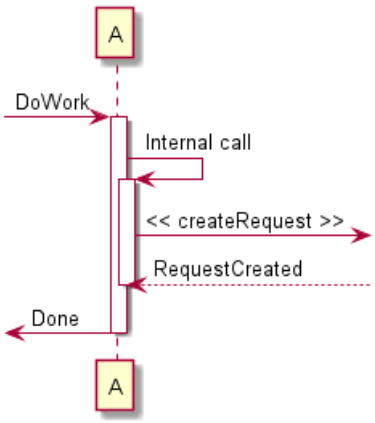
[Ref. QA-4834, QA-9573 and QA-13234]

1.28 Messages entrant et sortant

Vous pouvez utiliser des flèches qui viennent de la droite ou de la gauche pour dessiner un sous-diagramme.

Il faut utiliser des crochets pour indiquer la gauche "`[`" ou la droite "`]`" du diagramme.

```
@startuml
[-> A: DoWork

activate A

A -> A: Internal call
activate A

A ->] : << createRequest >>

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



Vous pouvez aussi utiliser la syntaxe suivante:

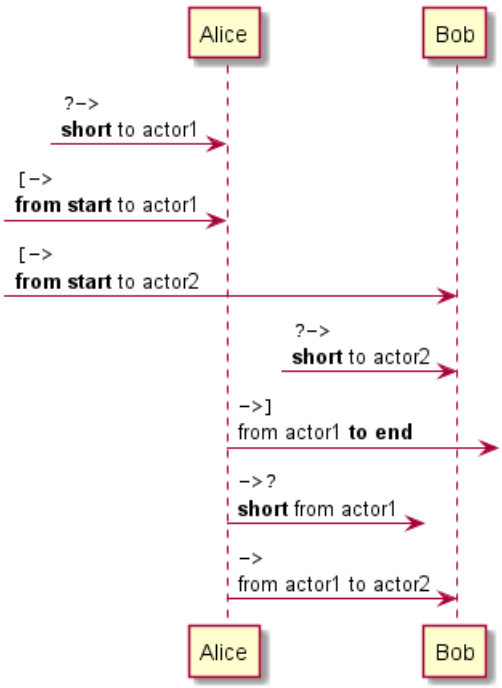
```
@startuml
[-> Bob
[o-> Bob
[o->o Bob
[x-> Bob

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

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

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

1.29 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
```

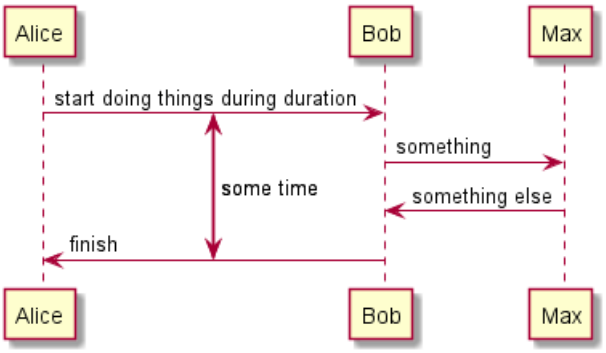


[Ref. QA-310]

1.30 Anchors and Duration

With `teoz` usage 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
```

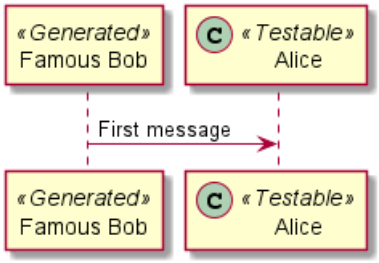


1.31 Stéréotypes et décoration

Il est possible de rajouter un stéréotype aux participants en utilisant "<<" et ">>".

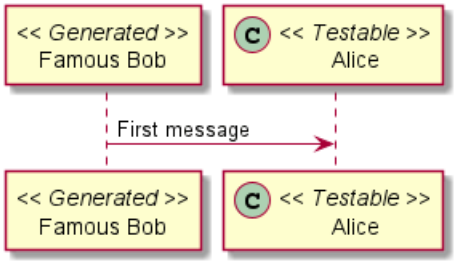
Dans le stéréotype, vous pouvez ajouter un caractère entouré d'un cercle coloré en utilisant la syntaxe `(X,couleur)`.

```
@startuml

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

Bob->Alice: First message

@enduml
```

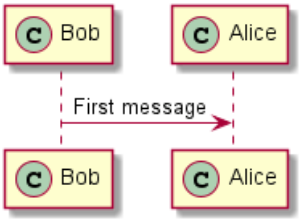


Par défaut, le caractère *guillemet* est utilisé pour afficher les stéréotypes. Vous pouvez changer ce comportement en utilisant la propriété skinparam `guillemet`:

```
@startuml

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

Bob->Alice: First message

@enduml
```



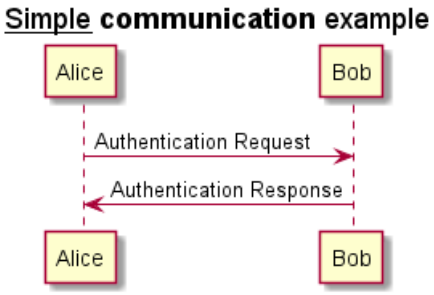

```
@startuml

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

Bob->Alice: First message

@enduml
```



1.32 Plus d'information sur les titres

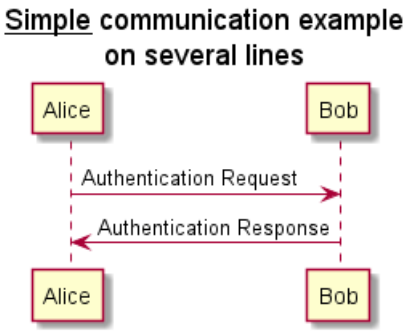
Vous pouvez utiliser le formatage creole dans le titre.

```
@startuml

title __Simple__ **communication** example

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

@enduml
```



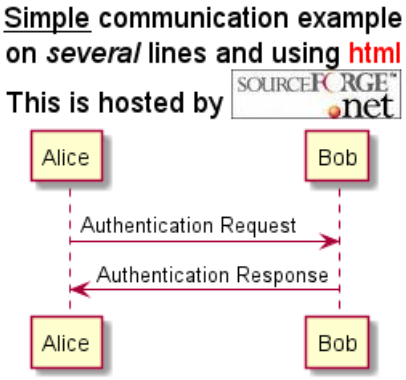
Vous pouvez mettre des retours à la ligne en utilisant dans la description.

```
@startuml

title __Simple__ communication example\non several lines

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

@enduml
```



Vous pouvez aussi mettre un titre sur plusieurs lignes à l'aide des mots-clé `title` et `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
```

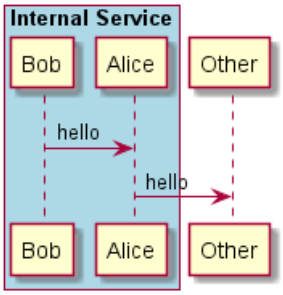


1.33 Cadre pour les participants

Il est possible de dessiner un cadre autour de certains participants, en utilisant les commandes `box` et `end box`.

Vous pouvez ajouter un titre ou bien une couleur de fond après le mot-clé `box`.

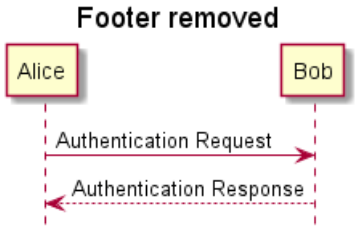
```
@startuml

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

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

@enduml
```



1.34 Supprimer les participants en pied de page

Vous pouvez utiliser le mot-clé `hide footbox` pour supprimer la partie basse du diagramme.

```
@startuml

hide footbox
title Footer removed

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

@enduml
```



1.35 Personnalisation

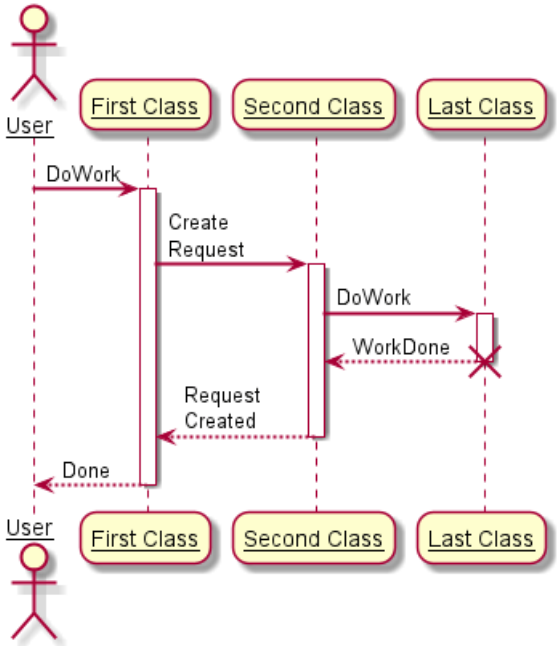
Utilisez la commande skinparam pour changer la couleur et la mise en forme du texte du schéma.

Vous pouvez utiliser cette commande :

- Dans la définition du diagramme, comme pour les autres commandes,
- Dans un fichier inclus,
- Dans un fichier de configuration, renseigné dans la ligne de commande ou la tâche ANT.

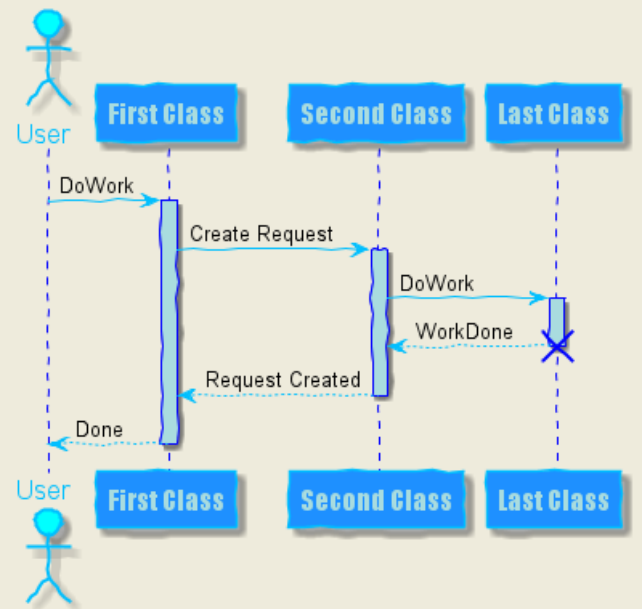
Vous pouvez aussi modifier d'autres paramètres pour le rendu, comme le montrent les exemples suivants:

```
@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

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
```

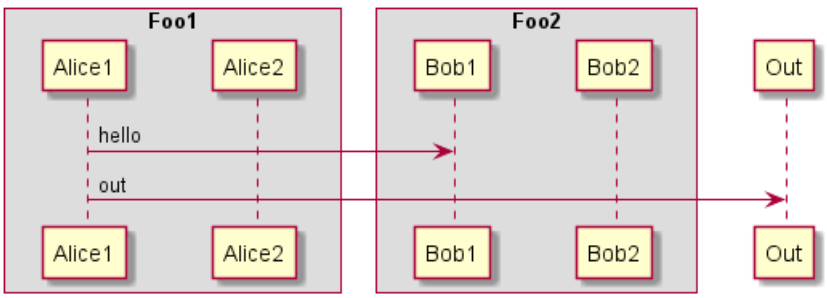
```
B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml
```



1.36 Changer le padding

Il est possible de changer certains paramètres du padding.

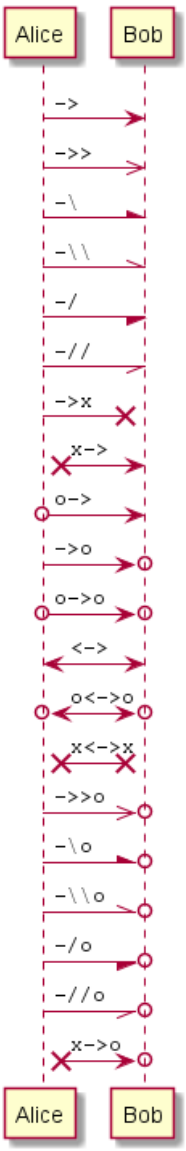
```
@startuml
skinparam ParticipantPadding 20
skinparam BoxPadding 10

box "Foo1"
participant Alice1
participant Alice2
end box
box "Foo2"
participant Bob1
participant Bob2
end box
Alice1 -> Bob1 : hello
Alice1 -> Out : out
@enduml
```



1.37 Appendix: Examples of all arrow type

1.37.1 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
```

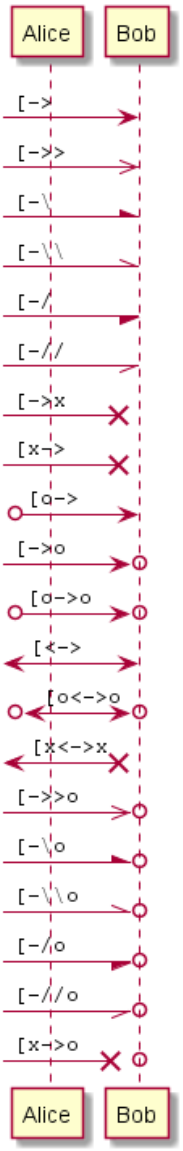


1.37.2 Incoming and outgoing messages (with '[', ']')

1.37.3 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 ""
```

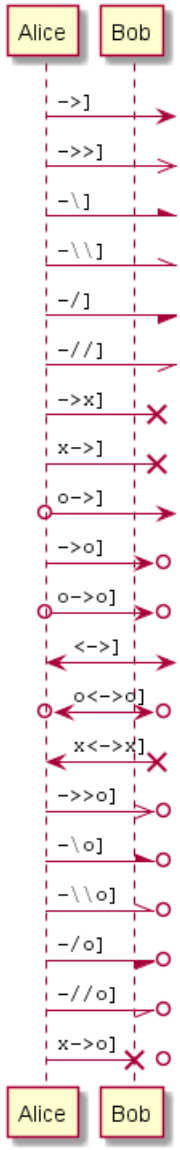
```
[-\o     b : ""[-\o  ""
[-\\o    b : ""[-\\\\o""
[-/o     b : ""[-/o  ""
[-//o    b : ""[-//o ""
[x->o    b : ""[x->o ""
@enduml
```



1.37.4 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
```

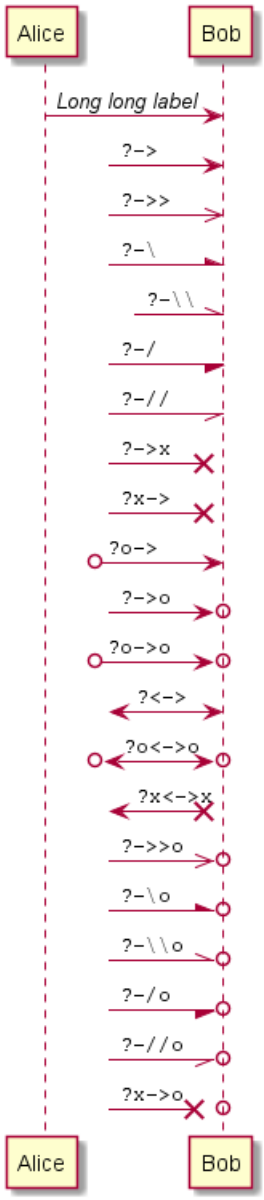


1.37.5 Short incoming and outgoing messages (with '?')

1.37.6 Short incoming (with '?')

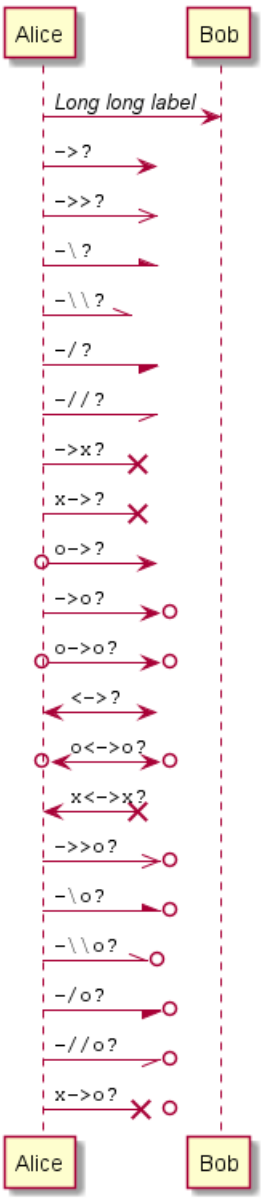
```
@startuml
participant Alice as a
participant Bob   as b
a ->      b : //Long long label//
?->       b : ""?->   ""
?->>      b : ""?->>  ""
```

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



1.37.7 Short outgoing (with '?')

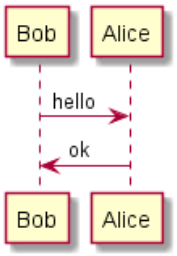
```
@startuml
participant Alice as a
participant Bob   as b
a ->     b : //Long long label//
a ->?      : ""->?   ""
a ->>?     : ""->>?  ""
a -\?      : ""-\?   ""
a -\\?     : ""-\\\\?""
a -/?      : ""-/?   ""
a -//?     : ""-//?  ""
a ->x?     : ""->x?  ""
a x->?     : ""x->?  ""
a o->?     : ""o->?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



1.38 Specific SkinParameter

1.38.1 By default

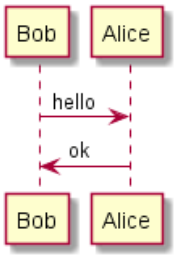
```
@startuml
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



1.38.2 LifelineStrategy

- nosolid *(by default)*

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

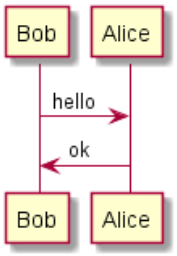


[Ref. QA-9016]

- 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
```



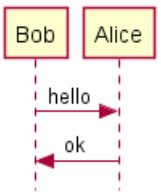
[Ref. QA-2794]

1.38.3 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
```

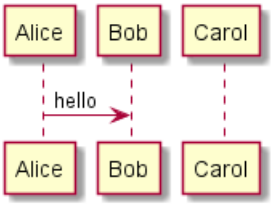


[Ref. QA-1047]

1.39 Hide unlinked participant

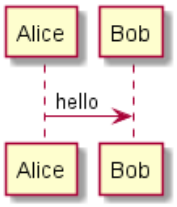
By default, all participants are displayed.

```
@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]

2 Diagramme de cas d'utilisation

Let's have few examples :

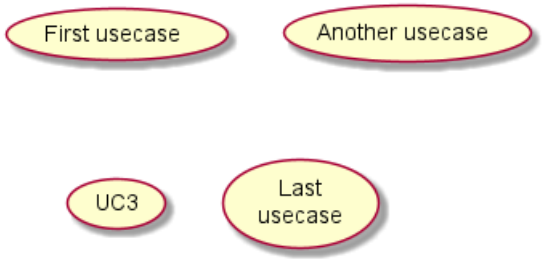
Note that you can disable the shadowing using the `skinparam shadowing false` command.

2.1 Cas d'utilisation

Les cas d'utilisation sont mis entre parenthèses (car deux parenthèses forment un ovale).

Vous pouvez aussi utiliser le mot-clé `usecase` pour définir un cas d'utilisation. Et vous pouvez définir un alias avec le mot-clé `as`. Cet alias sera ensuite utilisé lors de la définition des relations.

```
@startuml

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

@enduml
```



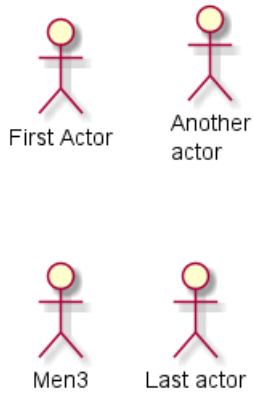
2.2 Acteurs

Un Acteur est encadré par des deux points.

Vous pouvez aussi utiliser le mot-clé `actor` pour définir un acteur. Et vous pouvez définir un alias avec le mot-clé `as`. Cet alias sera ensuite utilisé lors de la définition des relations.

Nous verrons que la définition des acteurs est optionnelle.

```
@startuml

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

@enduml
```



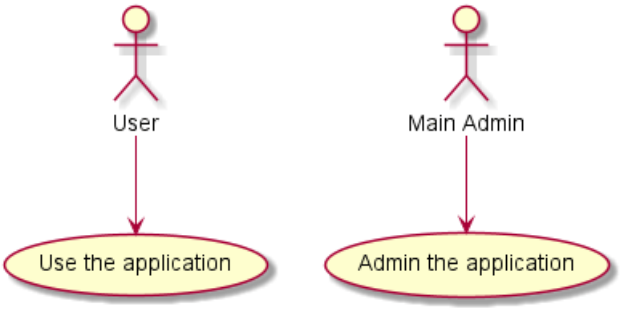
2.3 Change Actor style

You can change the actor style from stick man *(by default)* to:

- an awesome man with the `skinparam actorStyle awesome` command;
- a hollow man with the `skinparam actorStyle hollow` command.

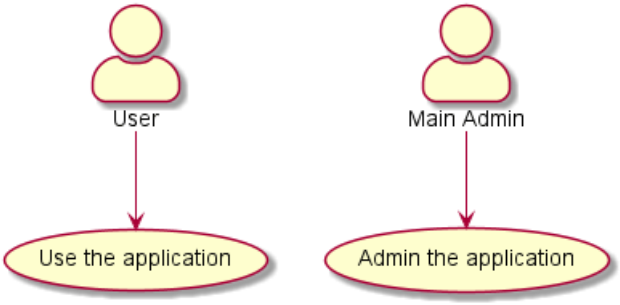
2.3.1 Stick man *(by default)*

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



2.3.2 Awesome man

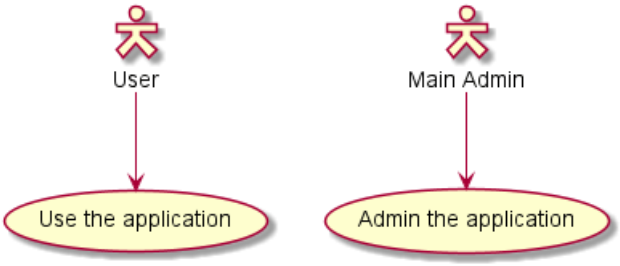
```
@startuml
skinparam actorStyle awesome
:User: --> (Use)
"Main Admin" as Admin
"Use the application" as (Use)
Admin --> (Admin the application)
@enduml
```



[Ref. QA-10493]

2.3.3 Hollow man

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



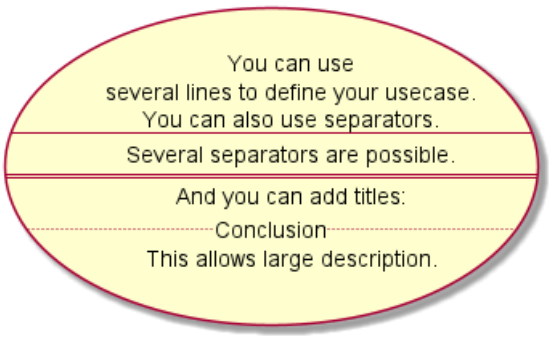
[Ref. PR#396]

2.4 Description des cas d'utilisation

Si vous voulez une description sur plusieurs lignes, vous pouvez utiliser des guillemets.

Vous pouvez aussi utiliser les séparateurs suivants: `--` `..` `==` `__`. Et vous pouvez mettre un titre dans les séparateurs.

```
@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
```

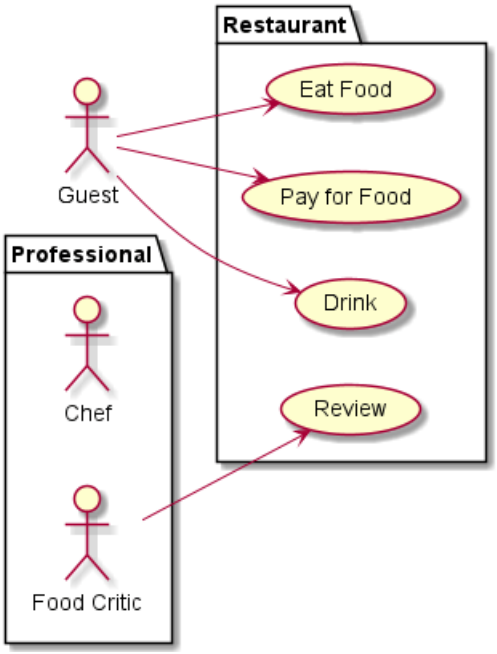


2.5 Use package

You can use packages to group actors or use cases.

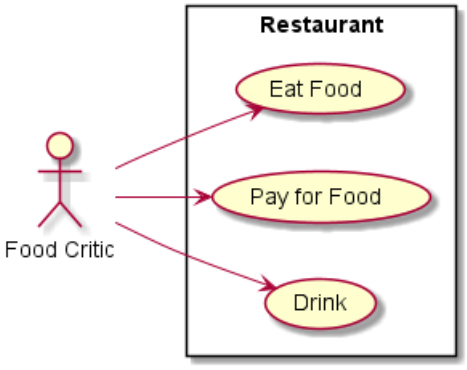
```
@startuml
left to right direction
actor Guest as g
package Professional {
  actor Chef as c
  actor "Food Critic" as fc
}
package Restaurant {
  usecase "Eat Food" as UC1
  usecase "Pay for Food" as UC2
  usecase "Drink" as UC3
  usecase "Review" as UC4
```

```
}
fc --> UC4
g --> UC1
g --> UC2
g --> UC3
@enduml
```



You can use `rectangle` to change the display of the package.

```
@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
```



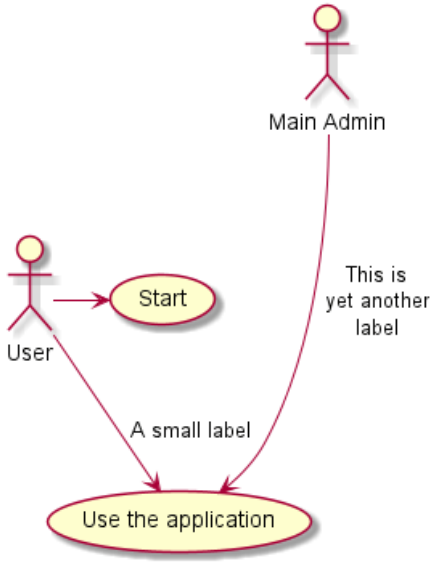
2.6 Exemples très simples

Pour lier les acteurs et les cas d'utilisation, la flèche `-->` est utilisée.

Plus il y a de tirets - dans la flèche, plus elle sera longue. Vous pouvez ajouter un libellé sur la flèche, en ajoutant un caractère : dans la définition de la flèche.

Dans cet exemple, vous voyez que *User* n'a pas été défini préalablement, et qu'il est implicitement reconnu comme acteur.

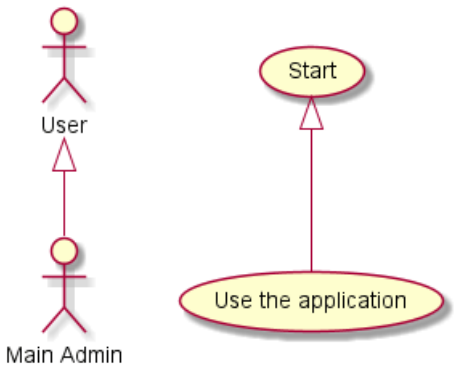
```
@startuml

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

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

@enduml
```



2.7 Héritage

Si un acteur ou un cas d'utilisation en étend un autre, vous pouvez utiliser le symbole <|--.

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

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

@enduml
```

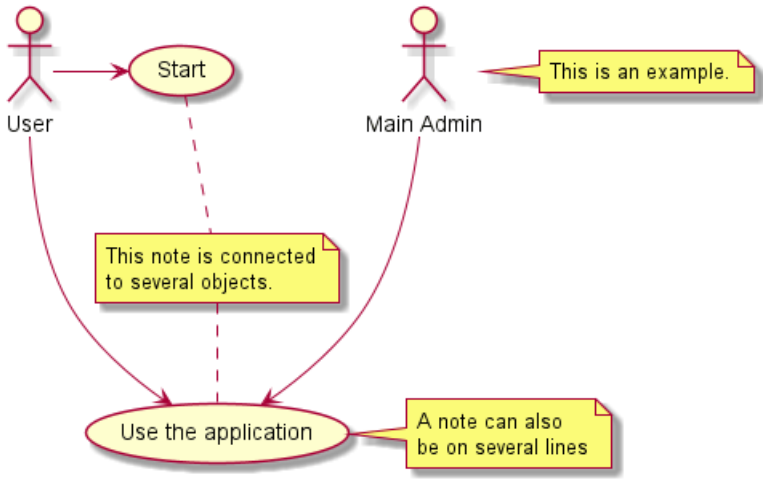


2.8 Notes

Vous pouvez utiliser les mots clés `note left of` , `note right of` , `note top of` , `note bottom of` pour définir les notes en relation avec un objet.

Une note peut également être définie seule avec des mots-clés, puis liée à d'autres objets en utilisant le symbole `..` .

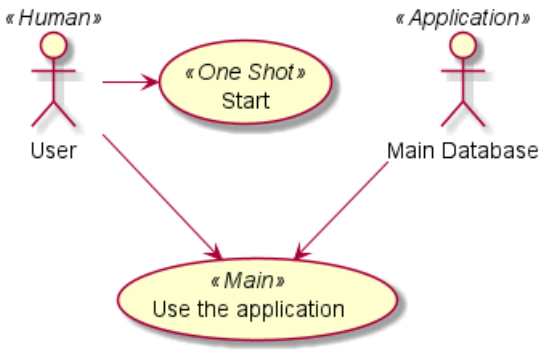
```
@startuml
:Main Admin: as Admin
(Use the application) as (Use)

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 Stéréotypes

Vous pouvez ajouter des stéréotypes à la définition des acteurs et des cas d'utilisation avec `<<` et `>>`.

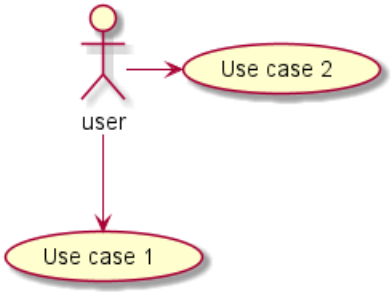
```
@startuml
User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>

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

MySql --> (Use)

@enduml
```



2.10 Changer les directions des flèches

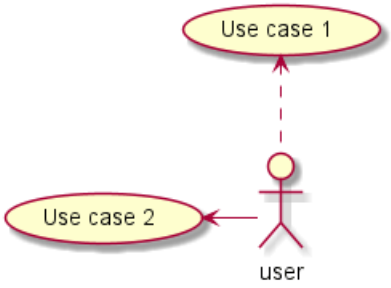
Par défaut, les liens entre les classes ont deux tirets `--` et sont orientés verticalement. Il est possible de mettre des liens horizontaux en mettant un seul tiret (ou un point) comme ceci:

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



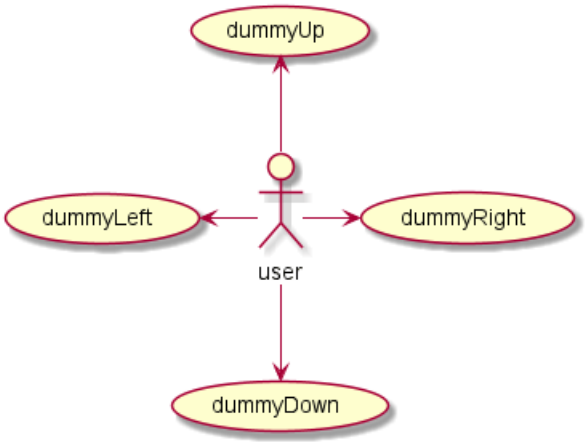
Vous pouvez aussi changer le sens en renversant le lien :

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



Il est possible de changer la direction d'une flèche en utilisant les mots-clé `left`, `right`, `up` ou `down` à l'intérieur de la flèche :

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



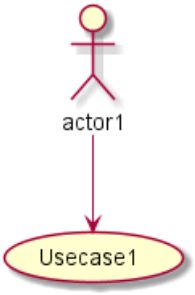
Vous pouvez abréger les noms des flèches en indiquant seulement le premier caractère de la direction (par exemple `-d-` pour `-down-`) ou les deux premiers caractères (`-do-`).

Il est conseillé de ne pas abuser de cette fonctionnalité : *Graphviz* qui donne d'assez bon résultats quoique non "garantis".

2.11 Découper les diagrames

Le mot-clé `newpage` est utilisé pour découper un diagrame en plusieurs images.

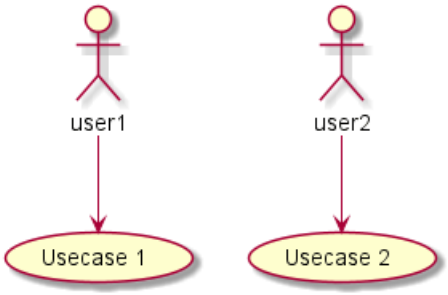
```
@startuml
:actor1: --> (Usecase1)
newpage
:actor2: --> (Usecase2)
@enduml
```



2.12 De droite à gauche

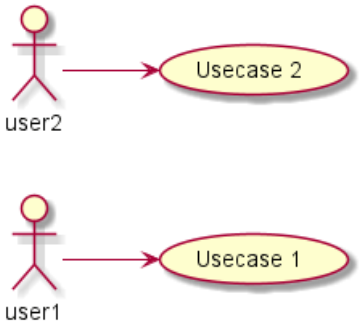
Le comportement général de construction des diagrammes est de haut en bas.

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

@enduml
```



Il est possible de changer pour aller plutôt de la droite vers la gauche avec la commande `left to right direction`. Le résultat est parfois meilleur dans ce cas.

```
@startuml

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

@enduml
```



2.13 La commande Skinparam

Utilisez la commande skinparam pour changer la couleur et la mise en forme du texte du schéma.

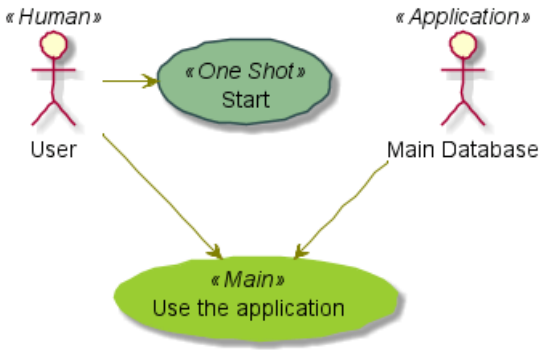
Vous pouvez utiliser cette commande :

- Dans la définition du diagramme, comme pour les autres commandes,
- Dans un fichier inclus,
- Dans un fichier de configuration, renseigné dans la ligne de commande ou la tâche ANT.

Vous pouvez aussi spécifier les polices et les couleurs pour les acteurs et cas d'utilisation avec des stéréotypes.

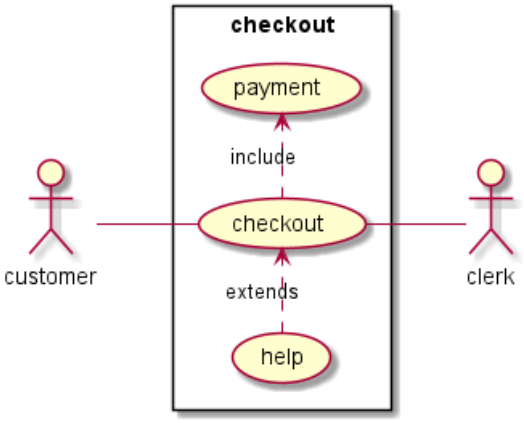
```
@startuml
skinparam handwritten true

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 Exemple complet

```
@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
```

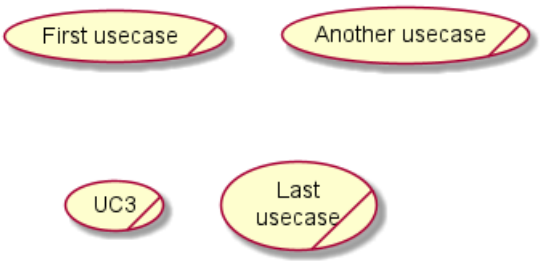


2.15 Business Use Case

You can add / to make Business Use Case.

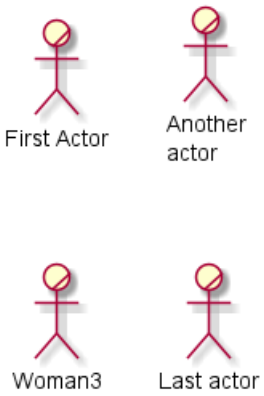
2.15.1 Business Usecase

```
@startuml

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

@enduml
```



2.15.2 Business Actor

```
@startuml

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

@enduml
```



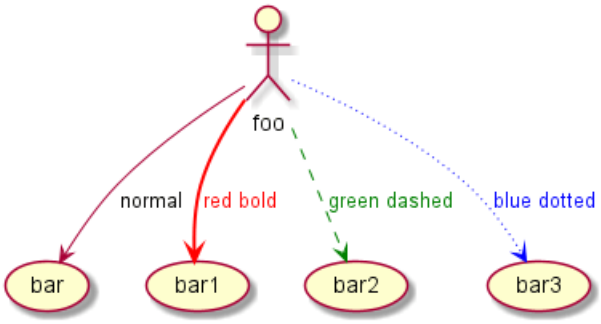
[Ref. QA-12179]

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
```



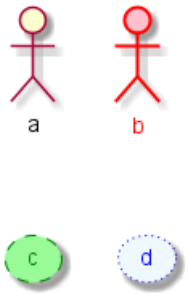
[Ref. QA-3770 and QA-3816] [See similar feature on deployment-diagram or class diagram]

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
```

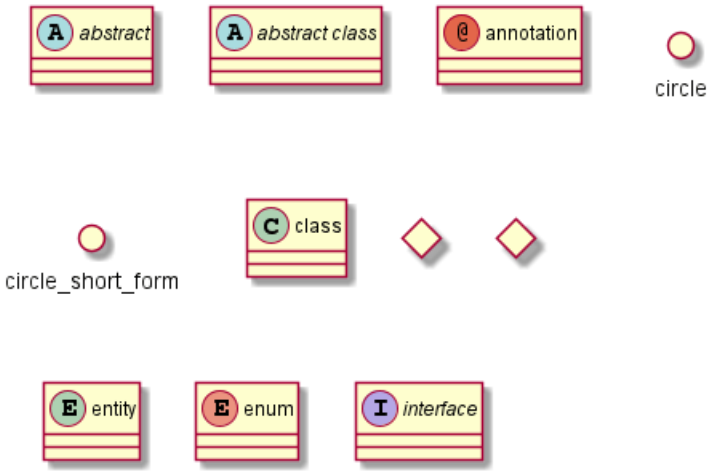


[Ref. QA-5340 and adapted from QA-6852]

3 Diagramme de classes

3.1 Déclaration des éléments

```
@startuml
abstract        abstract
abstract class  "abstract class"
annotation      annotation
circle          circle
()              circle_short_form
class           class
diamond         diamond
<>              diamond_short_form
entity          entity
enum            enum
interface       interface
@enduml
```



3.2 Relations entre classes

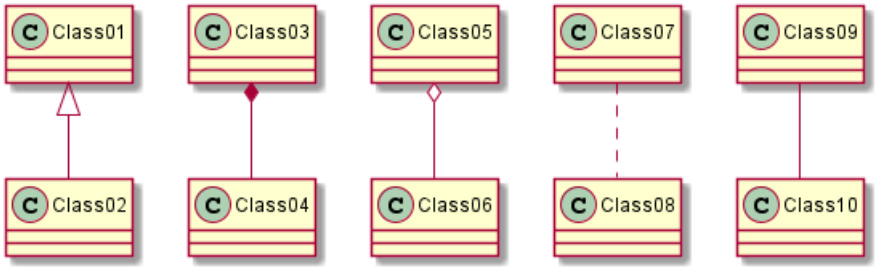
Les relations entre les classes sont définies en utilisant les symboles suivants :

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

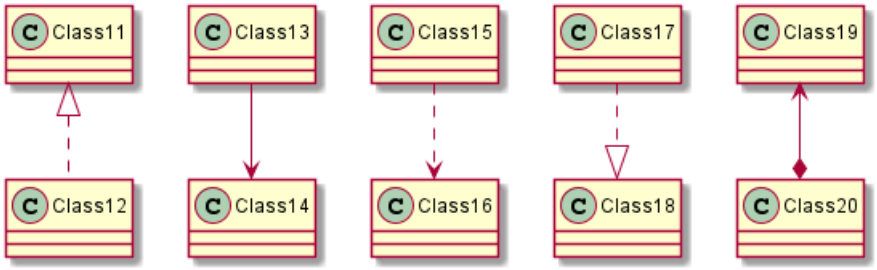
Il est possible de substituer `--` par `..` pour obtenir une ligne en pointillée.

Grâce à ces règles, il est possible de faire les diagrammes suivants :

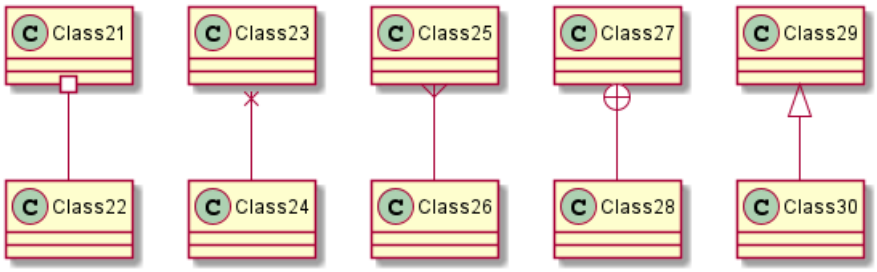
```
@startuml
Class01 <|-- Class02
Class03 *-- Class04
Class05 o-- Class06
Class07 .. Class08
Class09 -- Class10
@enduml
```




```
@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
```

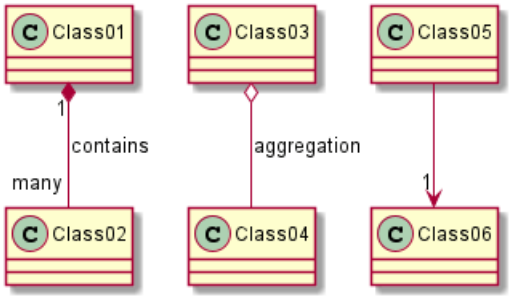


3.3 Libellés sur les relations

Il est possible de rajouter un libellé sur une relation, en utilisant les deux points `:`, suivi du texte du libellé.

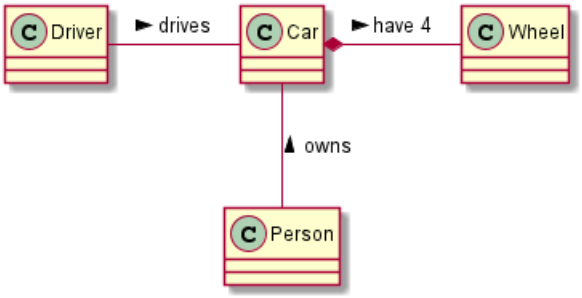
Pour les cardinalité, vous pouvez utiliser des guillemets `""` des deux cotés de la relation.

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

Class03 o-- Class04 : aggregation

Class05 --> "1" Class06
@enduml
```



Vous pouvez ajouter une flèche désignant quel objet agit sur l'autre en utilisant < ou > au début ou à la fin du libellé.

```
@startuml
class Car

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

@enduml
```

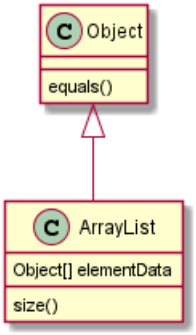


3.4 Définir les méthodes

Pour déclarer des méthodes ou des champs, vous pouvez utiliser le caractère : suivi de la méthode ou du champ.

Le système utilise la présence de parenthèses pour choisir entre méthodes et champs.

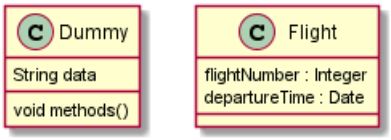
```
@startuml
Object <|-- ArrayList

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

@enduml
```



Il est possible de regrouper tous les champs et méthodes en utilisant des crochets `{}`.

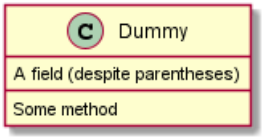
Notez que la syntaxe est très souple sur l'ordre des champs et des méthodes.

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

class Flight {
   flightNumber : Integer
   departureTime : Date
}
@enduml
```



You can use `{field}` and `{method}` modifiers to override default behaviour of the parser about fields and methods.

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

@enduml
```

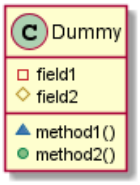


3.5 Définir les visibilités

Quand vous déclarez des champs ou des méthodes, vous pouvez utiliser certains caractères pour définir la visibilité des éléments :

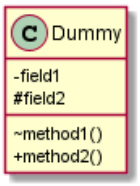
Character	Icon for field	Icon for method	Visibility
-	□	■	private
#	◇	◆	protected
~	△	▲	package private
+	○	●	public

```
@startuml

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

@enduml
```



Vous pouvez invalider cette fonctionnalité par la commande `skinparam classAttributeIconSize 0` :

```
@startuml
skinparam classAttributeIconSize 0
class Dummy {
 -field1
 #field2
 ~method1()
 +method2()
}

@enduml
```

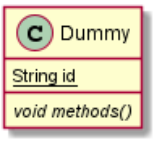


3.6 Abstrait et statique

Vous pouvez définir une méthode statique ou abstraite ou un champ utilisant `{static}` ou `{abstract}` modificateur.

Ce modificateur peut être utilisé au début ou à la fin de la ligne. Vous pouvez alors utiliser `{classifier}` plutôt que `{static}`.

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



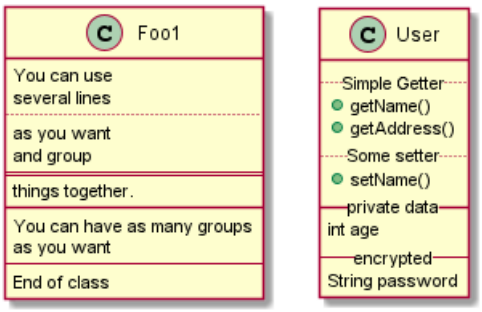
3.7 Corps de classe avancé

Par défaut, méthodes et champs sont automatiquement regroupés par PlantUML. Vous pouvez utiliser un séparateur pour définir votre propre manière d'ordonner les champs et les méthodes. Les séparateurs suivants sont possibles : `--` `..` `==` `__`.

Vous pouvez aussi utiliser les titres dans les séparateurs.

```
@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
```



3.8 Notes et stéréotypes

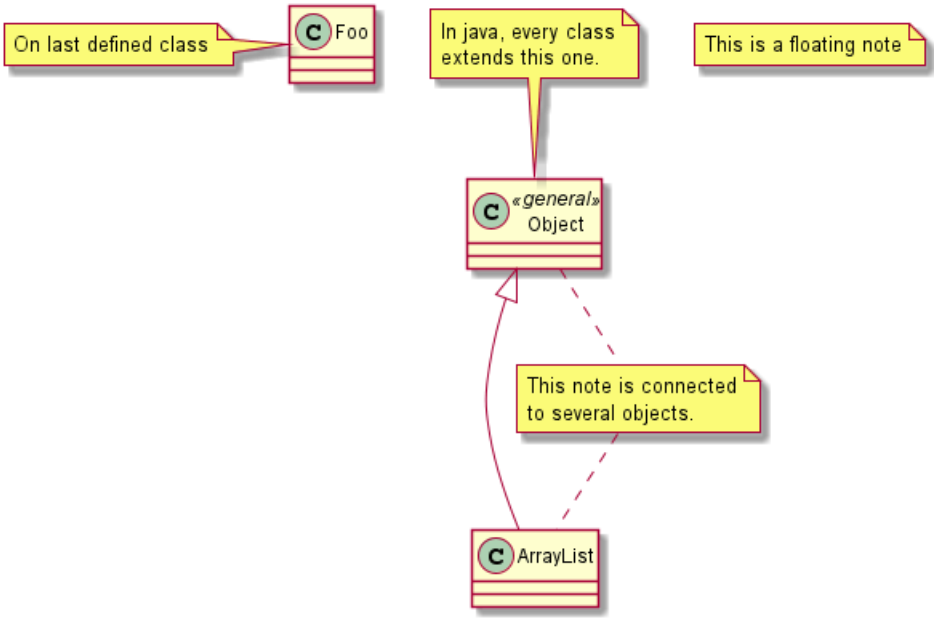
Stéréotypes sont définies avec le mot clé `class`, `<<` et `>>`.

Vous pouvez aussi définir une note en utilisant les mots clés `note left of` , `note right of` , `note top of` , `note bottom of`.

Vous pouvez aussi définir une note sur la dernière classe utilisant `note left`, `note right`, `note top`, `note bottom`.

Une note peut aussi être définie le mot clé `note`, puis être lié à un autre objet en utilisant le symbole `...`

```
@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
```



3.9 Encore des notes

Il est possible d'utiliser quelques tag HTML comme :

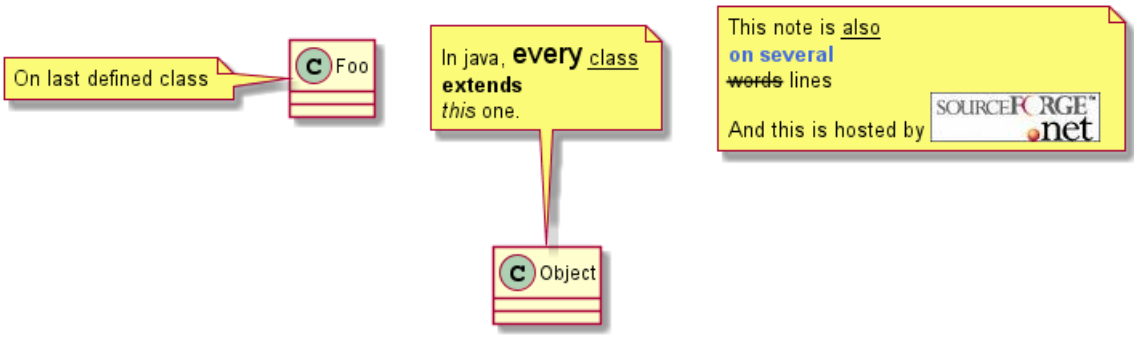
- `<b>`
- `<u>`
- `<i>`
- `<s>`, `<del>`, `<strike>`
- `<font color="#AAAAAA">` or `<font color="colorName">`
- `<color:#AAAAAA>` or `<color:colorName>`
- `<size:nn>` to change font size
- `<img src="file">` or `<img:file>`: the file must be accessible by the filesystem

Vous pouvez aussi définir des notes sur plusieurs lignes.

Vous pouvez également définir une note sur la dernière classe définie en utilisant `note left`, `note right`, `note top`, `note bottom`.

```
@startuml

class Foo
note left: On last defined class

note top of Object
  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
```

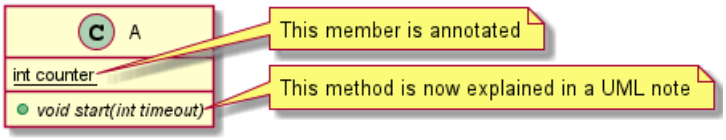


3.10 Note on field (field, attribute, member) or method

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

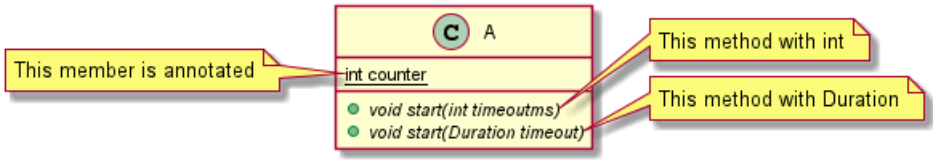
3.10.1 Note on field or method

```
@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
```



3.10.2 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
end note
@enduml
```



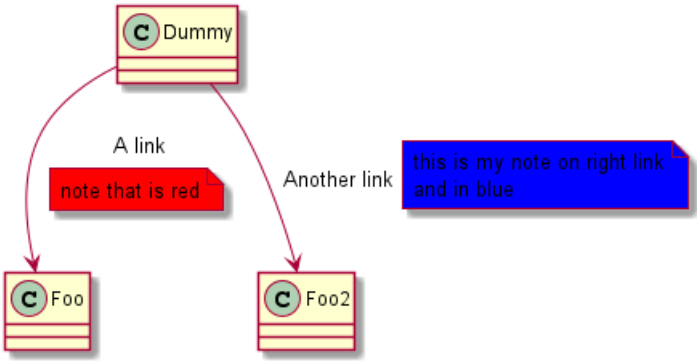
[Ref. QA-3474 and QA-5835]

3.11 Note sur les liens

Il est possible d'ajouter une note sur un lien, juste après la définition d'un lien, utiliser `note on link`.

Vous pouvez aussi utiliser `note left on link`, `note right on link`, `note top on link`, `note bottom on link` si vous voulez changer la position relative de la note avec l'étiquette.

```
@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
```



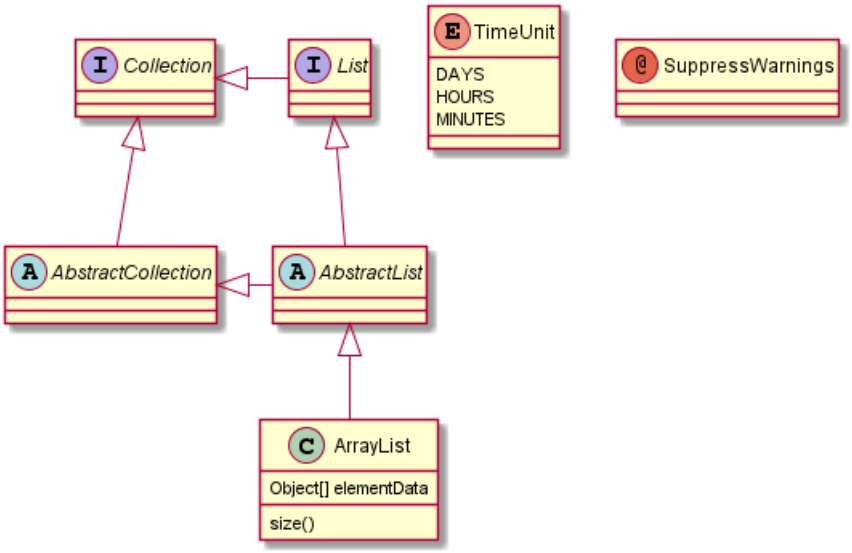
3.12 Classe abstraite et Interface

Vous pouvez déclarer un classe abstraite en utilisant `abstract` ou `abstract class`. La classe sera alors écrite en *italique*.

Vous pouvez aussi utiliser `interface`, `annotation` et `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
```



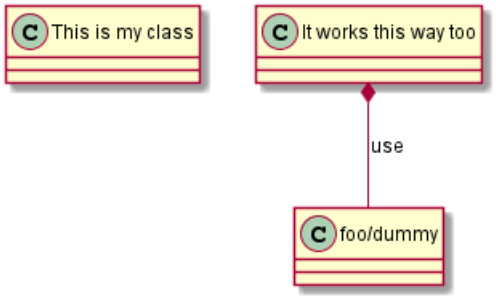
[Ref. 'Annotation with members' [Issue#458](https://github.com/plantuml/plantuml/issues/458)]

3.13 Caractères non alphabétiques

Si vous voulez utiliser autre chose que des lettres dans les classes (ou les enums...), vous pouvez:

- Utiliser le mot clé `as` dans la définition de la classe
- Metter des guillemets `""` autour du nom de la classe

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

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



3.14 Masquer les attributs et les méthodes

Vous pouvez paramétrer l'affichage des classes à l'aide de la commande `hide/show` .

La commande de base est: `hide empty members`. Cette commande va masquer la zone des champs ou des méthodes si celle-ci est vide.

A la place de `empty members`, vous pouvez utiliser:

- `empty fields` ou `empty attributes` pour des champs vides,
- `empty methods` pour des méthodes vides,
- `fields` or `attributes` qui masque les champs, même s'il y en a de définis,
- `methods` qui masque les méthodes, même s'il y en a de définies,
- `members` qui masque les méthodes ou les champs, même s'il y en a de définies,
- `circle` pour le caractère entouré en face du nom de la classe,
- `stereotype` pour le stéréotype.

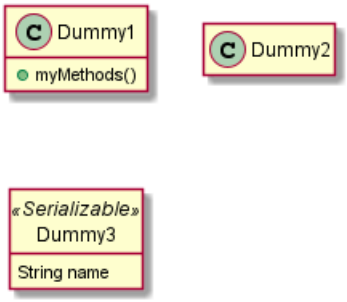
Vous pouvez aussi fournir, juste après le mot-clé `hide` ou `show` :

- `class` pour toutes les classes,
- `interface` pour toutes les interfaces,
- `enum` pour tous les enums,
- `<<foo1>>` pour les classes qui sont stéréotypée avec *foo1*,
- Un nom de classe existant

Vous pouvez utiliser plusieurs commandes `show/hide` pour définir des règles et des exceptions.

```
@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 Cacher des classes

Vous pouvez également utiliser la commande `show/hide` pour cacher une classe.

Cela peut être utile si vous définissez un fichier inclus de grande taille, et si vous voulez en cacher quelques classes après l'inclusion de ce fichier.

```
@startuml

class Foo1
class Foo2

Foo2 *-- Foo1

hide Foo2

@enduml
```

3.16 Remove classes

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

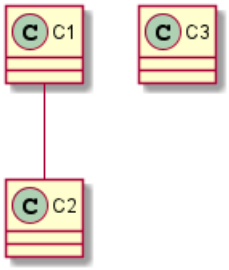
This may be useful if you define a large !included file, and if you want to remove some classes after file inclusion.

```
@startuml

class Foo1
class Foo2

Foo2 *-- Foo1

remove Foo2

@enduml
```

3.17 Hide 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:

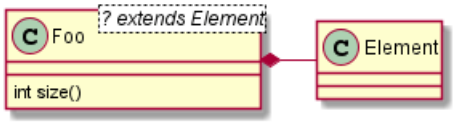
```
@startuml
class C1
class C2
class C3
C1 -- C2

remove @unlinked
@enduml
```

[Adapted from QA-11052]

3.18 Utilisation de la généricité

Vous pouvez aussi utiliser les signes inférieur `<` et supérieur `>` pour définir l'utilisation de la généricité dans une classe.

```
@startuml

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

@enduml
```



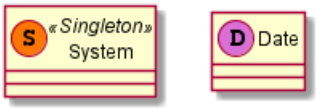
On peut désactiver ce comportement avec la commande `skinparam genericDisplay old`.

3.19 Caractère spécial

Normalement, un caractère (C, I, E ou A) est utilisé pour les classes, les interfaces ou les énum.

Vous pouvez aussi utiliser le caractère de votre choix, en définissant le stéréotype et en ajoutant une couleur, comme par exemple :

```
@startuml

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



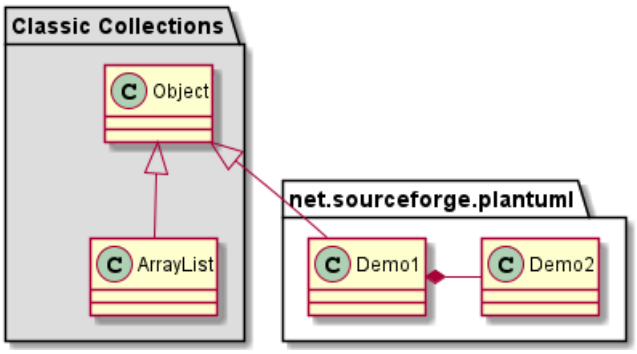
3.20 Packages

Vous pouvez définir un package en utilisant le mot-clé `package`, et optionnellement déclarer une couleur de fond pour votre package (en utilisant un code couleur HTML ou son nom).

Notez que les définitions de packages peuvent être imbriquées.

```
@startuml

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

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

@enduml
```

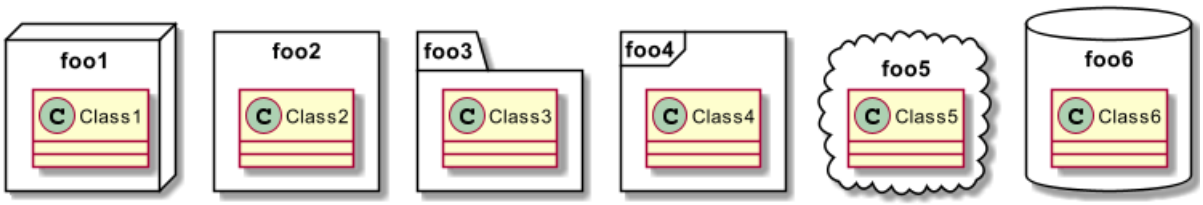


3.21 Modèle de paquet

Il y a différents styles de paquets disponibles.

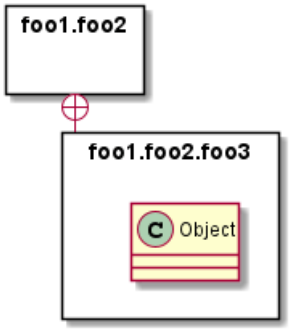
Vous pouvez les spécifier chacun par un réglage par défaut avec la commande : `skinparam packageStyle`, ou par l'utilisation d'un stéréotype sur le paquet:

```
@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
```



Vous pouvez aussi définir les liens entre les paquets, comme dans l'exemple suivant :

```
@startuml

skinparam packageStyle rectangle

package foo1.foo2 {
}

package foo1.foo2.foo3 {
  class Object
}

foo1.foo2 +-- foo1.foo2.foo3

@enduml
```



3.22 Les espaces de nommage

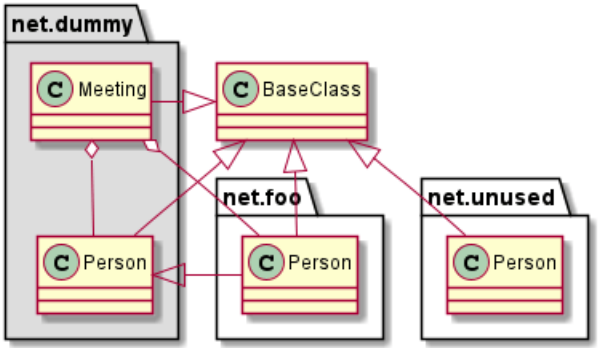
Avec les packages, le nom de la classe est l'identifiant unique de la classe. Cela signifie qu'on ne peux pas avoir deux classes avec le même nom dans deux packages différents. Pour ce faire, vous devez utiliser des espace de nommage (*namespace*) à la place des packages.

Vous pouvez faire référence à des classes d'autres espace de nommage en les nommant complétement. Les classes de l'espace de nommage par défaut (racine) sont nommées en commençant par un point.

Il n'est pas obligatoire de créer les espaces de nom. Un classe avec son nom complet sera automatiquement ajoutée au bon espace de nommage.

```
@startuml

class BaseClass

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

    .BaseClass <|- Meeting
}

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

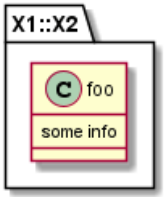
```
  .BaseClass <|-- Person

  net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person

@enduml
```



3.23 Creation automatique d'espace de nommage

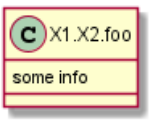
Vous pouvez définir une autre séparateur (autre que le point) en utilisant la commande : `set namespaceSeparator ???`.

```
@startuml

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

@enduml
```



Vous pouvez désactiver la création automatique de package en utilisant la commande `set namespaceSeparator none`.

```
@startuml

set namespaceSeparator none
class X1.X2.foo {
  some info
}

@enduml
```

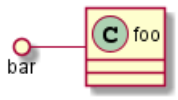


3.24 Interface boucle

Vous pouvez aussi rajouter des interfaces sur les classes avec la syntaxe suivante:

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

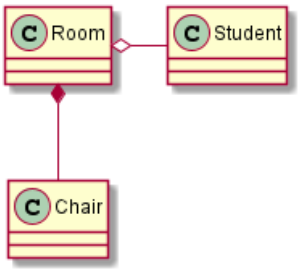
```
@startuml
class foo
bar ()- foo
@enduml
```



3.25 Changer la direction

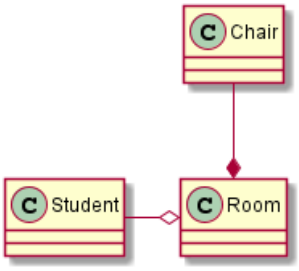
Par défaut, les liens entre les classe ont deux tirets `--` et sont orientés verticalement. Il est possible d'utiliser une ligne horizontal en mettant un simple tiret (Ou un point) comme ceci:

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



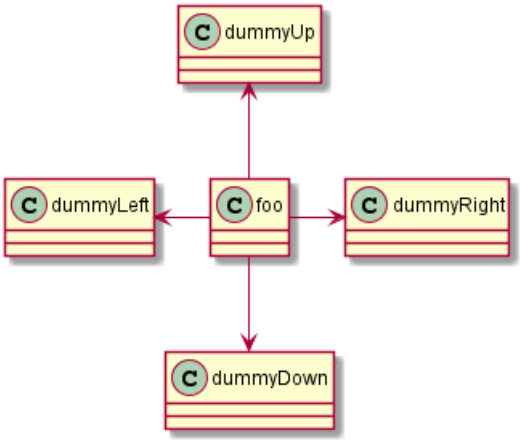
Vous pouvez aussi changer le sens en renversant le lien :

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



Il est aussi possible de changer la direction d'une flèche en ajoutant les mots clés `left`, `right`, `up` ou `down` à l'intérieur de la flèche:

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

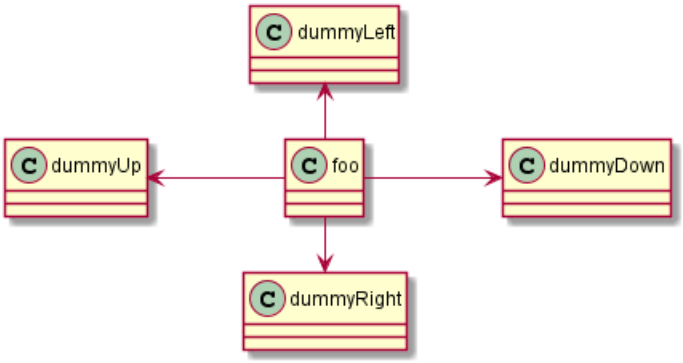


Il est possible de raccourcir la flèche en n'utilisant que la première lettre de la direction (par exemple, `-d-` au lieu de `-down-`) ou les deux premières lettres (`-do-`)

Attention à ne pas abuser de cette fonctionnalité : *GraphViz* donne généralement de bons résultats sans trop de raffistolages.

Et avec le paramètre `left to right direction`:

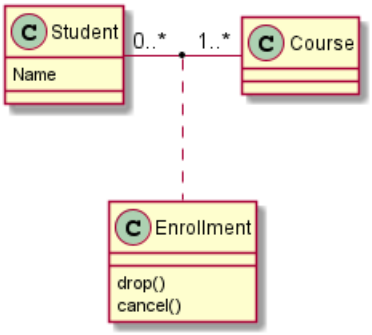
```
@startuml
left to right direction
foo -left-> dummyLeft
foo -right-> dummyRight
foo -up-> dummyUp
foo -down-> dummyDown
@enduml
```



3.26 Classes d'association

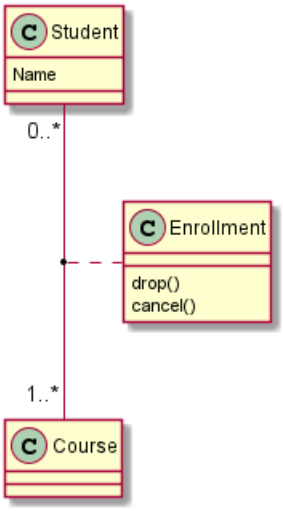
Vous pouvez définir une *classe d'association* après qu'une relation ait été définie entre deux classes, comme dans l'exemple suivant:

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

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



Vous pouvez la définir dans une autre direction :

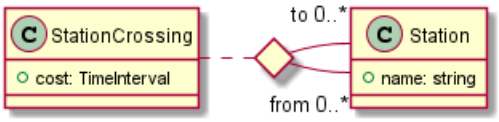
```
@startuml
class Student {
  Name
}
Student "0..*" -- "1..*" Course
(Student, Course) . Enrollment

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



3.27 Association sur la même classe

```
@startuml
class Station {
    +name: string
}

class StationCrossing {
    +cost: TimeInterval
}

<> diamond

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

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



[Réf. Incubation : Associations]

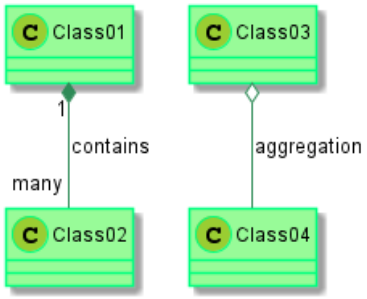
3.28 Personnalisation

La commande skinparam permet de changer la couleur et les polices de caractères.

Vous pouvez utiliser cette commande :

- Dans le diagramme, comme toutes les autre commandes,
- Dans un fichier inclus,
- Dans un fichier de configuration précisé par la ligne de commande ou la tâche ANT.

```
@startuml

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

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

Class03 o-- Class04 : aggregation

@enduml
```

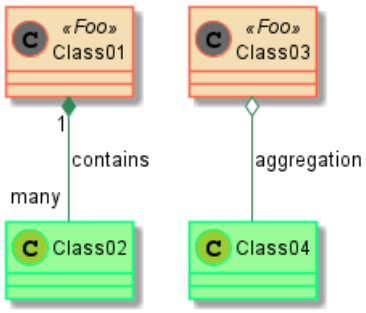


3.29 Stéréotypes Personnalisés

Vous pouvez définir des couleurs et des fontes de caractères spécifiques pour les classes stéréotypées.

```
@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
```



3.30 Dégradé de couleur

Il est possible de déclarer individuellement une couleur pour des classes ou une note en utilisant la notation #.

Vous pouvez utiliser un nom de couleur standard ou un code RGB.

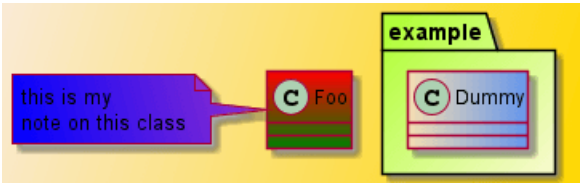
Vous pouvez aussi utiliser un dégradé de couleur en fond, avec la syntaxe suivante : deux noms de couleurs séparés par :

- |,
- /,
- \,
- ou -

en fonction de la direction du dégradé

Par exemple, vous pouvez avoir :

```
@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
```



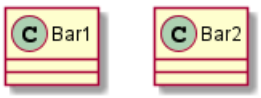
3.31 Aide pour la mise en page

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
```



3.32 Découper les grands diagrammes

Parfois, vous obtiendrez des images de taille importante.

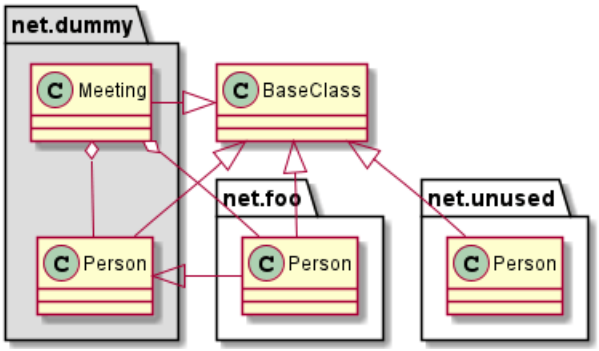
Vous pouvez utiliser la commande `page (hpages)x(vpages)` pour découper l'image en plusieurs fichiers:

`hpages` est le nombre de pages horizontales et `vpages` indique le nombre de pages verticales.

Vous pouvez aussi utiliser des paramètres spécifiques pour rajouter des bords sur les pages découpées (voir l'exemple).

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

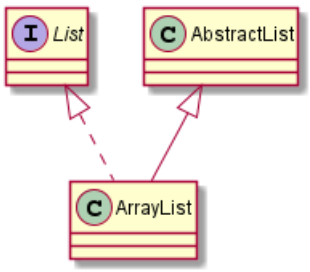
```
class BaseClass

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
```



3.33 Extension et implementation [extends, implements]

Il est aussi possible d'utiliser directement les mots clés `extends` and `implements`.

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



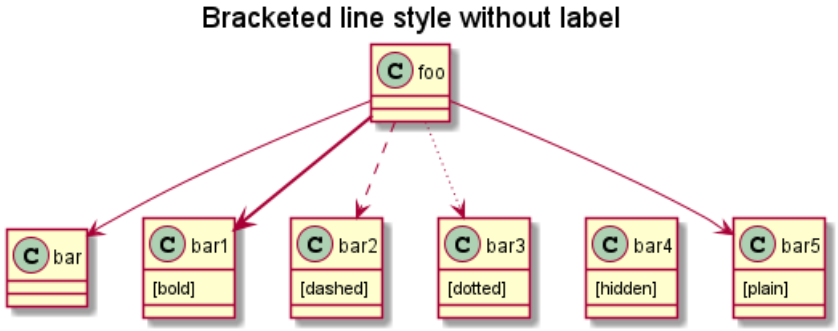
3.34 Bracketed relations (linking or arrow) style

3.34.1 Line style

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

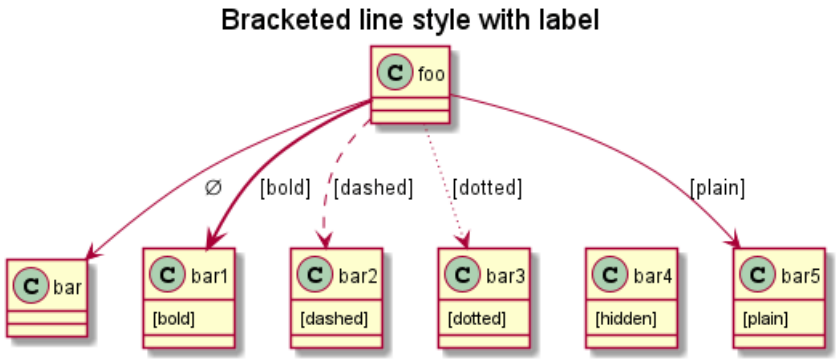
- without label

```
@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
```



- with label

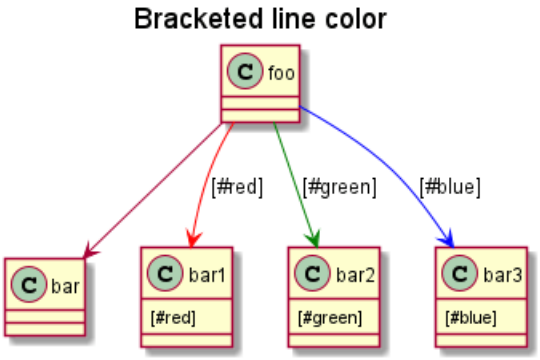
```
@startuml
title Bracketed line style with label
class foo
class bar
bar1 : [bold]
bar2 : [dashed]
bar3 : [dotted]
bar4 : [hidden]
bar5 : [plain]

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]

3.34.2 Line color

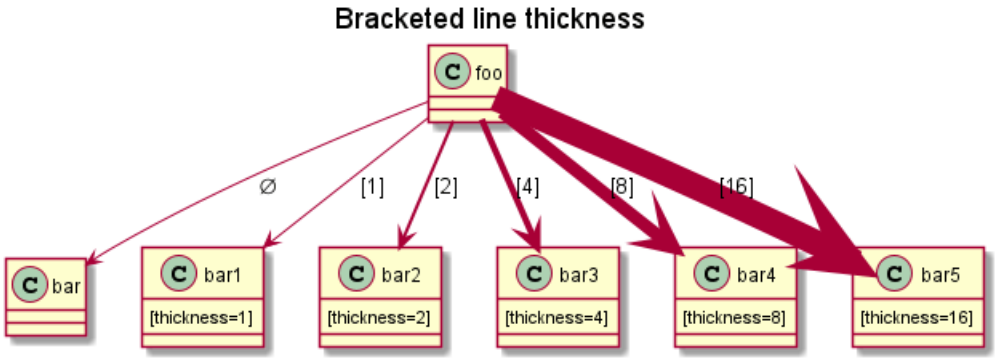
```
@startuml
title Bracketed line color
class foo
class bar
bar1 : [#red]
bar2 : [#green]
bar3 : [#blue]

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



3.34.3 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]
```

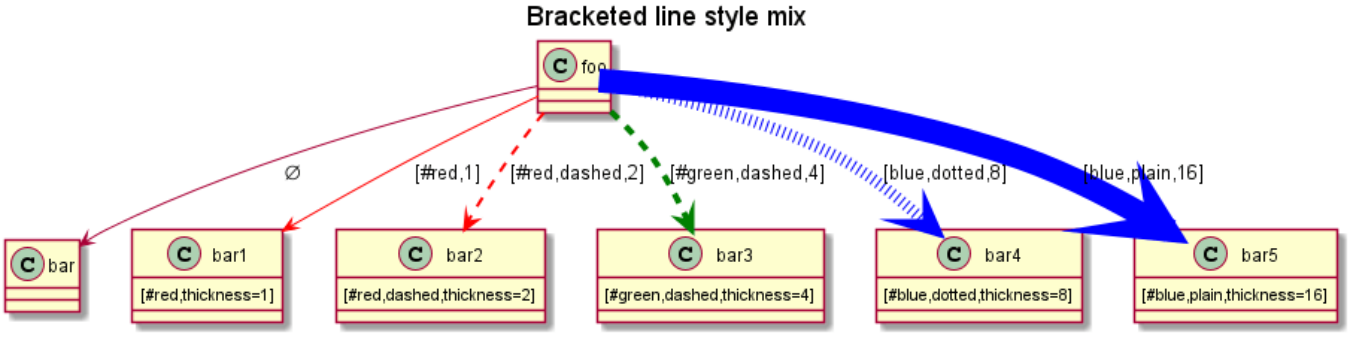
```
foo -[thickness=8]-> bar4   : [8]
foo -[thickness=16]-> bar5  : [16]

@enduml
```



[Ref. QA-4949]

3.34.4 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
```



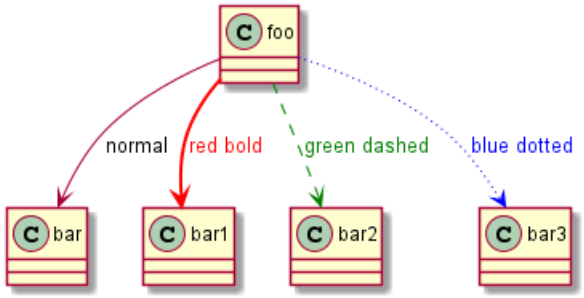
3.35 Change relation (linking or arrow) color and style (inline style)

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
```



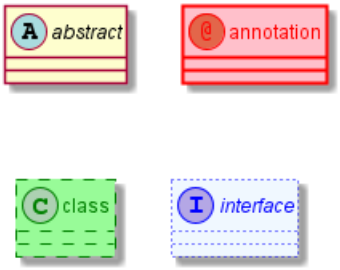
[See similar feature on deployment]

3.36 Change class color and style (inline style)

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

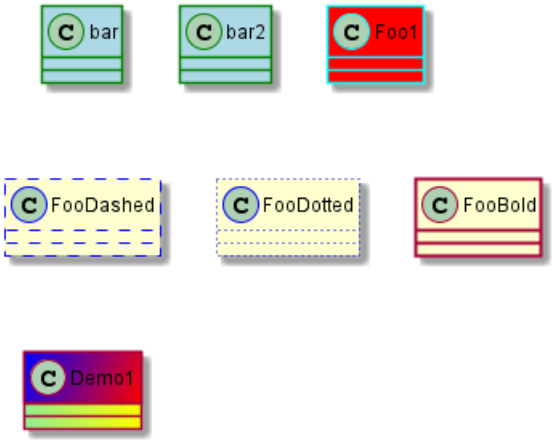
- `#[color|back:color];header:color;line:color;line.[bold|dashed|dotted];text:color`

```
@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:

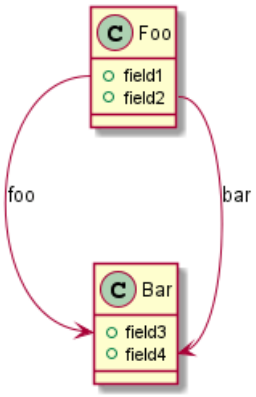
```
@startuml
class bar #line:green;back:lightblue
class bar2 #lightblue;line:green

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]

3.37 Arrows from/to class members

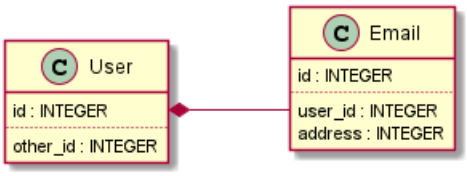
```
@startuml
class Foo {
+ field1
+ field2
}

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]

4 Diagrammes d'objets

4.1 Définition des objets

Les instances d'objets sont défnies avec le mot clé `object`.

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

4.2 Relations entre les objets

Les relations entre objets sont définies à l'aide des symboles suivants :

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

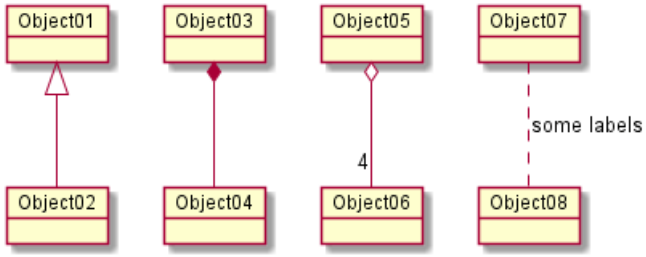
Il est possible de remplacer `--` par `..` pour avoir des pointillés.

Grâce à ces règles, on peut avoir les dessins suivants:

Il est possible d'ajouter une étiquette sur la relation, en utilisant `:` suivi par le texte de l'étiquette.

Pour les cardinalités, vous pouvez utiliser les doubles quotes `""` sur chaque côté de la relation.

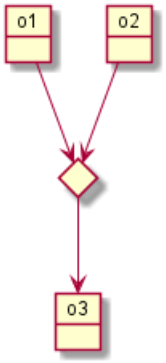
```
@startuml
object Object01
object Object02
object Object03
object Object04
object Object05
object Object06
object Object07
object Object08

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



4.3 Associations objects

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

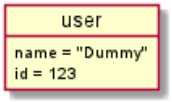
```
object o3

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



4.4 Ajout de champs

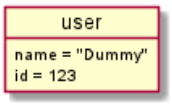
Pour déclarer un champ, vous pouvez utiliser le symbole : suivi par le nom du champs.

```
@startuml

object user

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

@enduml
```



It is also possible to ground between brackets {} all fields.

```
@startuml

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

@enduml
```



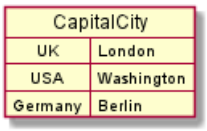
4.5 Caractéristiques communes avec les diagrammes de classes

- Visibilité
- Ajout de notes
- Utilisation de packages
- Personnalisation de l'affichage

4.6 Map table or associative array

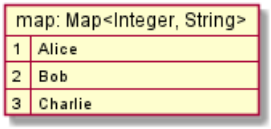
You can define a map table or associative array, with `map` keyword and `=>` separator.

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



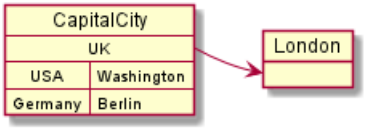

```
@startuml
map "Map **Contry => CapitalCity**" as CC {
 UK => London
 USA => Washington
 Germany => Berlin
}
@enduml
```


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



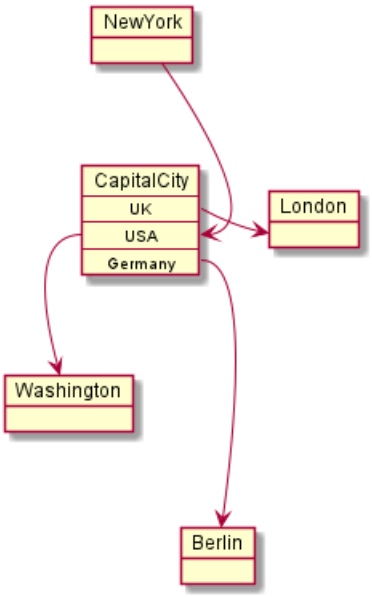
And add link with object.

```
@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]

5 Diagrammes d'activité (ancienne syntaxe)

Il s'agit de l'ancienne syntaxe du diagramme d'activités, pour voir la nouvelle version actuelle, voir: Diagrammes d'activité (nouvelle syntaxe).

5.1 Exemple de base

Vous devez utiliser `(*)` pour le début et la fin du diagramme d'activité.

Dans certaines occasions, vous pouriez vouloir utiliser `(*top)` pour forcer le début à être en haut du diagramme.

Utiliser `-->` pour les flèches.

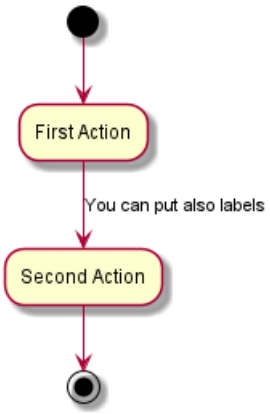
```
@startuml

(*) --> "First Action"
"First Action" --> (*)

@enduml
```

5.2 Texte sur les flèches

Par défaut, une flèche commence à partir de la dernière activité définie.

Vous pouvez rajouter un libellé sur une flèche en mettant des crochets `[` et `]` juste après la définition de la flèche.

```
@startuml

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

@enduml
```

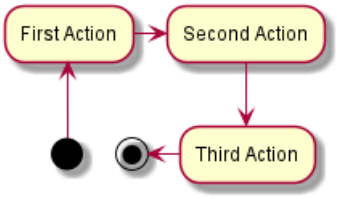


5.3 Changer la direction des flèches

Vous pouvez utiliser `->` pour les flèches horizontales. Il est aussi possible de forcer la direction d'une flèche en utilisant la syntaxe suivante :

- `-down->` (default arrow)
- `-right->` or `->`
- `-left->`
- `-up->`

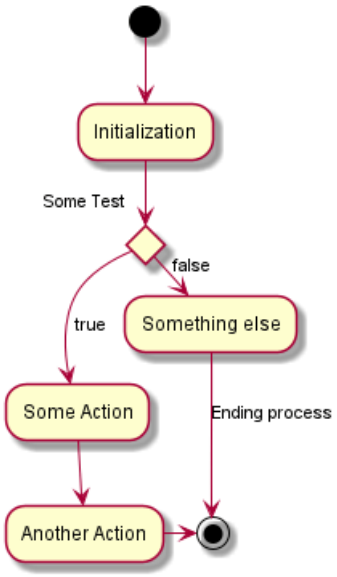
```
@startuml

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

@enduml
```



5.4 Branches

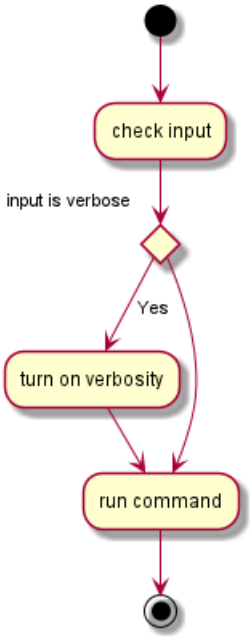
Vous pouvez utiliser le mot clé `if/then/else` pour définir une branche.

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

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

@enduml
```



Malheureusement, vous devez parfois avoir à répéter la même activité dans le diagramme de texte.

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

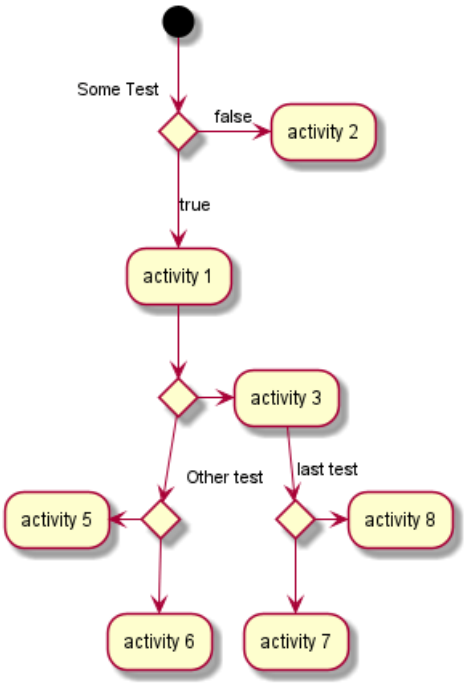


5.5 Encore des branches

Par défaut, une branche commence à la dernière activité définie, mais il est possible de passer outre et de définir un lien avec le mot clé `if`.

Il est aussi possible d'imbriquer les branches.

```
@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
```

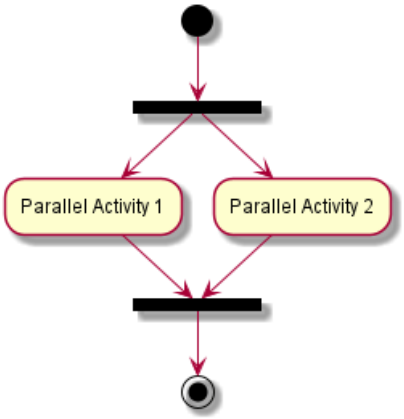


5.6 Synchronisation

Vous pouvez utiliser la syntaxe `=== code ===` pour afficher des barres de synchronisation.

```
@startuml

(*) --> ===B1===
```

```
--> "Parallel Activity 1"
--> ===B2===

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

--> (*)

@enduml
```

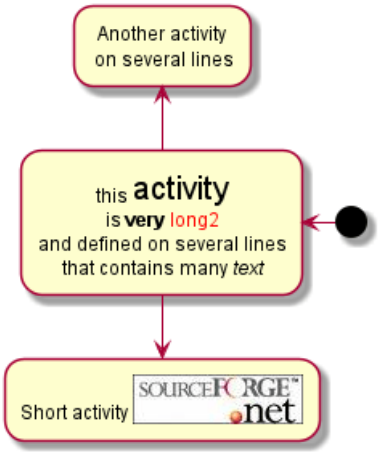


5.7 Description détaillée

Lorsque vous déclarez des activités, vous pouvez positionner sur plusieurs lignes le texte de description Vous pouvez également ajouter dans la description. Il est également possible d'utiliser quelques tags HTML tels que :

Vous pouvez aussi donner un court code à l'activité avec le mot clé `as`. Ce code peut être utilisé plus tard dans le diagramme de description.

```
@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
```



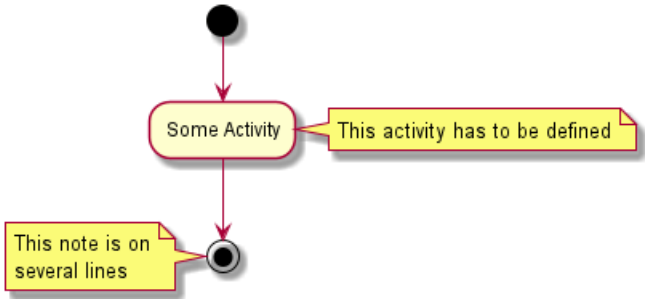
5.8 Notes

Vous pouvez rajouter des notes sur une activités en utilisant les commandes: `note left`, `note right`, `note top` ou `note bottom`, juste après la définition de l'activité concernée.

Si vous voulez mettre une note sur le démarrage du diagramme, définissez la note au tout début du diagramme.

Vous pouvez aussi avoir une note sur plusieurs lignes, en utilisant les mots clés `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 Partition

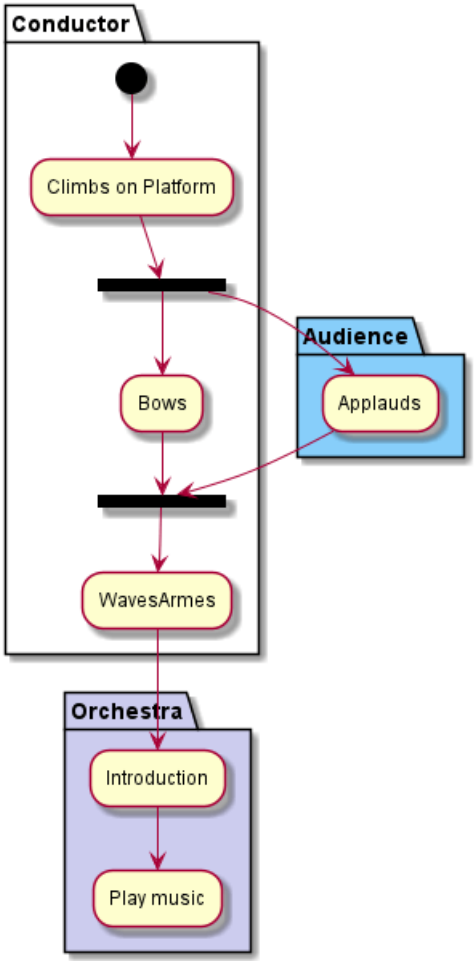
Vous pouvez définir une partition en utilisant le mot clé `partition`, et optionnellement déclarer un fond de couleur pour votre partition (En utilisant un code couleur html ou un nom)

Quand vous déclarez les activités, ils sont automatiquement mis dans la dernière partition utilisée.

Vous pouvez fermer la partition de définition en utilisant les crochets fermants `}`.

```
@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 Paramètre de thème

Vous pouvez utiliser la commande `skinparam` pour changer la couleur et la police d'écriture pour dessiner.

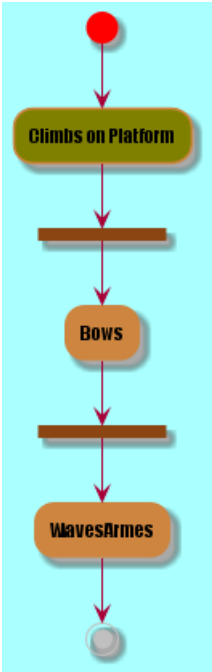
Vous pouvez utiliser cette commande :

- Dans le diagramme de définition, comme n'importe quelle autre commande,
- Dans un fichier inclus,
- Dans un fichier de configuration, à l'aide de la la ligne de commande ou la tâche ANT.

Vous pouvez spécifier une couleur et une police d'écriture dans les stéréotypes d'activités.

```
@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 Octogone

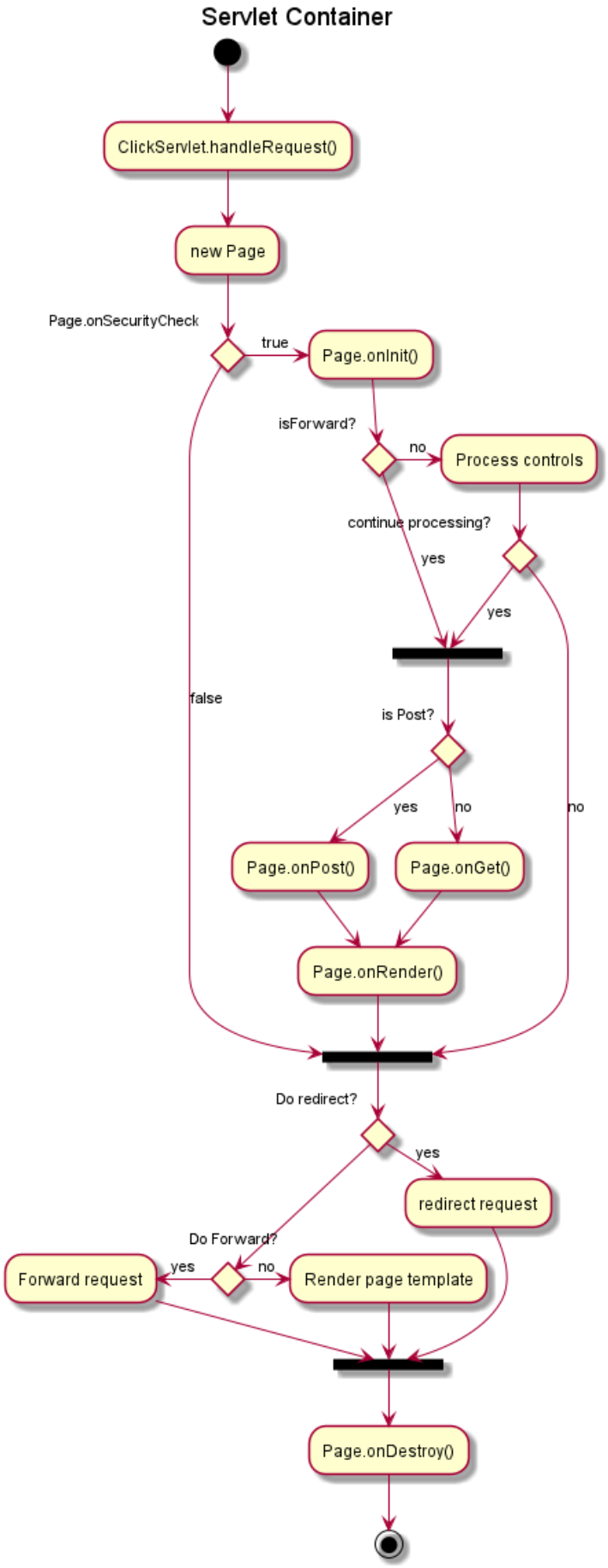
Vous pouvez changer la forme des activités en octogone en utilisant la commande `skinparam activityShape octagon`.

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

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

@enduml
```

5.12 Exemple complet

```
@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
```


Servlet Container
ClickServlet.handleRequest()
new Page
Page.onSecurityCheck
true
Page.onInit()
isForward?
no
Process controls
continue processing?
yes
yes
false
is Post?
yes
no
no
Page.onPost()
Page.onGet()
Page.onRender()
Do redirect?
yes
redirect request
Do Forward?
yes
no
Forward request
Render page template
Page.onDestroy()

6 Diagrammes d'activité (nouvelle syntaxe)

L'ancienne syntaxe pour les diagrammes d'activité possédait plusieurs limitations et inconvénients (par exemple, la difficulté à maintenir un diagramme lors de modifications).

Une nouvelle syntaxe est maintenant proposée aux utilisateurs. Un autre avantage de cette nouvelle implémentation est qu'il n'y a pas besoin d'avoir Graphviz d'installé (comme pour les diagrammes de séquences).

La nouvelle syntaxe remplace l'ancienne. Cependant, pour des raisons de compatibilité, l'ancienne syntaxe reste reconnue, pour assurer *la compatibilité ascendante*.

Les utilisateurs sont simplement encouragés à migrer vers la nouvelle syntaxe.

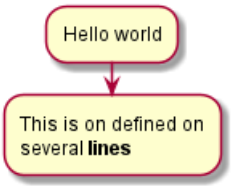
6.1 Activité simple

Les étiquettes d'activités commencent avec `:` et finissent avec `;`.

Le formatage de texte peut être fait en utilisant la syntaxe créole wiki.

Les activités sont implicitement liées à leur ordre de définition.

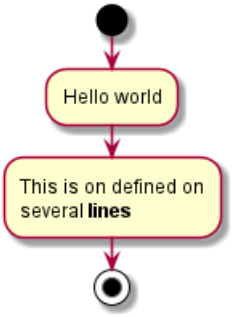
```
@startuml
:Hello world;
:This is on defined on
several **lines**;
@enduml
```



6.2 Départ/Arrêt [start, stop, end]

Vous pouvez utiliser les mots clés `start` et `stop` pour indiquer le début et la fin du diagramme.

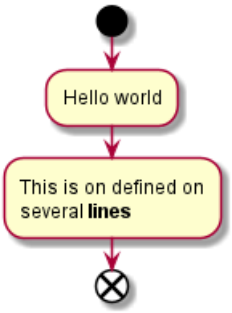
```
@startuml
start
:Hello world;
:This is on defined on
several **lines**;
stop
@enduml
```



Vous pouvez aussi utiliser le mot clé `end`.

```
@startuml
start
:Hello world;
```

```
:This is on defined on
several **lines**;
end
@enduml
```



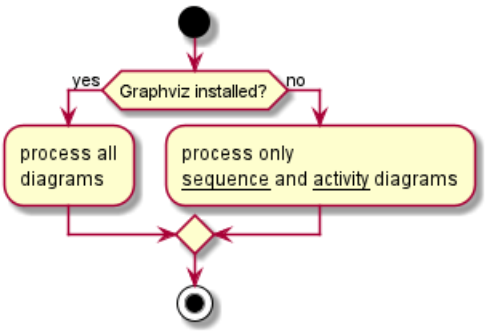
6.3 Conditionnel [if, then, else]

Vous pouvez utiliser les mots clés `if`, `then` et `else` pour mettre des tests dans votre diagramme. Les étiquettes peuvent être fournies entre parenthèses.

Les trois syntaxes possibles sont:

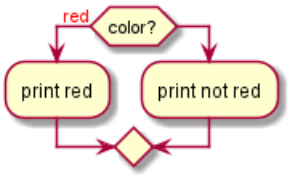
- `if (...) then (...)`

```
@startuml

start

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

stop

@enduml
```



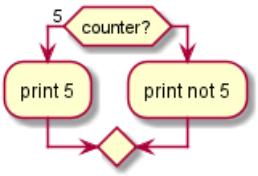
- `if (...) is (...) then`

```
@startuml
if (color?) is (<color:red>red) then
:print red;
else
:print not red;
@enduml
```



- `if (...) equals (...) then`

```
@startuml
if (counter?) equals (5) then
:print 5;
else
:print not 5;
@enduml
```

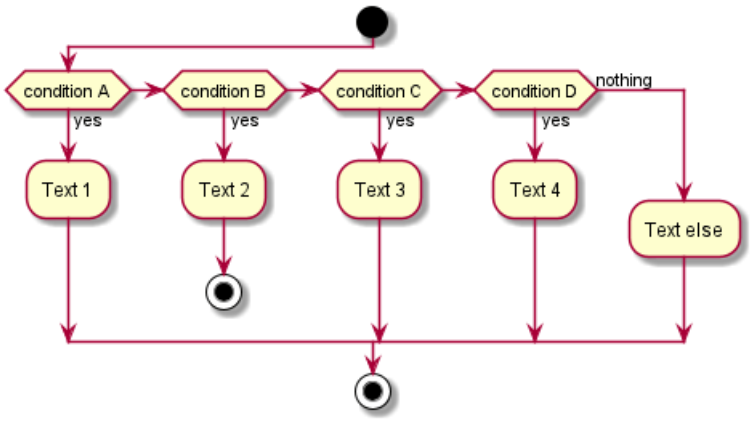


[Ref. QA-301]

6.3.1 Plusieurs conditions (en mode horizontal)

Vous pouvez utiliser le mot clé `elseif` pour avoir plusieurs tests, par défaut le mode est horizontal :

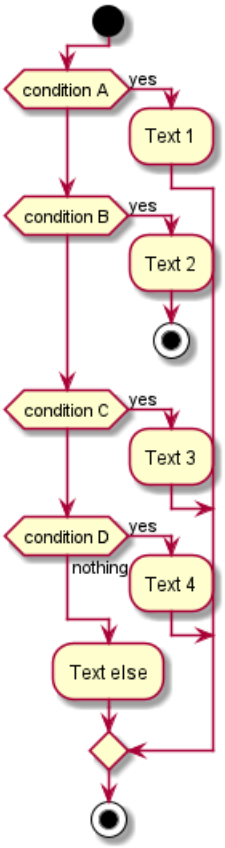
```
@startuml
start
if (condition A) then (yes)
  :Text 1;
elseif (condition B) then (yes)
  :Text 2;
  stop
elseif (condition C) then (yes)
  :Text 3;
elseif (condition D) then (yes)
  :Text 4;
else (nothing)
  :Text else;
endif
stop
@enduml
```



6.3.2 Plusieurs conditions (en mode vertical)

Vous pouvez utiliser la commande `!pragma useVerticalIf on` pour avoir les conditions en mode vertical :

```
@startuml
!pragma useVerticalIf on
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
```



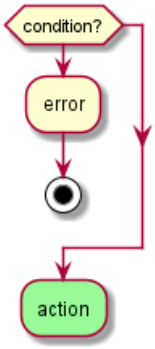
[Réf. QA-3931]

6.4 Arrêt après une action au sein d'une condition [kill, detach]

Vous pouvez arrêter le processus après une action.

```
@startuml
if (condition?) then
  :error;
```

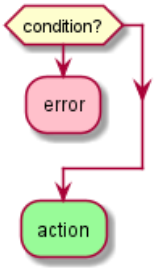
```
  stop
endif
#palegreen:action;
@enduml
```



Vous pouvez également utiliser les mots clé `kill` ou `detach` pour mettre fin au processus directement dans une action.

- `kill`

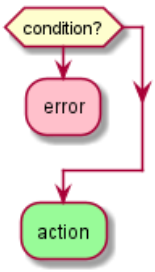
```
@startuml
if (condition?) then
  #pink:error;
  kill
endif
#palegreen:action;
@enduml
```



[Ref. QA-265]

- `detach`

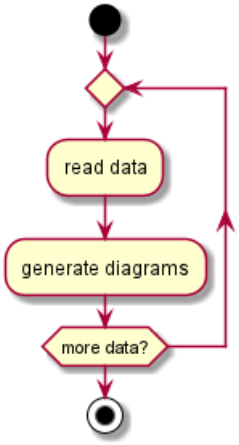
```
@startuml
if (condition?) then
  #pink:error;
  detach
endif
#palegreen:action;
@enduml
```



6.5 Boucle de répétition [repeat, repeatwhile, backward]

Vous pouvez utiliser les mots clés `repeat` et `repeatwhile` pour créer une boucle.

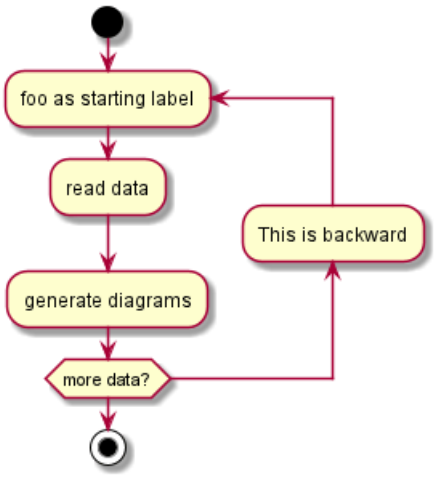
```
@startuml

start

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

stop

@enduml
```



Il est également possible :

- d'utiliser une vrai action comme cible de répétition, après le premier mot clé `repeat`,
- d'insérer une action dans le chemin de retour à l'aide du mot clé `backward`.

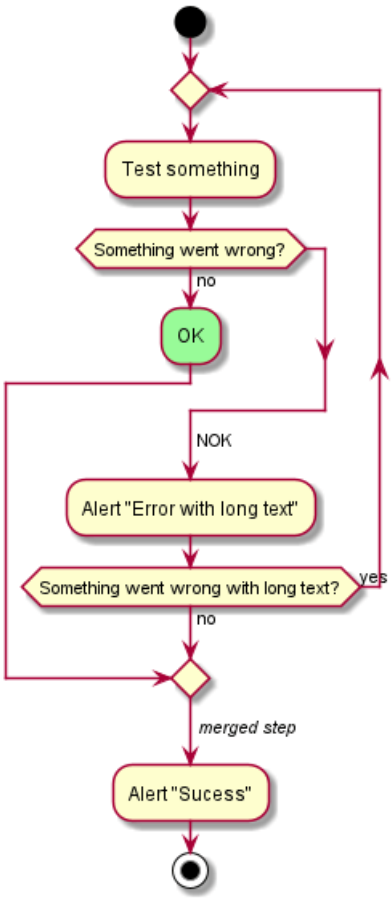
```
@startuml

start

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

stop

@enduml
```



6.6 Interruption d'une boucle [break]

Vous pouvez interrompre une boucle après une action en utilisant le mot clé `break` :

```
@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 "Sucess";
stop
@enduml
```

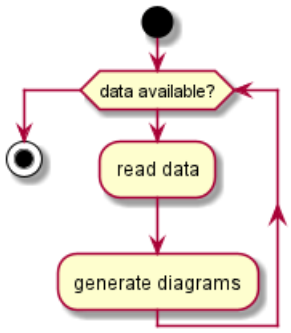


[Ref. QA-6105]

6.7 Boucle « tant que » [while]

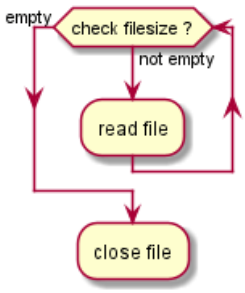
Vous pouvez utiliser les mots clés `while` et `end while` pour définir une boucle.

```
@startuml

start

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

stop

@enduml
```



Il est possible de mettre un libellé après le mot clé `endwhile` ou bien avec le mot clé `is`.

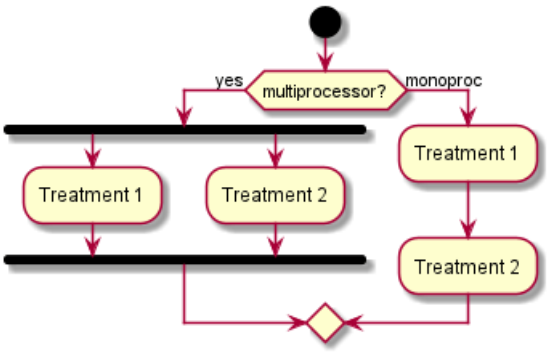
```
@startuml
while (check filesize ?) is (not empty)
  :read file;
endwhile (empty)
:close file;
@enduml
```



6.8 Processus parallèle [fork, fork again, end fork]

Vous pouvez utiliser les mots clés `fork`, `fork again` et `end fork` pour indiquer un processus parallèle.

```
@startuml

start

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

@enduml
```

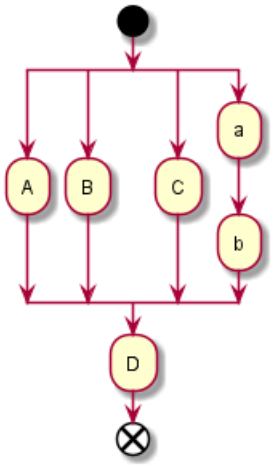


6.9 Split processing

6.9.1 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
```

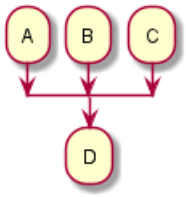


6.9.2 Input split (multi-start)

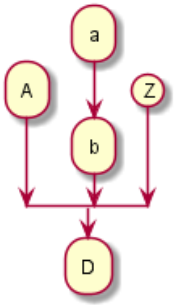
You can use `hidden` arrows to make an input split (multi-start):

```
@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
```



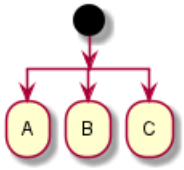
[Ref. QA-8662]

6.9.3 Output split (multi-end)

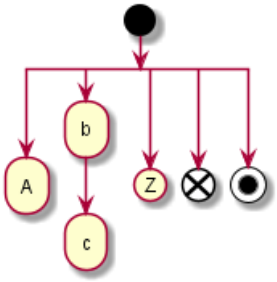
You can use `kill` or `detach` to make an output split (multi-end):

```
@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
```



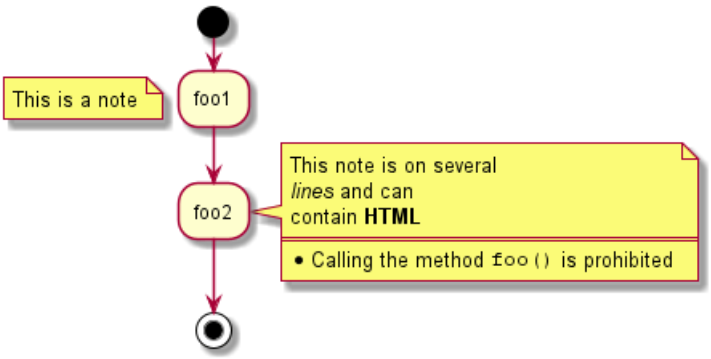
6.10 Notes

Le formattage de texte peut être fait en utilisant la syntaxe créole wiki.

Une note peut aussi être détachée, à l'aide du mot-clé `floating`.

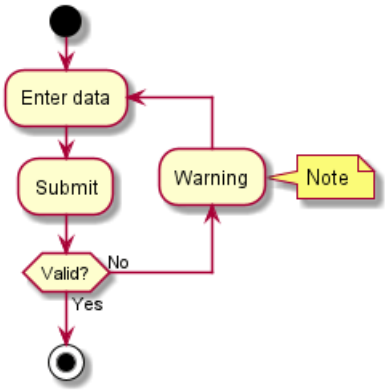
```
@startuml

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
```



Vous pouvez ajouter une note sur un chemin de retour.

```
@startuml
start
repeat :Enter data;
:Submit;
backward :Warning;
note right: Note
repeat while (Valid?) is (No) not (Yes)
stop
@enduml
```

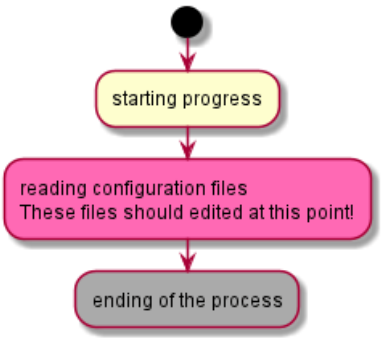


[Ref. QA-11788]

6.11 Couleurs

Vous pouvez spécifier une couleur pour certaines activités.

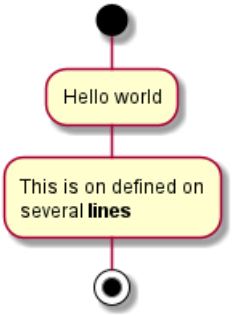
```
@startuml

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

@enduml
```



6.12 Lignes sans pointe de flèches

Vous pouvez utiliser `skinparam ArrowHeadColor none` pour connecter des activités en utilisant uniquement des lignes, sans flèches (sans pointe sur les flèches).

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



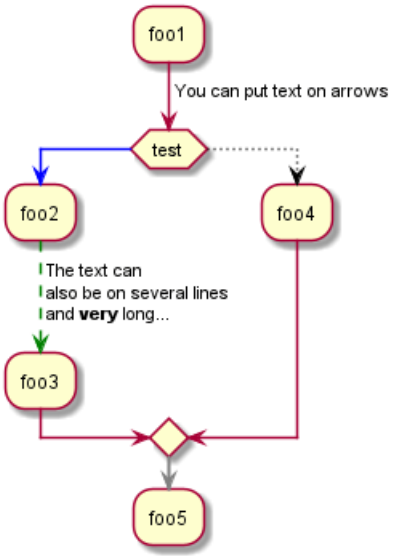

```
@startuml
skinparam ArrowHeadColor none
start
repeat :Enter data;
:Submit;
backward :Warning;
repeat while (Valid?) is (No) not (Yes)
stop
@enduml
```

6.13 Flèches

En utilisant la notation ->, vous pouvez ajouter du texte à une flèche, et changer sa couleur.

Il est aussi possible d'avoir des flèches en pointillé, en gras, avec des tirets ou bien complètement cachées.

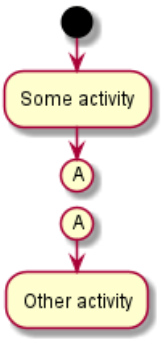
```
@startuml
:foo1;
-> You can put text on arrows;
if (test) then
  -[#blue]->
  :foo2;
  -[#green,dashed]-> The text can
  also be on several lines
  and **very** long...;
  :foo3;
else
  -[#black,dotted]->
  :foo4;
endif
-[#gray,bold]->
:foo5;
@enduml
```



6.14 Connecteurs

Il est possible d'utiliser des parenthèses pour dessiner des connecteurs.

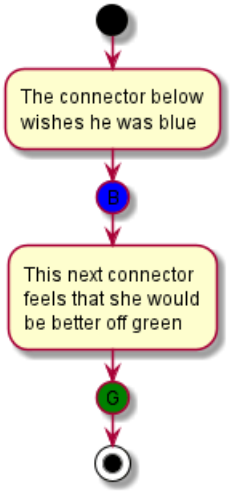
```
@startuml
start
:Some activity;
(A)
detach
(A)
:Other activity;
@enduml
```



6.15 Connecteurs en couleur

Vous pouvez ajouter des couleurs aux connecteurs.

```
@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
```

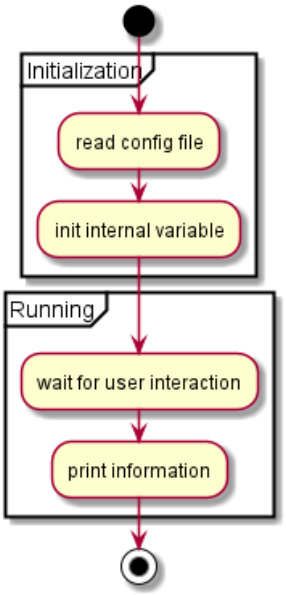


[Ref. QA-10077]

6.16 Groupement [partition]

Vous pouvez grouper les activités ensembles en définissant les partitions.

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

stop
@enduml
```

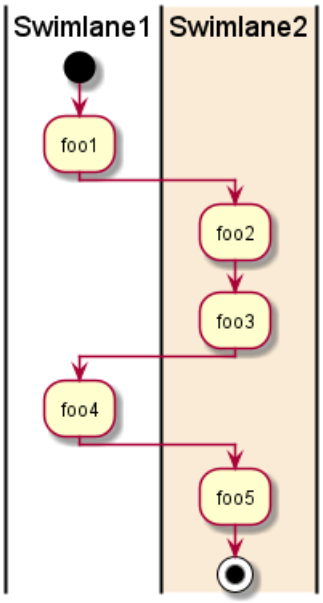


6.17 Couloirs

À l'aide du symbole |, il est possible de définir des couloirs d'éxecution.

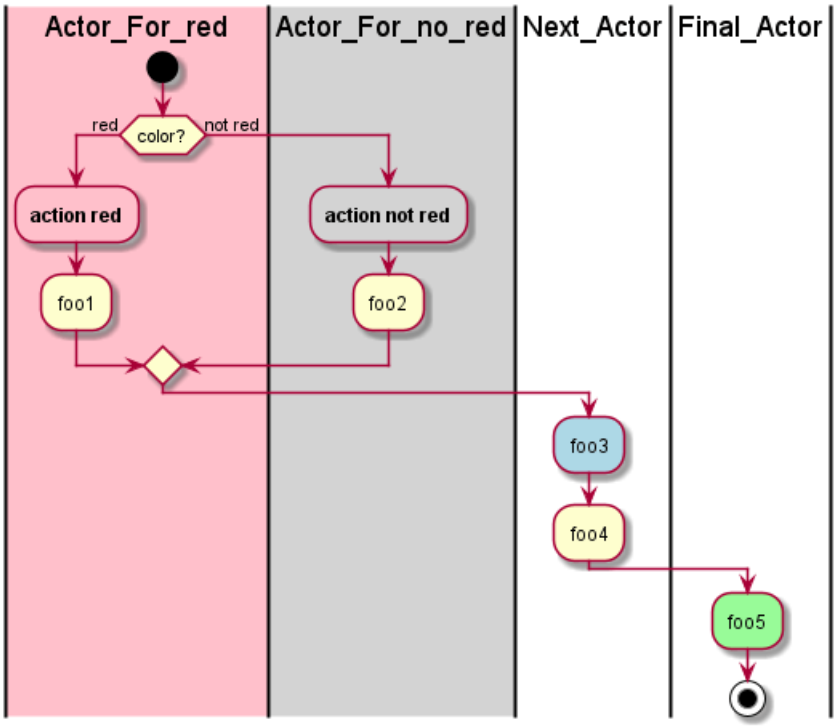
Il est aussi possible de changer la couleur d'un couloir.

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



You can add `if` conditional or `repeat` or `while` loop within swimlanes.

```
@startuml
|#pink|Actor_For_red|
start
if (color?) is (red) then
#pink:**action red**;
:foo1;
else (not red)
|#lightgray|Actor_For_no_red|
#lightgray:**action not red**;
:foo2;
endif
|Next_Actor|
#lightblue:foo3;
:foo4;
|Final_Actor|
#palegreen:foo5;
stop
@enduml
```

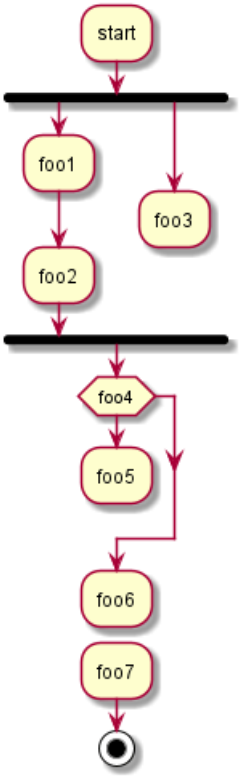


6.18 Détacher ou arrêter [detach, kill]

Il est possible de supprimer une flèche en utilisant le mot clé `detach` ou `kill` :

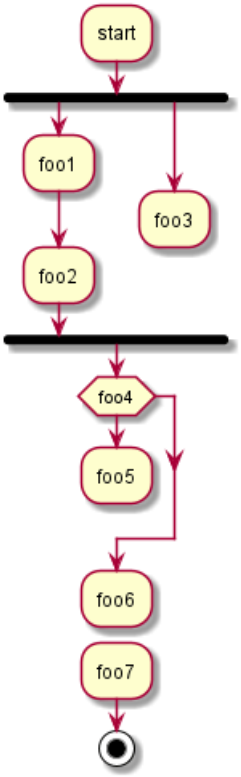
- `detach`

```
@startuml
 :start;
 fork
   :foo1;
   :foo2;
 fork again
   :foo3;
   detach
 endfork
 if (foo4) then
   :foo5;
   detach
 endif
 :foo6;
 detach
 :foo7;
 stop
@enduml
```



- kill

```
@startuml
 :start;
 fork
   :foo1;
   :foo2;
 fork again
   :foo3;
   kill
 endfork
 if (foo4) then
   :foo5;
   kill
 endif
 :foo6;
 kill
 :foo7;
 stop
@enduml
```



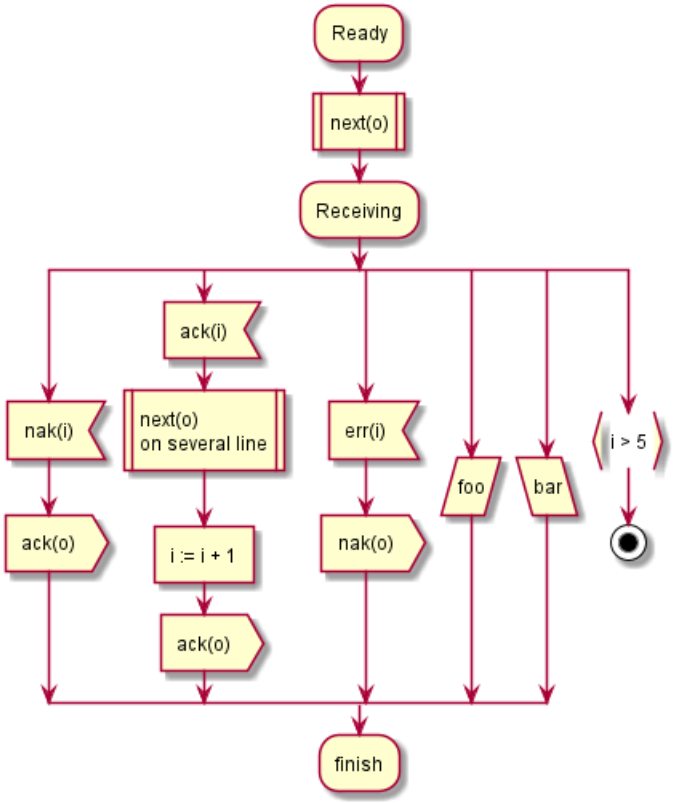
6.19 SDL (Specification and Description Language)

En changeant le séparateur final ;, vous pouvez déterminer différents rendus pour l' activité, conformément au *langage de description et de spécification (LDS)* ou *Specification and Description Language (SDL)* (en anglais) :

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

```
@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)>
```

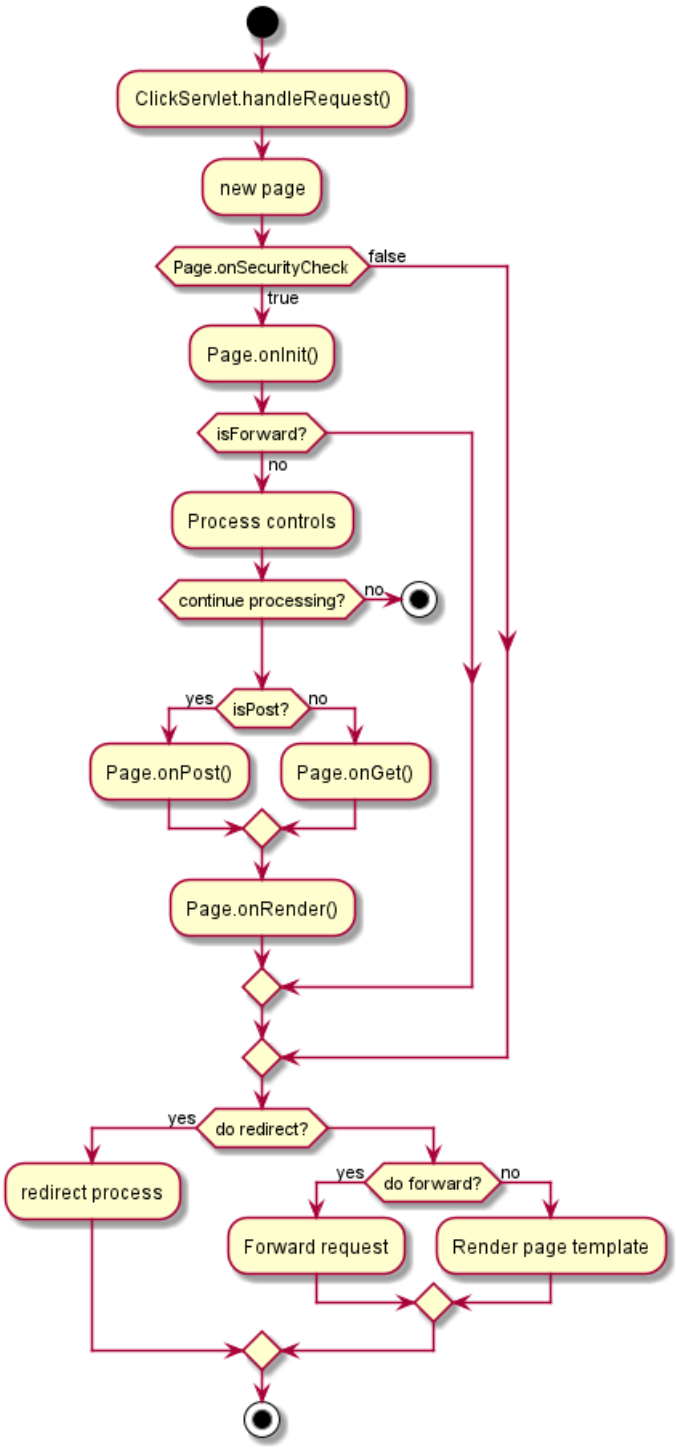
```
split again
 :foo/
split again
 :bar\\
split again
 :i > 5}
stop
end split
:finish;
@enduml
```



6.20 Exemple complet

```
@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
```

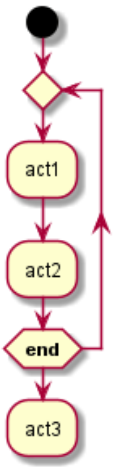


6.21 Condition Style

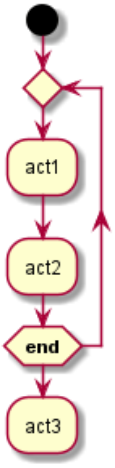
6.21.1 Inside style (by default)

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

```
@enduml
```

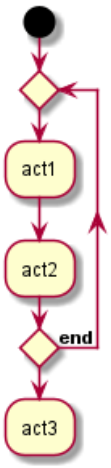



```
@startuml
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end)
:act3;
@enduml
```



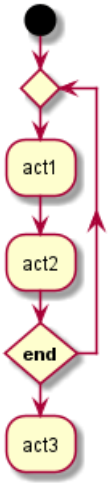
6.21.2 Diamond style

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

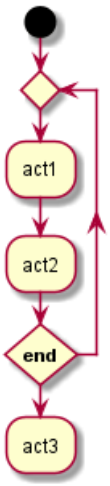


6.21.3 InsideDiamond (or *Foo1*) style

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




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



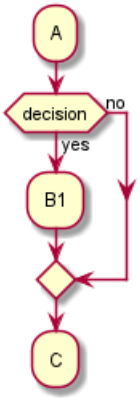
[Ref. QA-1290 and #400]

6.22 Condition End Style

6.22.1 Diamond style (by default)

- With one branch

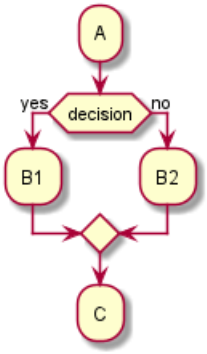
```
@startuml
skinparam ConditionEndStyle diamond
:A;
if (decision) then (yes)
    :B1;
else (no)
endif
:C;
@enduml
```



- With two branchs (`B1`, `B2`)

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

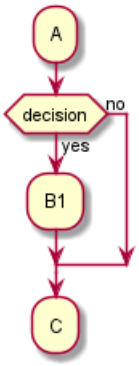
```
@enduml
```



6.22.2 Horizontal line (hline) style

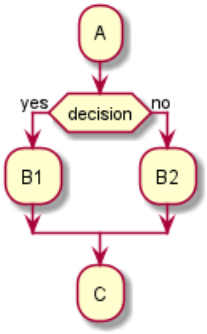
- With one branch

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



- With two branchs (`B1`, `B2`)

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



[Ref. QA-4015]

7 Diagrammes de composants

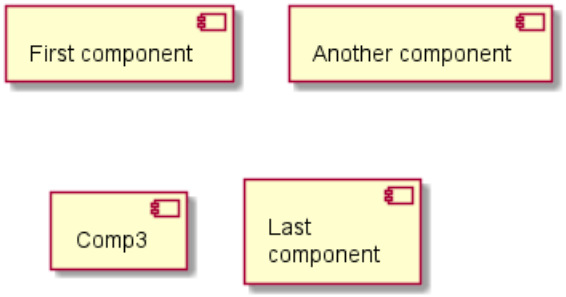
Prenons quelques exemples.

7.1 Composants

Les composants doivent être mis entre crochets.

Il est aussi possible d'utiliser le mot-clé `component` pour définir un composant. Et vous pouvez définir un alias, grâce au mot-clé `as`. Cet alias sera utile plus tard, pour définir des relations entre composants.

```
@startuml

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

@enduml
```



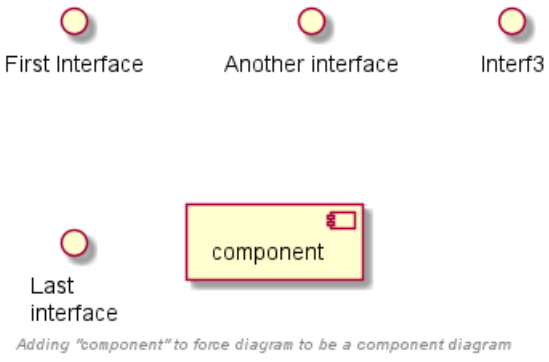
7.2 Interfaces

Les interfaces sont définies à l'aide du symbole `()` (parce que cela ressemble à un cercle).

Vous pouvez aussi utiliser le mot-clé `interface` pour définir une interface. Vous pouvez aussi définir un alias, à l'aide du mot-clé `as`. Cet alias pourrait être utilisé plus tard, lors de la définition des relations.

Nous verrons plus tard qu'il n'est pas obligatoire de définir les interfaces.

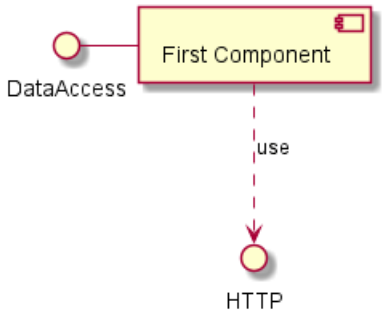
```
@startuml

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

[component]
footer //Adding "component" to force diagram to be a **component diagram**//
@enduml
```



7.3 Exemple simple

Les liens entre les éléments sont à utiliser avec des combinaisons de lignes pointillés (`..`), lignes droites(`--`), et de flèches (`-->`).

```
@startuml

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

@enduml
```

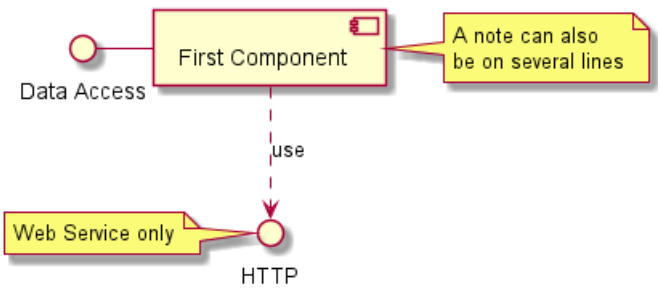


7.4 Mettre des notes

Vous pouvez utiliser les commandes suivantes : `note left of` , `note right of` , `note top of` , `note bottom of` suivi de l'identifiant du composant pour définir des notes reliées à un seul objet.

Une note peut aussi être créée seule avec le mot clé `note`, puis ensuite reliée à d'autres objets avec le symbole `...`

```
@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
```

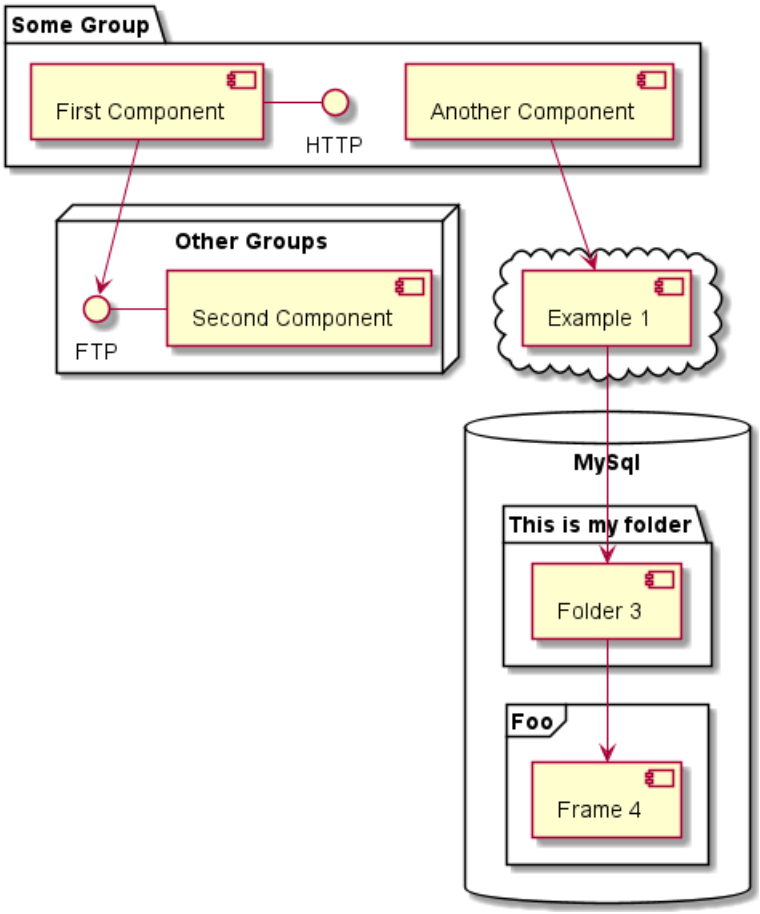


7.5 Regrouper des composants

Vous pouvez utiliser le mot-clé `package` pour regrouper des composants et des interfaces ensembles.

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

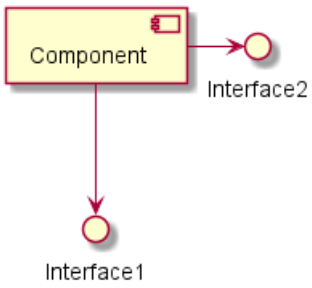
```
@startuml

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 Changer la direction des flèches

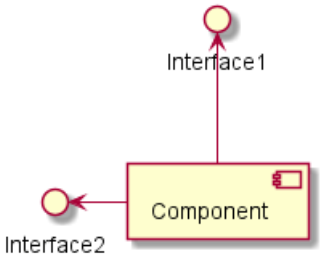
Par défaut, les liens entre classes ont deux tirets -- et sont orientées verticalement. Il est possible d'utiliser un lien horizontal en mettant un simple tiret (ou point) comme ceci :

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



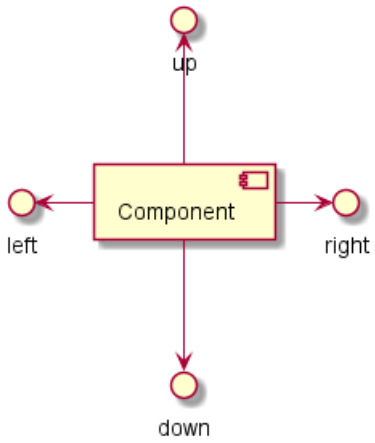
Vous pouvez aussi changer le sens en renversant le lien

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



Il est aussi possible de changer la direction des flèches en ajoutant les mots clés `left`, `right`, `up` ou `down` à l'intérieur des flèches :

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



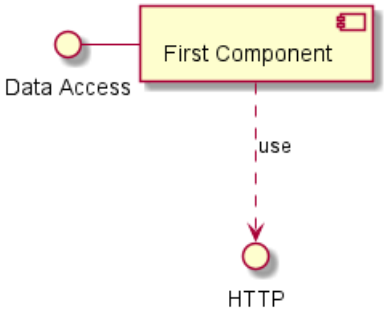
Vous pouvez raccourcir les flèches en utilisant seulement les premiers caractères de la direction (par exemple, `-d-` à la place de `-down-`) ou les deux premiers caractères (`-do-`).

Veuillez noter qu'il ne faut pas abuser de cette fonctionnalité : *Graphviz* donne généralement de bon résultat sans modification.

7.7 Utiliser la notation UML2

Par défaut *(à partir de la version v1.2020.13-14)*, la notation UML2 est utilisée.

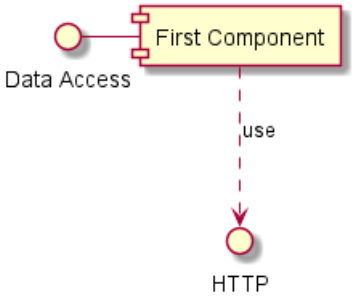
```
@startuml

interface "Data Access" as DA

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

@enduml
```



7.8 Utiliser la notation UML1

La commande `skinparam componentStyle uml1` est utilisée pour changer vers la notation UML1.

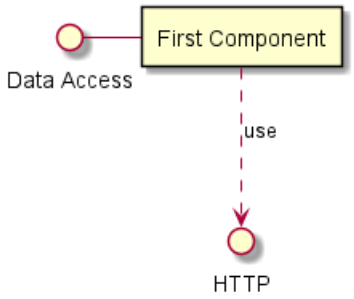
```
@startuml
skinparam componentStyle uml1

interface "Data Access" as DA

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

@enduml
```



7.9 Utiliser le style rectangle (supprime toute notation UML)

La commande `skinparam componentStyle rectangle` est utilisée pour changer vers le style rectangle *(sans aucune notation UML)*.

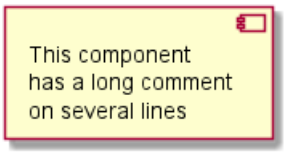
```
@startuml
skinparam componentStyle rectangle

interface "Data Access" as DA

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

@enduml
```



7.10 Description longue

Il est possible de mettre un long texte sur plusieurs lignes en utilisant des crochets.

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



7.11 Couleurs individuelles

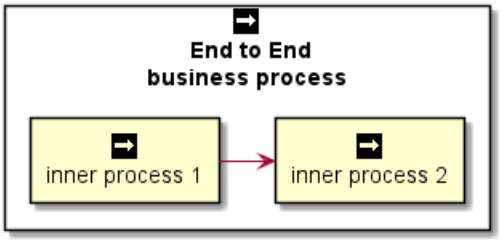
Il est possible de spécifier une couleur après la définition du composant.

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

7.12 Sprites et stéréotypes

Vous pouvez utiliser des sprites dans les stéréotypes des composants.

```
@startuml
sprite $businessProcess [16x16/16] {
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFF0FFFFF
FFFFFFFFFF00FFFF
FF00000000000FFF
FF0000000000000F
FF00000000000FFF
FFFFFFFFFF00FFFF
FFFFFFFFFF0FFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
}


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 Skinparam

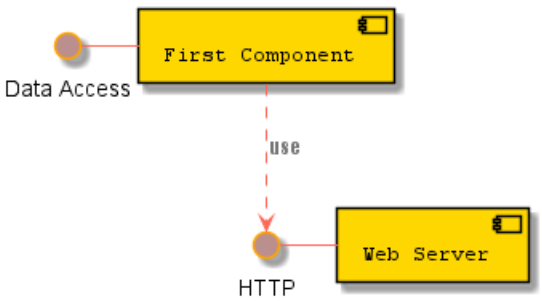
Utilisez la commande skinparam pour changer la couleur et la mise en forme du texte du schéma.

Vous pouvez utiliser cette commande :

- Dans la définition du diagramme, comme pour les autres commandes,
- Dans un fichier inclus,
- Dans un fichier de configuration, renseigné dans la ligne de commande ou la tâche ANT.

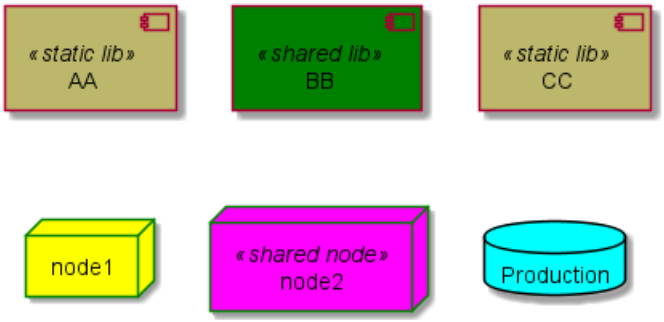
Vous pouvez définir des couleurs et des fontes spécifiques pour les composants et interfaces stéréotypés.

```
@startuml

skinparam interface {
  backgroundColor RosyBrown
  borderColor orange
}

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

() "Data Access" as DA

DA - [First Component]
[First Component] ..> () HTTP : use
HTTP - [Web Server] << Apache >>

@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
```

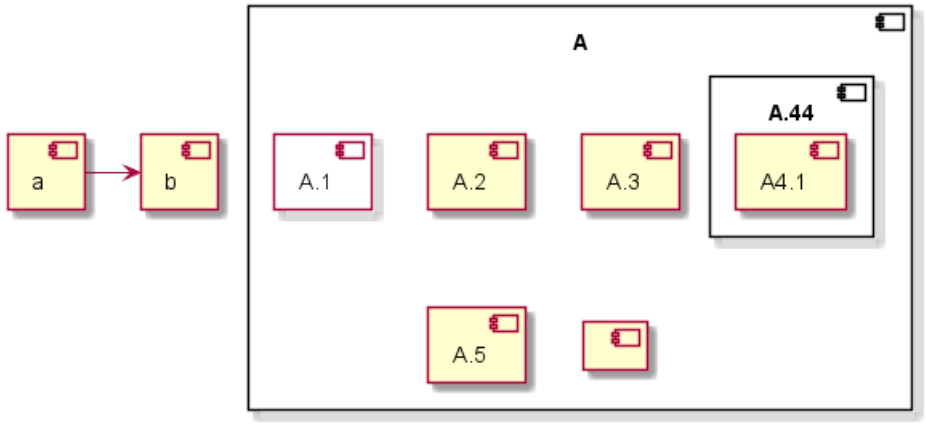


7.14 Specific SkinParameter

7.14.1 componentStyle

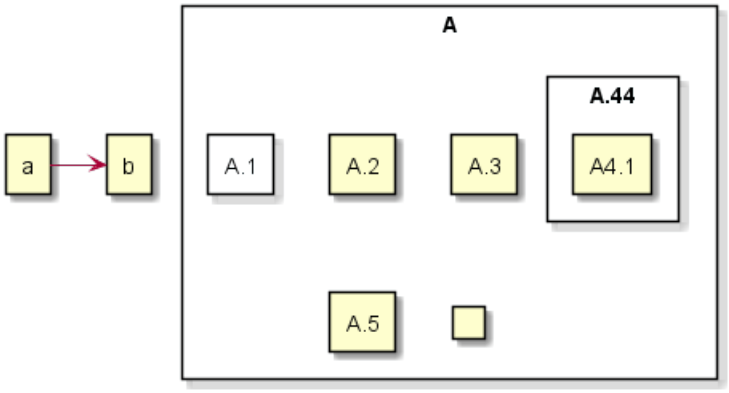
- By default (or with `skinparam componentStyle uml2`), you have an icon for component

```
@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
```



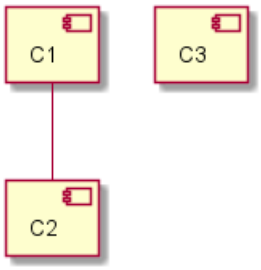
[Ref. 10798]

7.15 Hide or Remove unlinked component

By default, all components are displayed:

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

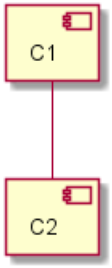
```
C1 -- C2
@enduml
```



But you can:

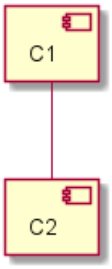
- `hide @unlinked` components:

```
@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
```

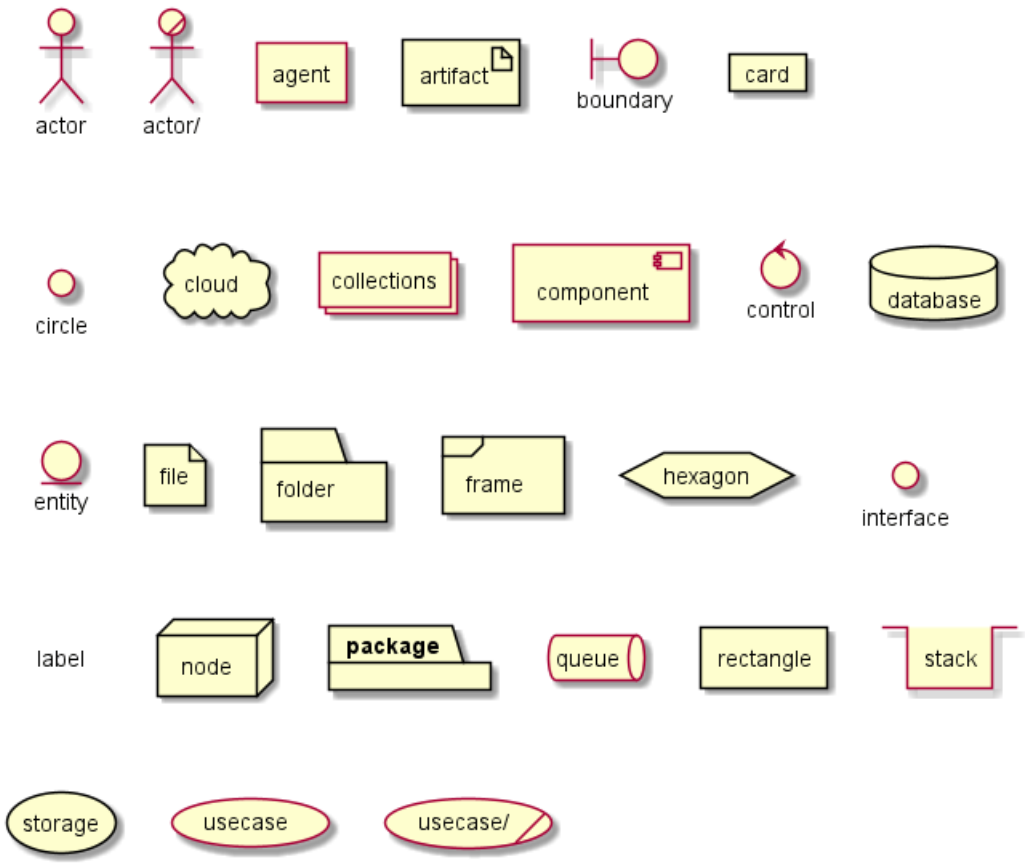


[Ref. QA-11052]

8 Deployment Diagram

8.1 Declaring element

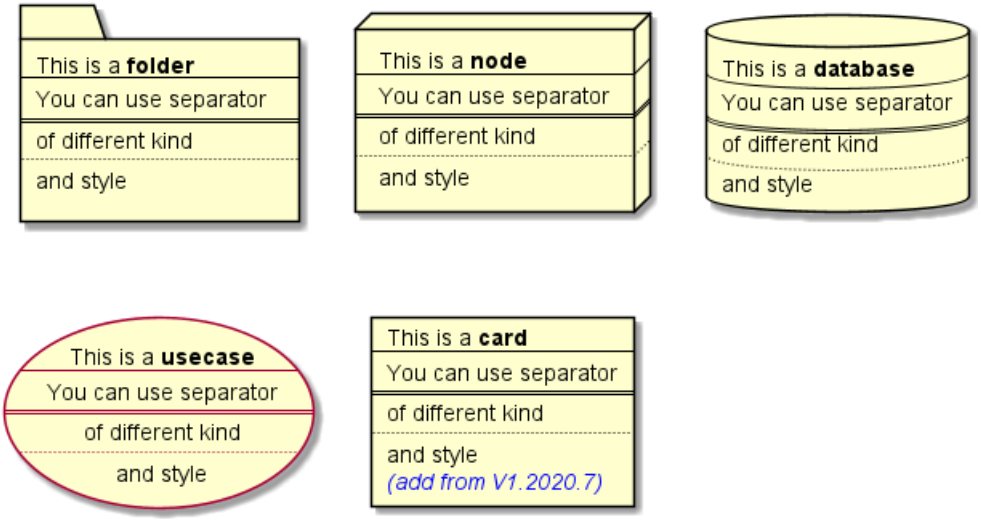
```
@startuml
actor actor
actor/ "actor/"
agent agent
artifact artifact
boundary boundary
card card
circle circle
cloud cloud
collections collections
component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label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



You can optionaly 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
```



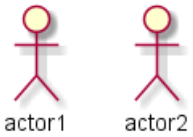
8.2 Declaring element (using short form)

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

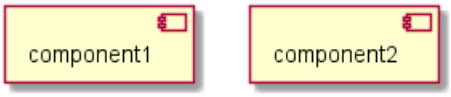
```
@startuml

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

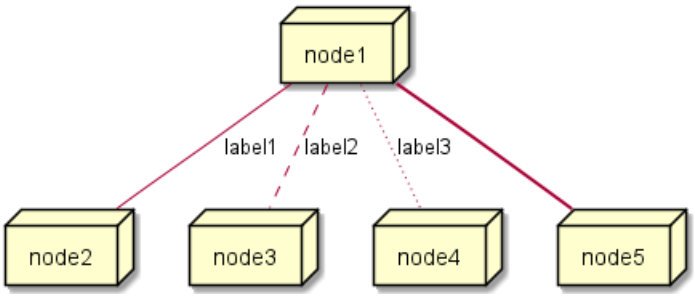
```
@startuml

usecase usecase1
(usecase2)

@enduml
```

8.3 Linking or arrow

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

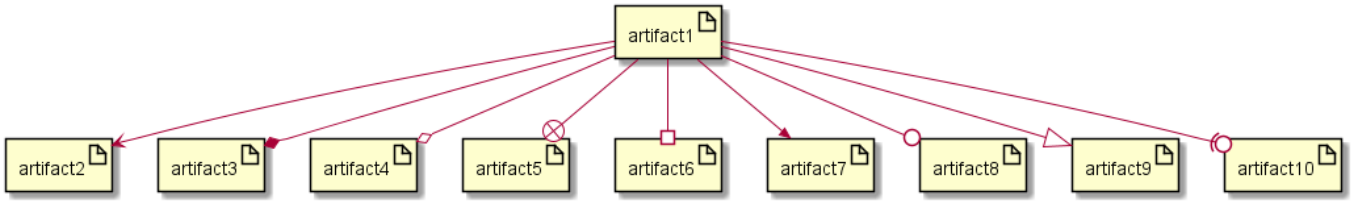
```
@startuml

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:

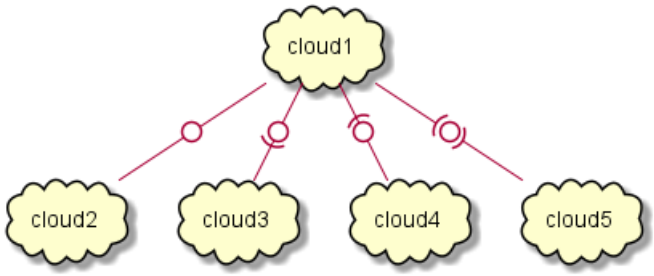
```
@startuml

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 --0 artifact8
artifact1 --^ artifact9
artifact1 --(0 artifact10

@enduml
```



You can also have the following types:

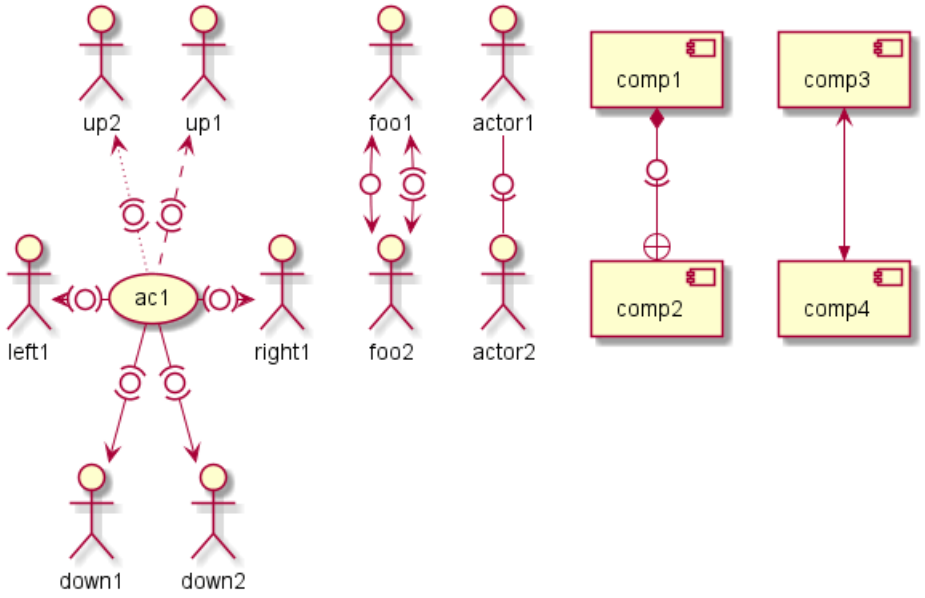
```
@startuml

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]
@enduml
```



[Ref. QA-1736]

See all type on **Appendix**.

8.4 Bracketed arrow style

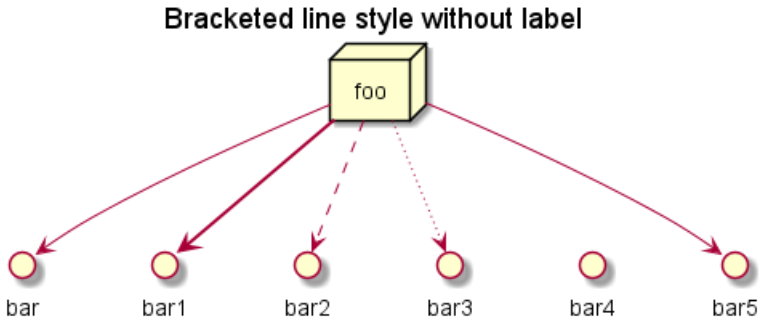
*Similar as Bracketed **class** relations (linking or arrow) style*

8.4.1 Line style

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

- without label

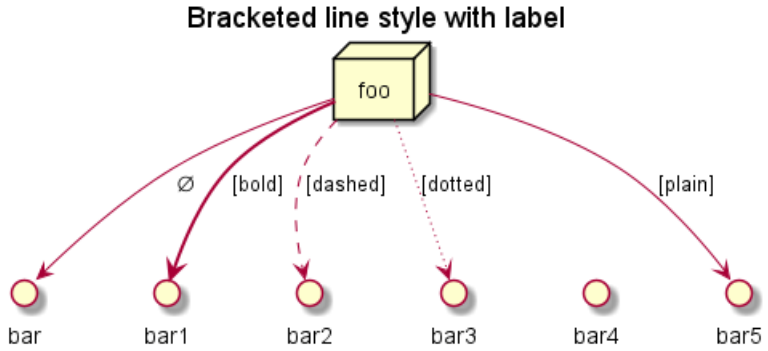
```
@startuml
node foo
title Bracketed line style without label
foo --> bar
foo -[bold]-> bar1
foo -[dashed]-> bar2
foo -[dotted]-> bar3
foo -[hidden]-> bar4
foo -[plain]-> bar5
@enduml
```



- with label

```
@startuml
title Bracketed line style with label
node foo
```

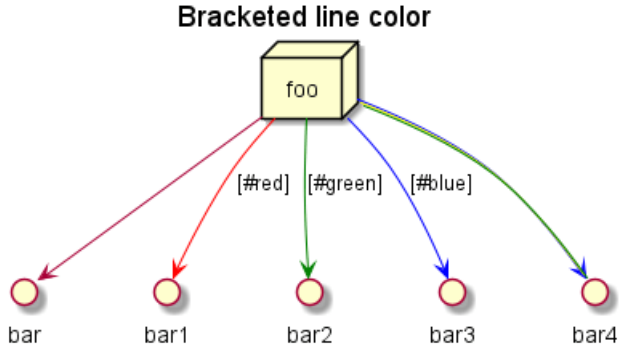
```
foo --> bar           :
foo -[bold]-> bar1    : [bold]
foo -[dashed]-> bar2  : [dashed]
foo -[dotted]-> bar3  : [dotted]
foo -[hidden]-> bar4  : [hidden]
foo -[plain]-> bar5   : [plain]
@enduml
```



[Adapted from QA-4181]

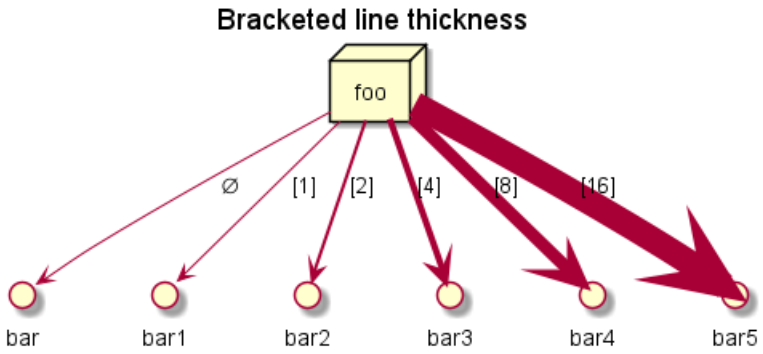
8.4.2 Line color

```
@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

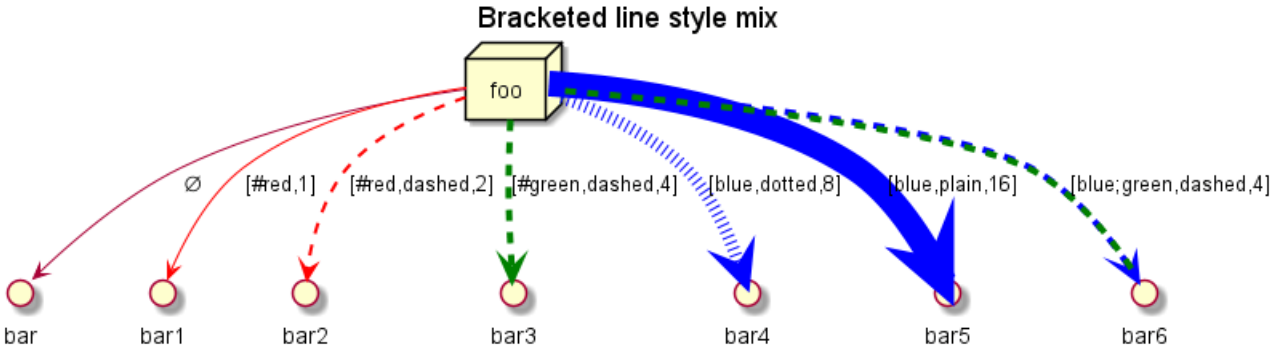
```
@startuml
title Bracketed line thickness
node foo
foo --> bar                 :
foo -[thickness=1]-> bar1   : [1]
foo -[thickness=2]-> bar2   : [2]
foo -[thickness=4]-> bar3   : [4]
foo -[thickness=8]-> bar4   : [8]
foo -[thickness=16]-> bar5  : [16]
@enduml
```



[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
```

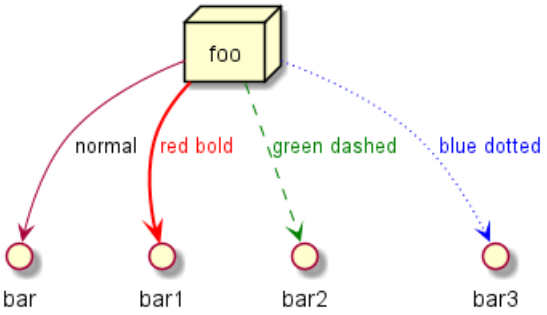


8.5 Change arrow color and style (inline style)

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
```



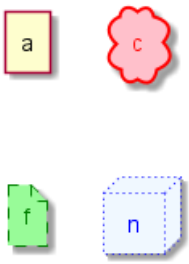
[Ref. QA-3770 and QA-3816] [See similar feature on class diagram]

8.6 Change element color and style (inline style)

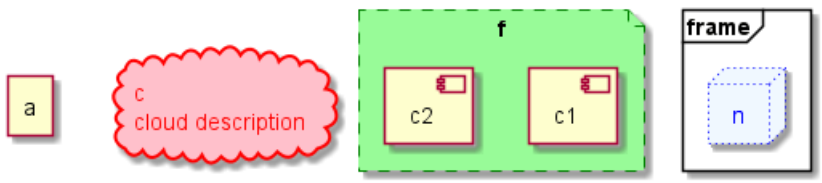
You can change the color or style of individual element using the following notation:

- `#[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:

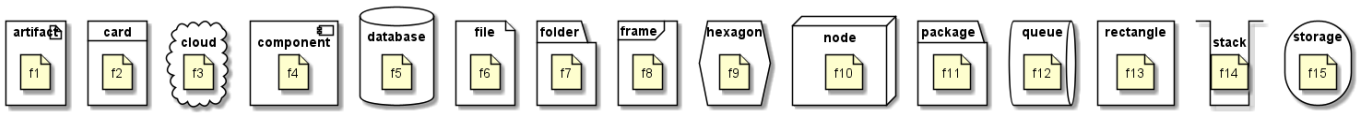
```
@startuml
artifact artifact {
}
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
}
node node {
}
package package {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml
```

8.8 Packages and nested elements

8.8.1 Example with one level

```
@startuml
artifact    artifactVeryLOOOOOOOOOOOOOOOOOOg    as "artifact" {
file f1
}
card        cardVeryLOOOOOOOOOOOOOOOOOOg        as "card" {
file f2
}
cloud       cloudVeryLOOOOOOOOOOOOOOOOOOg       as "cloud" {
file f3
}
component   componentVeryLOOOOOOOOOOOOOOOOOOg   as "component" {
file f4
}
database    databaseVeryLOOOOOOOOOOOOOOOOOOg    as "database" {
file f5
```

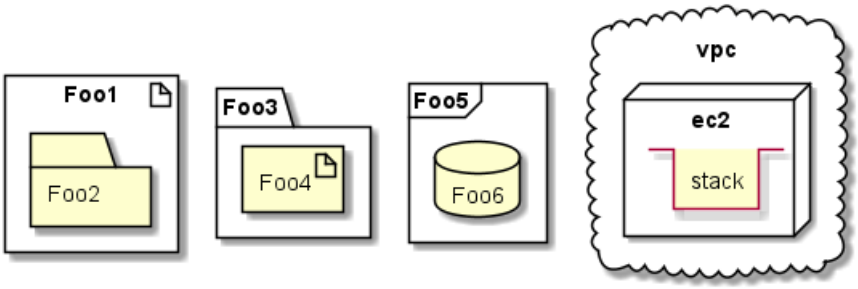
```
}
file        fileVeryLOOOOOOOOOOOOOOOOOOg        as "file" {
file f6
}
folder      folderVeryLOOOOOOOOOOOOOOOOOOg      as "folder" {
file f7
}
frame       frameVeryLOOOOOOOOOOOOOOOOOOg       as "frame" {
file f8
}
hexagon     hexagonVeryLOOOOOOOOOOOOOOOOOOg     as "hexagon" {
file f9
}
node        nodeVeryLOOOOOOOOOOOOOOOOOOg        as "node" {
file f10
}
package     packageVeryLOOOOOOOOOOOOOOOOOOg     as "package" {
file f11
}
queue       queueVeryLOOOOOOOOOOOOOOOOOOg       as "queue" {
file f12
}
rectangle   rectangleVeryLOOOOOOOOOOOOOOOOOOg   as "rectangle" {
file f13
}
stack       stackVeryLOOOOOOOOOOOOOOOOOOg       as "stack" {
file f14
}
storage     storageVeryLOOOOOOOOOOOOOOOOOOg     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
```

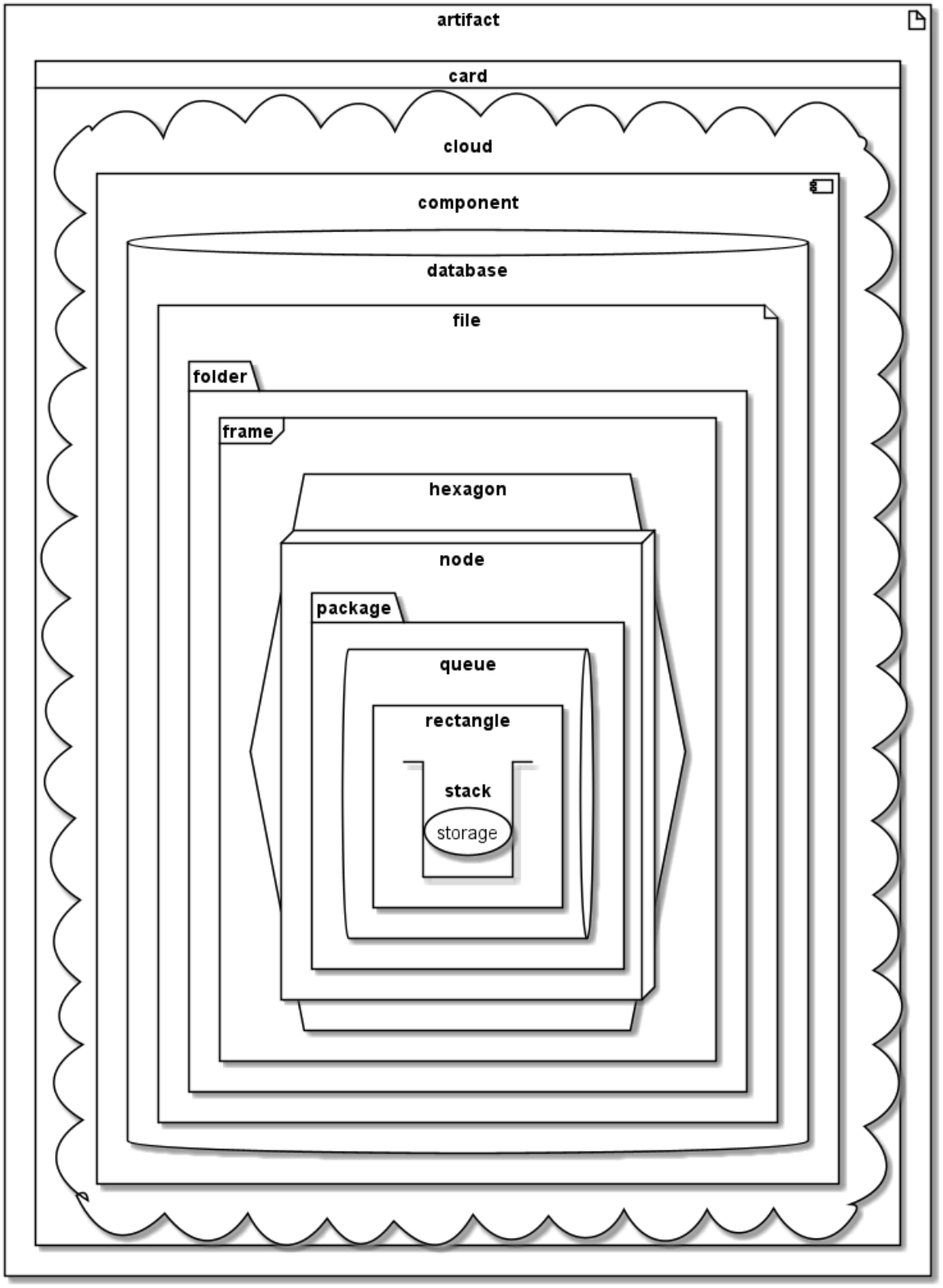
8.8.3 Full nesting

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 {
```

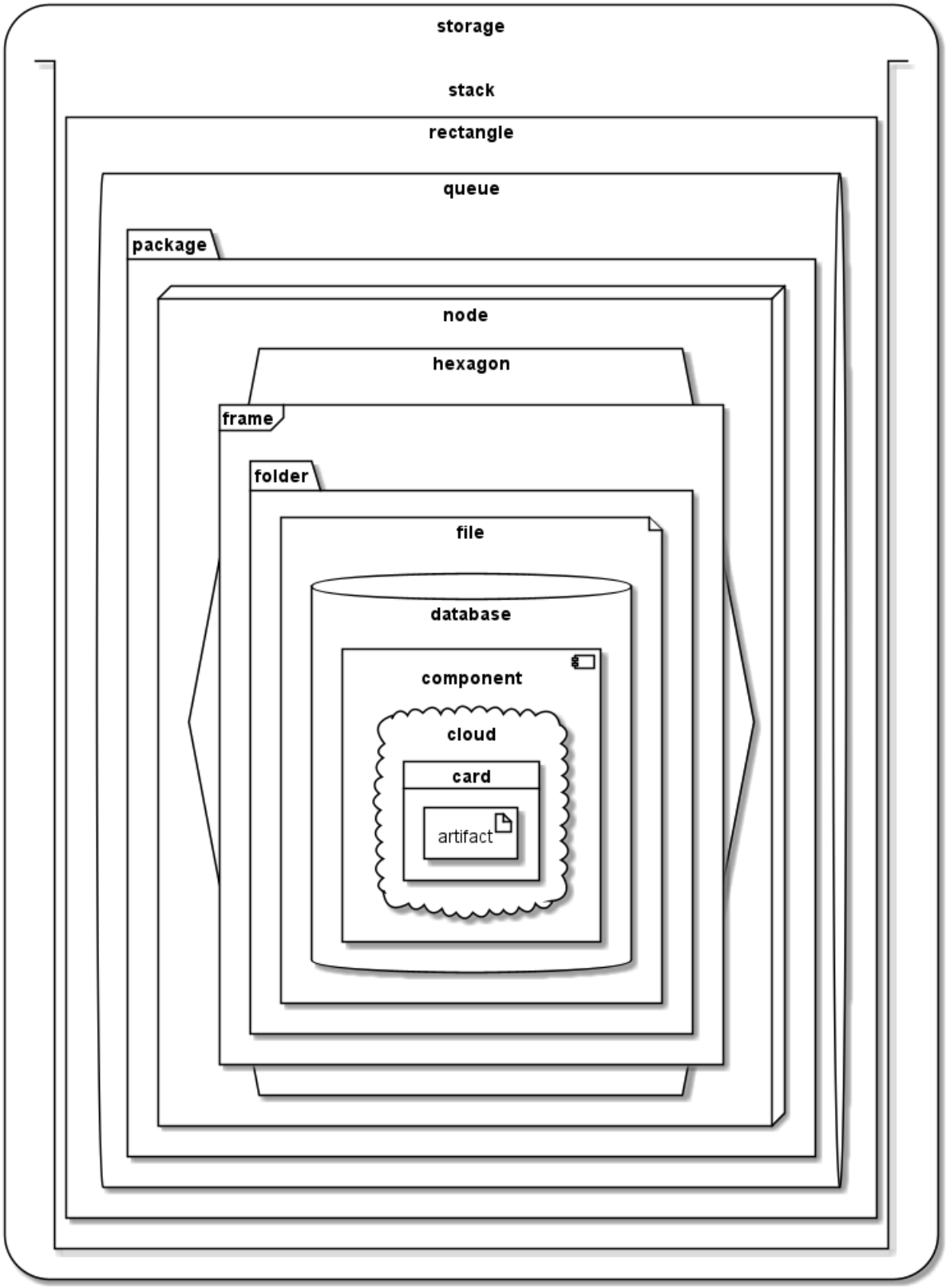
```
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
@enduml
```



- or reverse alphabetical order

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

```
package package {
node node {
hexagon hexagon {
frame frame {
folder folder {
file file {
database database {
component component {
cloud cloud {
card card {
artifact artifact {
}
}
}
}
}
}
}
}
}
}
}
}
}
@enduml
```

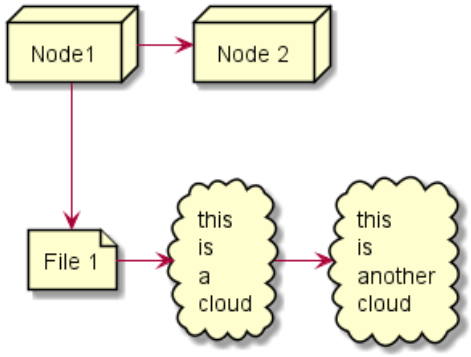


8.9 Alias

8.9.1 Simple alias with as

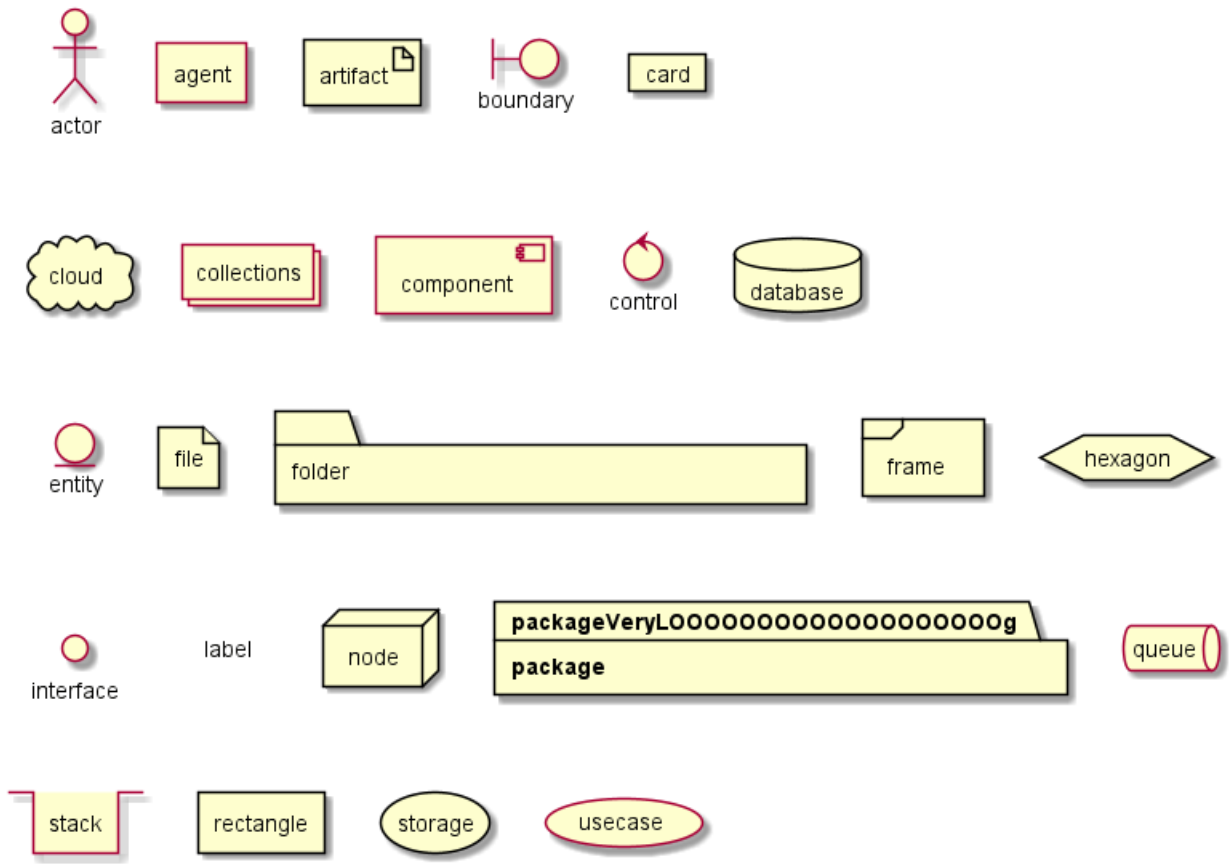
```
@startuml
node Node1 as n1
node "Node 2" as n2
```

```
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
```

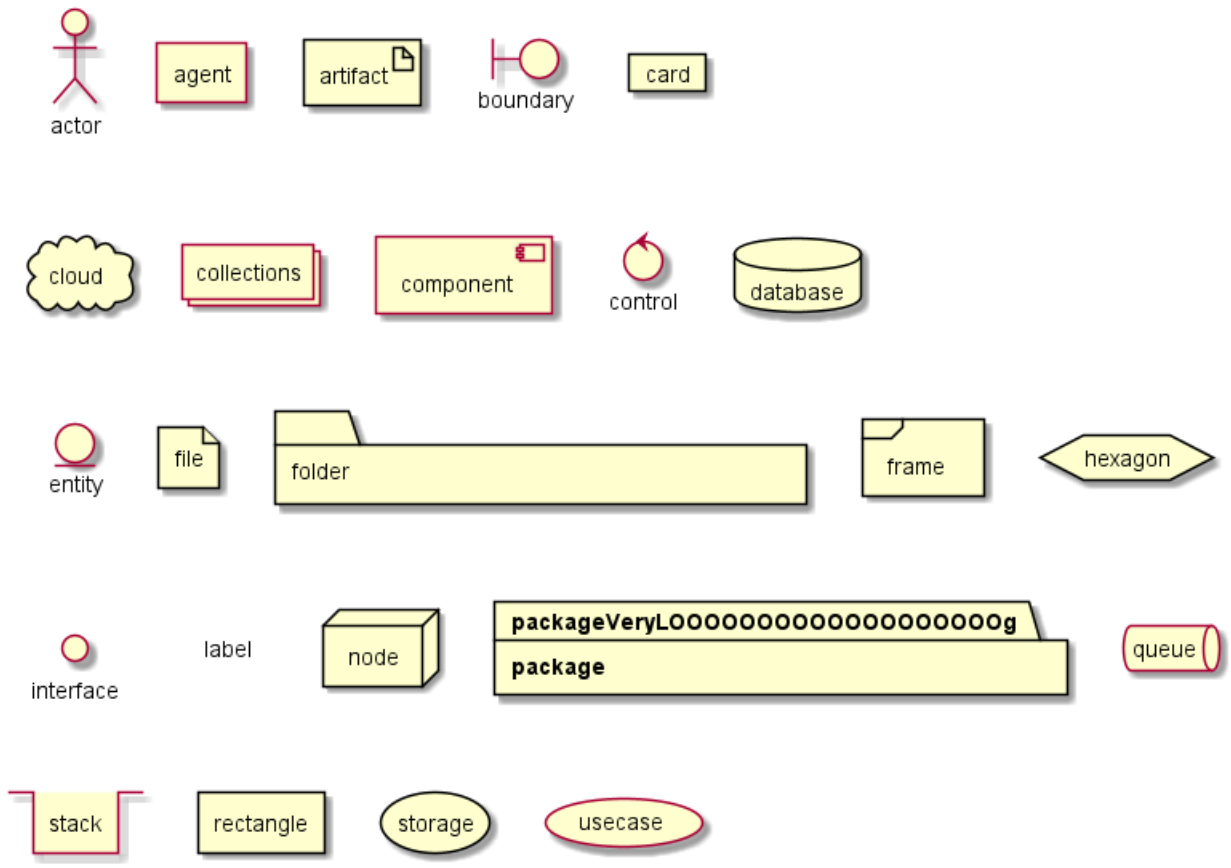


8.9.2 Examples of long alias

```
@startuml
actor        "actor"       as actorVeryLOOOOOOOOOOOOOOOOOOg
agent        "agent"       as agentVeryLOOOOOOOOOOOOOOOOOOg
artifact     "artifact"    as artifactVeryLOOOOOOOOOOOOOOOOOOg
boundary     "boundary"    as boundaryVeryLOOOOOOOOOOOOOOOOOOg
card         "card"        as cardVeryLOOOOOOOOOOOOOOOOOOg
cloud        "cloud"       as cloudVeryLOOOOOOOOOOOOOOOOOOg
collections  "collections" as collectionsVeryLOOOOOOOOOOOOOOOOOOg
component    "component"   as componentVeryLOOOOOOOOOOOOOOOOOOg
control      "control"     as controlVeryLOOOOOOOOOOOOOOOOOOg
database     "database"    as databaseVeryLOOOOOOOOOOOOOOOOOOg
entity       "entity"      as entityVeryLOOOOOOOOOOOOOOOOOOg
file         "file"        as fileVeryLOOOOOOOOOOOOOOOOOOg
folder       "folder"      as folderVeryLOOOOOOOOOOOOOOOOOOg
frame        "frame"       as frameVeryLOOOOOOOOOOOOOOOOOOg
hexagon      "hexagon"     as hexagonVeryLOOOOOOOOOOOOOOOOOOg
interface    "interface"   as interfaceVeryLOOOOOOOOOOOOOOOOOOg
label        "label"       as labelVeryLOOOOOOOOOOOOOOOOOOg
node         "node"        as nodeVeryLOOOOOOOOOOOOOOOOOOg
package      "package"     as packageVeryLOOOOOOOOOOOOOOOOOOg
queue        "queue"       as queueVeryLOOOOOOOOOOOOOOOOOOg
stack        "stack"       as stackVeryLOOOOOOOOOOOOOOOOOOg
rectangle    "rectangle"   as rectangleVeryLOOOOOOOOOOOOOOOOOOg
storage      "storage"     as storageVeryLOOOOOOOOOOOOOOOOOOg
usecase      "usecase"     as usecaseVeryLOOOOOOOOOOOOOOOOOOg
@enduml
```



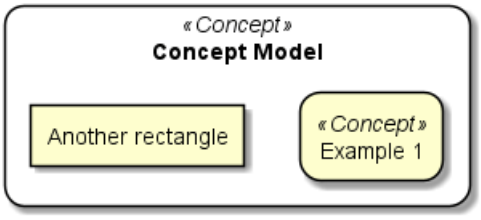

```
@startuml
actor       actorVeryLOOOOOOOOOOOOOOOOOOg       as "actor"
agent       agentVeryLOOOOOOOOOOOOOOOOOOg       as "agent"
artifact    artifactVeryLOOOOOOOOOOOOOOOOOOg    as "artifact"
boundary    boundaryVeryLOOOOOOOOOOOOOOOOOOg    as "boundary"
card        cardVeryLOOOOOOOOOOOOOOOOOOg        as "card"
cloud       cloudVeryLOOOOOOOOOOOOOOOOOOg       as "cloud"
collections collectionsVeryLOOOOOOOOOOOOOOOOOOg as "collections"
component   componentVeryLOOOOOOOOOOOOOOOOOOg   as "component"
control     controlVeryLOOOOOOOOOOOOOOOOOOg     as "control"
database    databaseVeryLOOOOOOOOOOOOOOOOOOg    as "database"
entity      entityVeryLOOOOOOOOOOOOOOOOOOg      as "entity"
file        fileVeryLOOOOOOOOOOOOOOOOOOg        as "file"
folder      folderVeryLOOOOOOOOOOOOOOOOOOg      as "folder"
frame       frameVeryLOOOOOOOOOOOOOOOOOOg       as "frame"
hexagon     hexagonVeryLOOOOOOOOOOOOOOOOOOg     as "hexagon"
interface   interfaceVeryLOOOOOOOOOOOOOOOOOOg   as "interface"
label       labelVeryLOOOOOOOOOOOOOOOOOOg       as "label"
node        nodeVeryLOOOOOOOOOOOOOOOOOOg        as "node"
package     packageVeryLOOOOOOOOOOOOOOOOOOg     as "package"
queue       queueVeryLOOOOOOOOOOOOOOOOOOg       as "queue"
stack       stackVeryLOOOOOOOOOOOOOOOOOOg       as "stack"
rectangle   rectangleVeryLOOOOOOOOOOOOOOOOOOg   as "rectangle"
storage     storageVeryLOOOOOOOOOOOOOOOOOOg     as "storage"
usecase     usecaseVeryLOOOOOOOOOOOOOOOOOOg     as "usecase"
@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
```

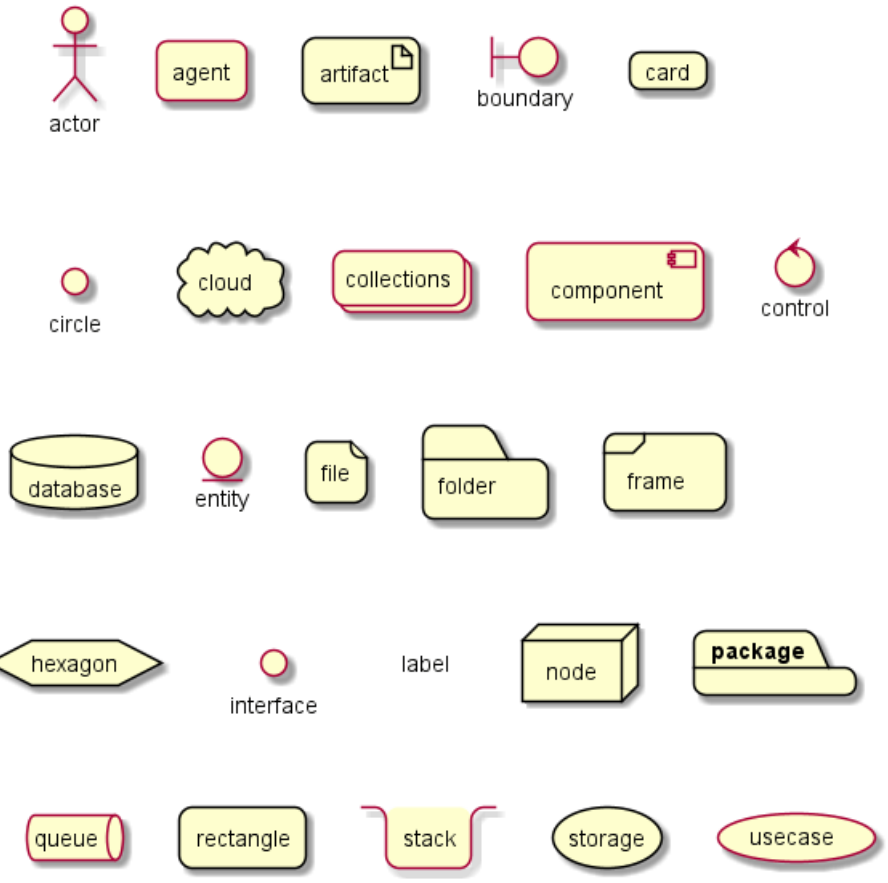


8.11 Specific SkinParameter

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
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
@enduml
```

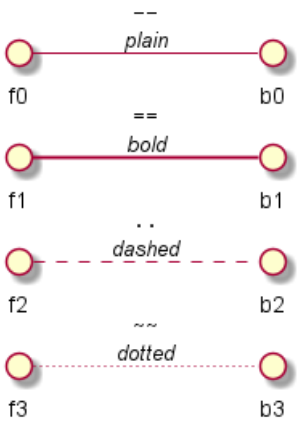


[Ref. QA-5299, QA-6915, QA-11943]

8.12 Appendix: All type of arrow line

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

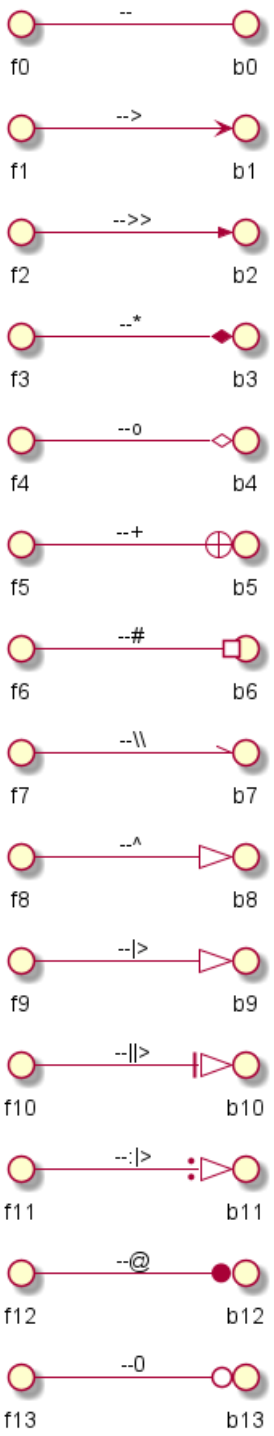
```
skinparam nodesep 5

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

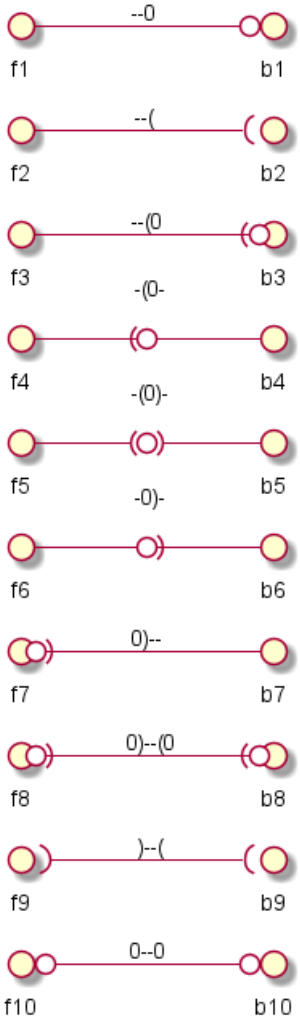
```
@startuml
left to right direction
skinparam nodesep 5

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)-\n ""
f5 -(0)- b5  : "" -(0)-\n""
```

```
f4 -(0-  b4  : "" -(0-\n ""
f3 --(0  b3  : "" --(0 ""
f2 --(   b2  : "" --(  ""
f1 --0   b1  : "" --0  ""
@enduml
```

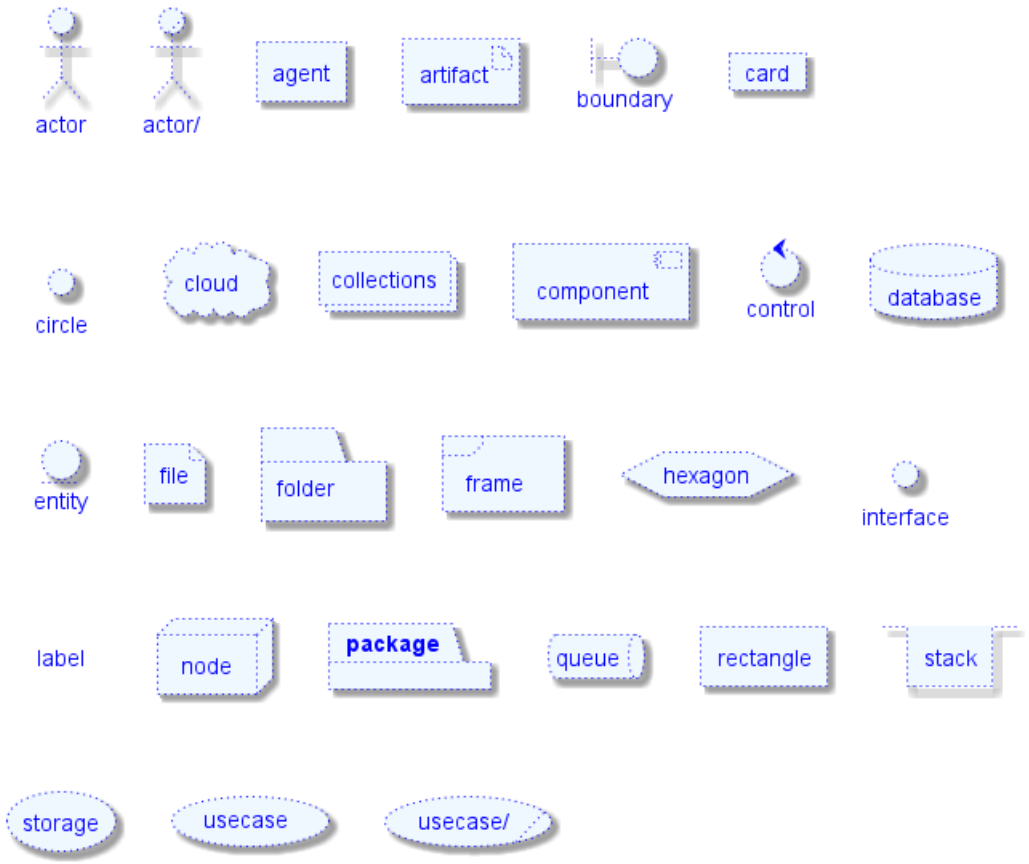


8.14 Appendix: Test of inline style on all element

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
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

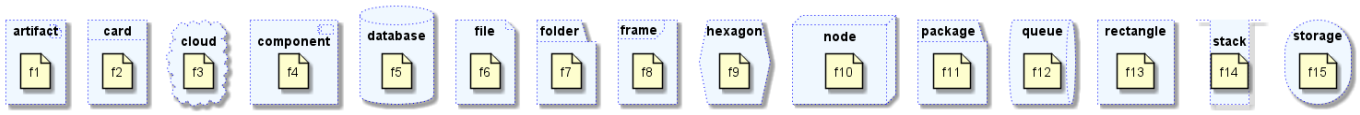
```
@startuml
artifact artifact #aliceblue;line:blue;line.dotted;text:blue {
}
card card #aliceblue;line:blue;line.dotted;text:blue {
}
cloud cloud #aliceblue;line:blue;line.dotted;text:blue {
}
component component #aliceblue;line:blue;line.dotted;text:blue {
}
database database #aliceblue;line:blue;line.dotted;text:blue {
}
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    artifactVeryLOOOOOOOOOOOOOOOOOOOg    as "artifact" #aliceblue;line:blue;line.dotted;text
file f1
}
card        cardVeryLOOOOOOOOOOOOOOOOOOg        as "card" #aliceblue;line:blue;line.dotted;text:blu
file f2
}
cloud       cloudVeryLOOOOOOOOOOOOOOOOOOg       as "cloud" #aliceblue;line:blue;line.dotted;text:bl
file f3
}
component   componentVeryLOOOOOOOOOOOOOOOOOOg   as "component" #aliceblue;line:blue;line.dotted;tex
file f4
}
database    databaseVeryLOOOOOOOOOOOOOOOOOOg    as "database" #aliceblue;line:blue;line.dotted;text
file f5
}
file        fileVeryLOOOOOOOOOOOOOOOOOOg        as "file" #aliceblue;line:blue;line.dotted;text:blu
file f6
}
folder      folderVeryLOOOOOOOOOOOOOOOOOOg      as "folder" #aliceblue;line:blue;line.dotted;text:b
file f7
}
frame       frameVeryLOOOOOOOOOOOOOOOOOOg       as "frame" #aliceblue;line:blue;line.dotted;text:bl
file f8
}
hexagon     hexagonVeryLOOOOOOOOOOOOOOOOOOg     as "hexagon" #aliceblue;line:blue;line.dotted;text:
file f9
}
node        nodeVeryLOOOOOOOOOOOOOOOOOOg        as "node" #aliceblue;line:blue;line.dotted;text:blu
file f10
}
package     packageVeryLOOOOOOOOOOOOOOOOOOg     as "package" #aliceblue;line:blue;line.dotted;text:
file f11
```

```
}
queue       queueVeryLOOOOOOOOOOOOOOOOOOg       as "queue" #aliceblue;line:blue;line.dotted;text:blu
file f12
}
rectangle   rectangleVeryLOOOOOOOOOOOOOOOOOOg   as "rectangle" #aliceblue;line:blue;line.dotted;text
file f13
}
stack       stackVeryLOOOOOOOOOOOOOOOOOOg       as "stack" #aliceblue;line:blue;line.dotted;text:blu
file f14
}
storage     storageVeryLOOOOOOOOOOOOOOOOOOg     as "storage" #aliceblue;line:blue;line.dotted;text:
file f15
}
@enduml
```



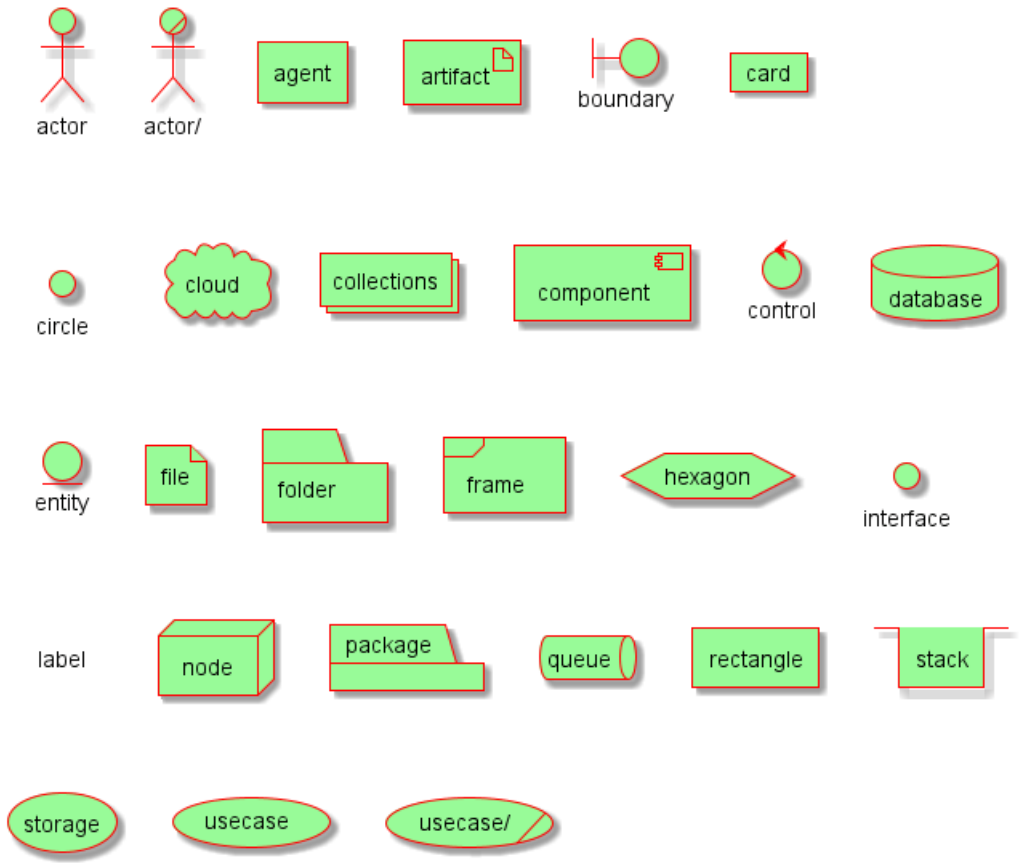
8.15 Appendix: Test of style on all element

8.15.1 Simple element

8.15.2 Global style (on componentDiagram)

```
@startuml
<style>
componentDiagram {
  BackGroundColor palegreen
  LineThickness 1
  LineColor red
}
</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
queue queue
rectangle rectangle
stack stack
storage storage
```

```
usecase usecase
usecase/ "usecase/"
@enduml
```



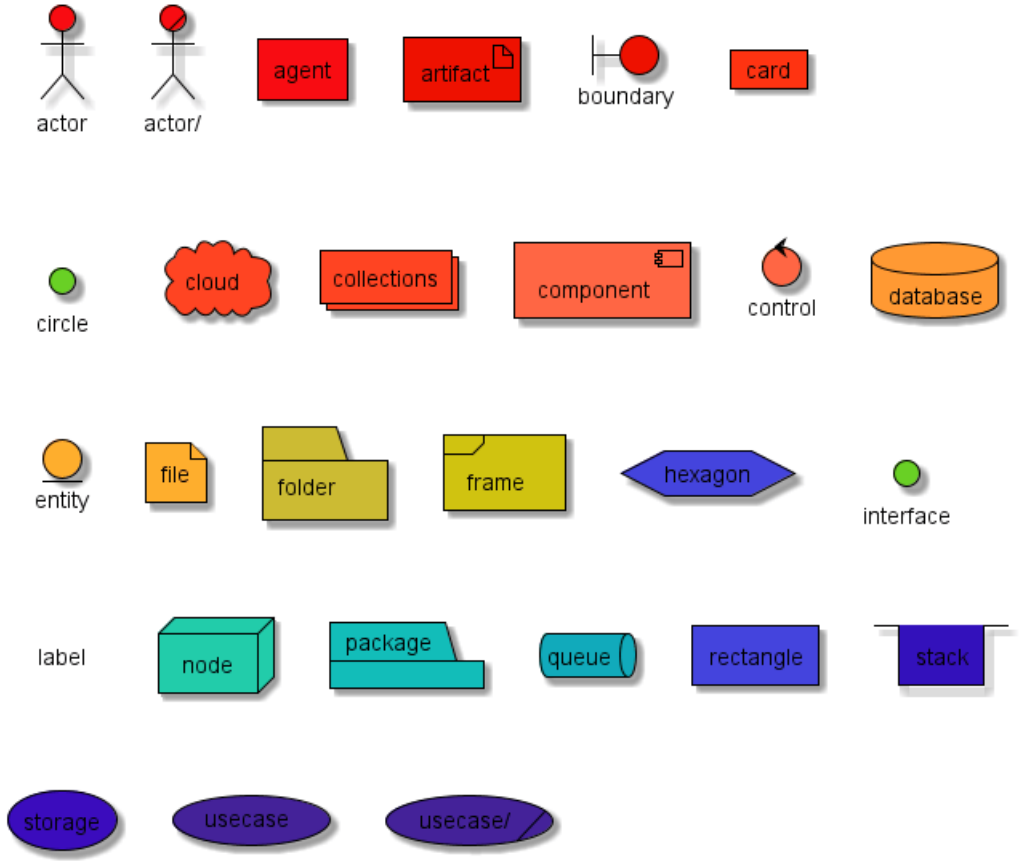
8.15.3 Style for each element

```
@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
}
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
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
usecase/ "usecase/"
@enduml
```



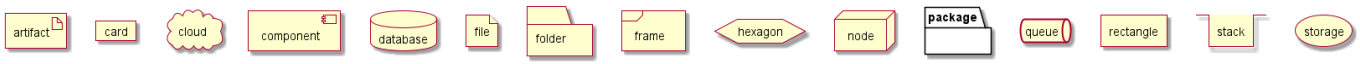
[Ref. QA-13261]

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 {
  BackGroundColor #3311bb
  LineThickness 1
  LineColor black
}
storage {
  BackGroundColor #3b0cbd
  LineThickness 1
  LineColor black
}

</style>
artifact artifact {
}
```

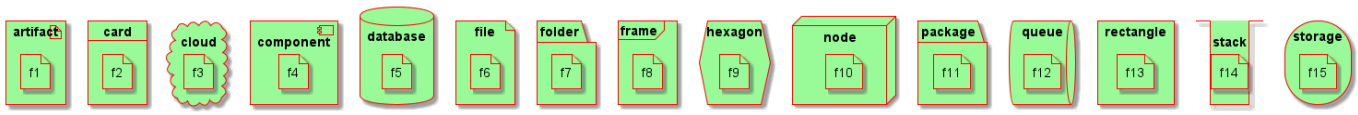
```
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
}
node node {
}
package package {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml
```

8.15.7 Nested element (with one level)

8.15.8 Global style (on componentDiagram)

```
@startuml
<style>
componentDiagram {
  BackGroundColor palegreen
  LineThickness 1
  LineColor red
}
</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
```

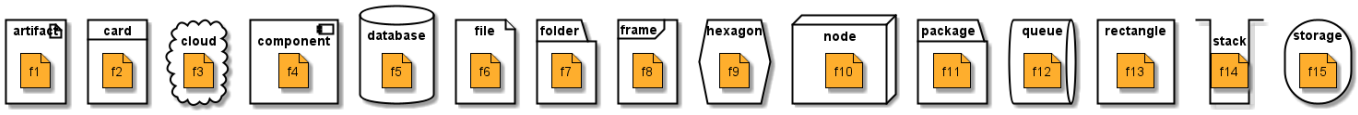


8.15.9 Style for each nested element

```
@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
```



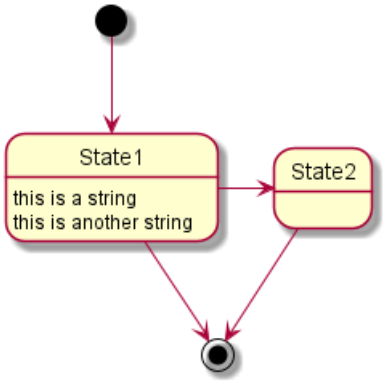
9 Diagrammes d'état

9.1 Exemple simple

Vous devez utiliser `[*]` pour le début et la fin du diagramme d'état.

Utilisez `-->` pour les flèches.

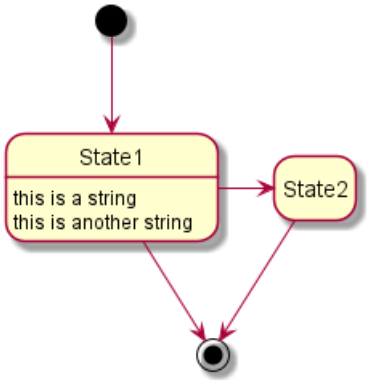
```
@startuml

[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string

State1 -> State2
State2 --> [*]

@enduml
```



9.2 Autre rendu

Il est possible d'utiliser la directive `hide empty description` pour afficher l'état de façon plus compact.

```
@startuml
hide empty description
[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string

State1 -> State2
State2 --> [*]
@enduml
```

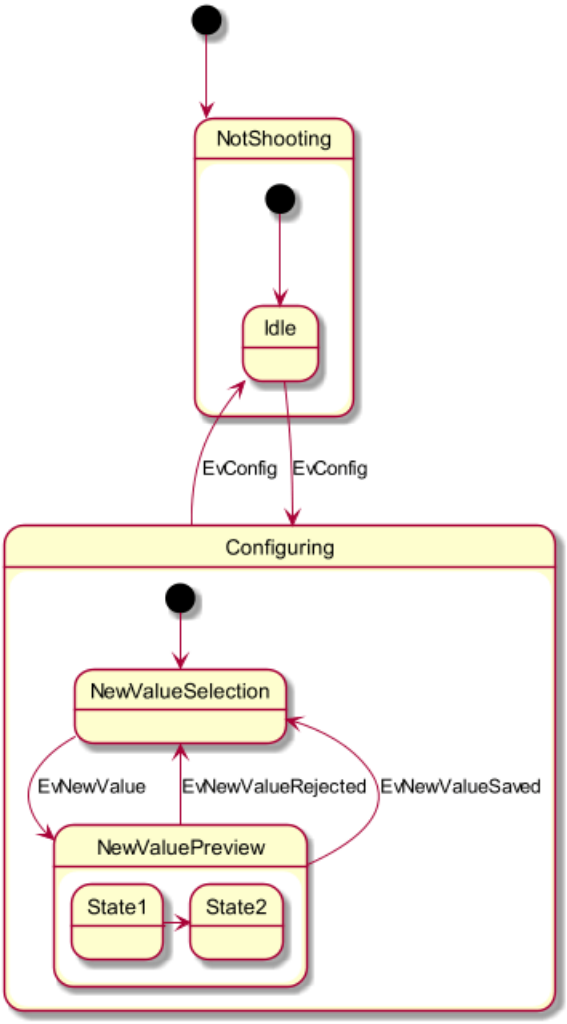


9.3 État composite

Un état peut également être composite. Vous devez alors le définir avec le mot-clé `state` et des accolades.

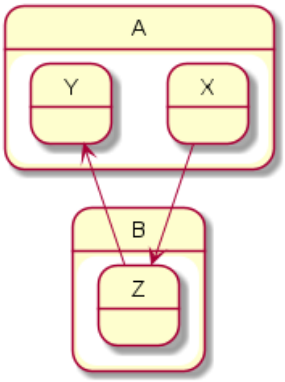
9.3.1 Sous-état interne

```
@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
```



9.3.2 Lien entre sous-états

```
@startuml
state A {
  state X {
  }
  state Y {
  }
}

state B {
  state Z {
  }
}

X --> Z
Z --> Y
@enduml
```

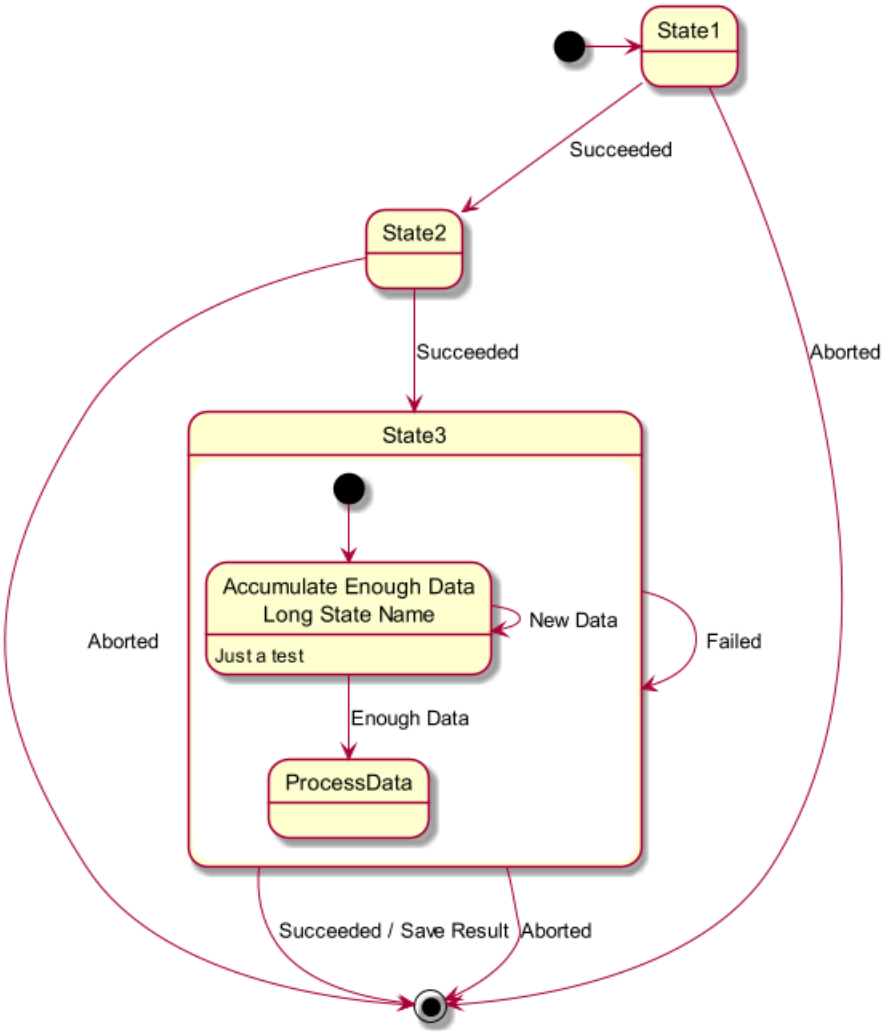


[Ref. QA-3300]

9.4 Nom long

Vous pouvez aussi utiliser le mot-clé `state` pour donner un nom avec des espaces à un état.

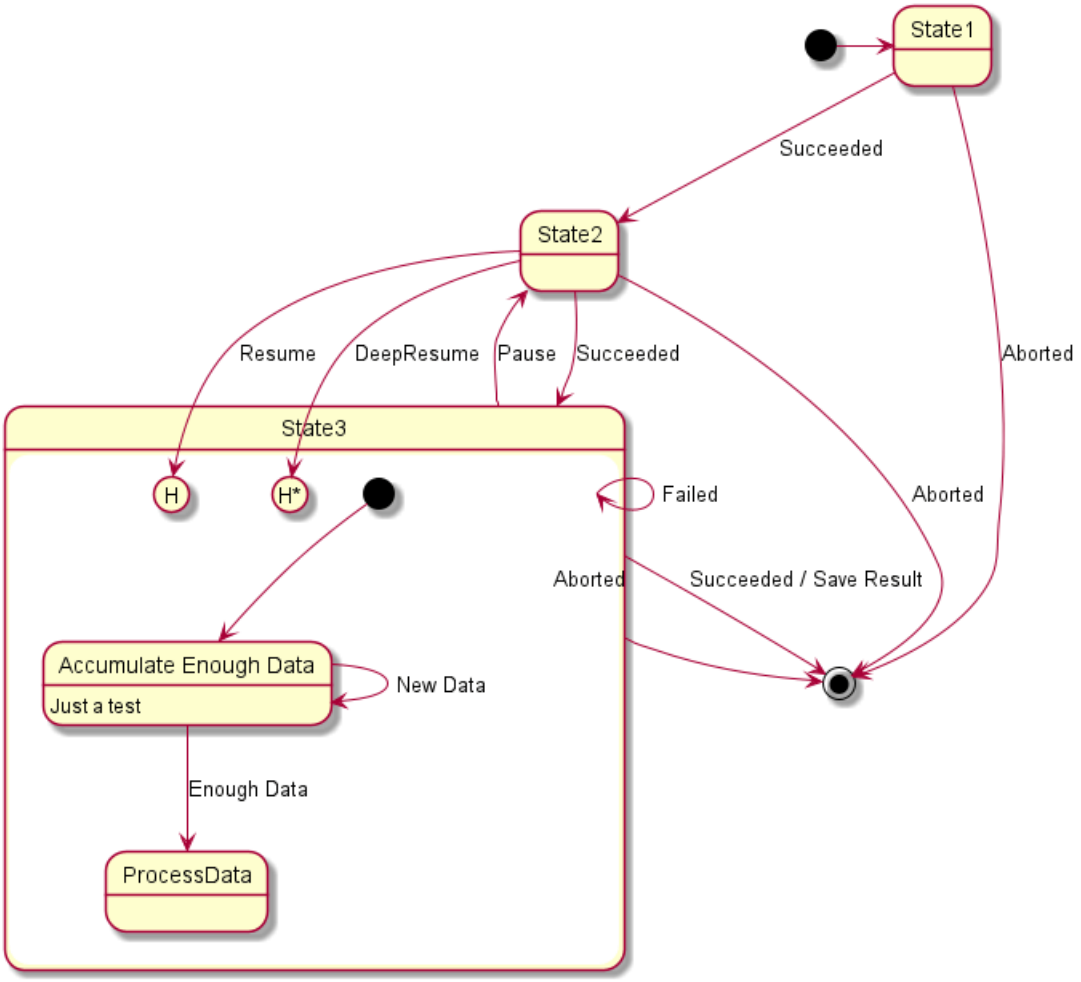
```
@startuml
scale 600 width

[*] -> 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 Historique de sous-état [[H], [H*]]

Vous pouvez utiliser `[H]` pour l'historique et `[H*]` pour l'historique profond d'un sous-état.

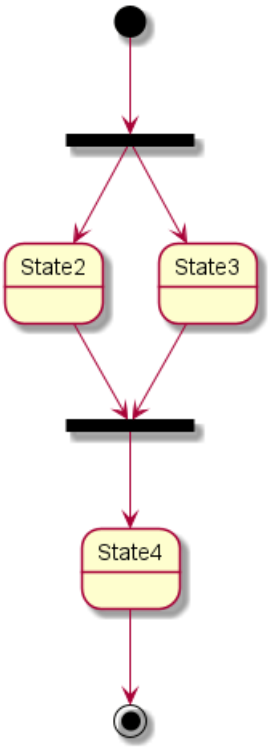
```
@startuml
[*] -> State1
State1 --> State2 : Succeeded
State1 --> [*] : Aborted
State2 --> State3 : Succeeded
State2 --> [*] : Aborted
state State3 {
  state "Accumulate Enough Data" as long1
  long1 : Just a test
  [*] --> long1
  long1 --> long1 : New Data
  long1 --> ProcessData : Enough Data
  State2 --> [H]: Resume
}
State3 --> State2 : Pause
State2 --> State3[H*]: DeepResume
State3 --> State3 : Failed
State3 --> [*] : Succeeded / Save Result
State3 --> [*] : Aborted
@enduml
```



9.6 États parallèles [fork, join]

Il est possible d'afficher des états parallèles grâce aux stéréotypes `<<fork>>` et `<<join>>`.

```
@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
```

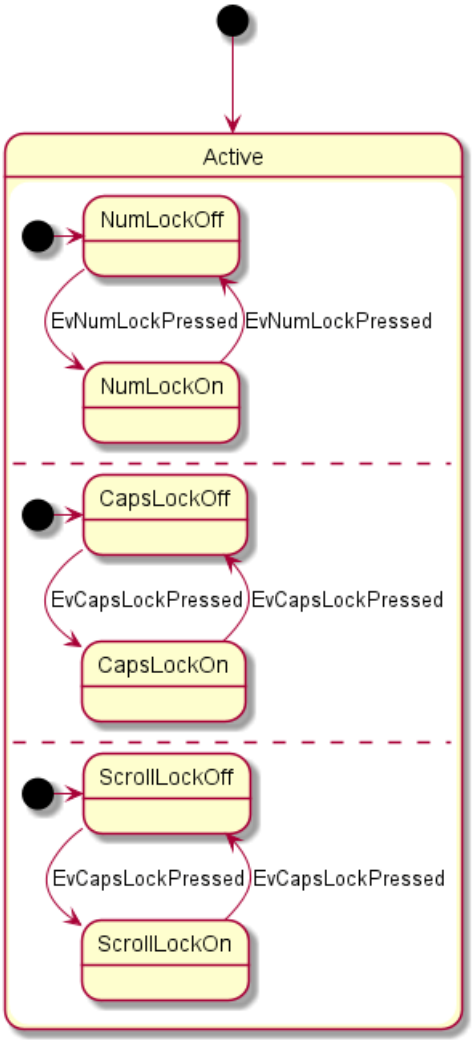


9.7 États concurrents [–, ||]

Vous pouvez définir un état concurrent dans un état composé en utilisant le symbole `--` ou `||` comme séparateur.

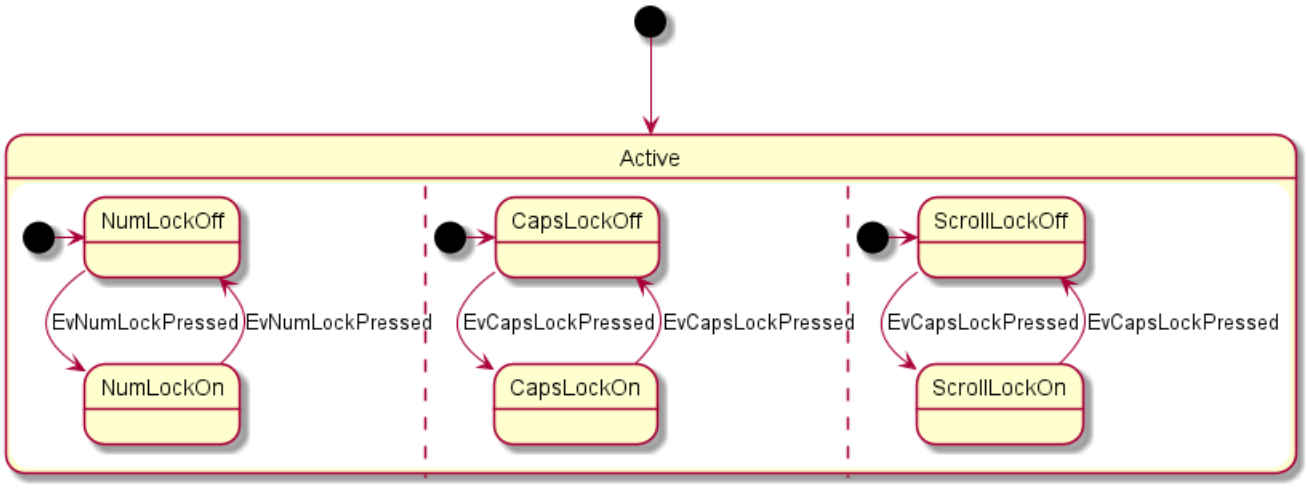
9.7.1 Séparateur horizontal --

```
@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 Séparateur vertical ||

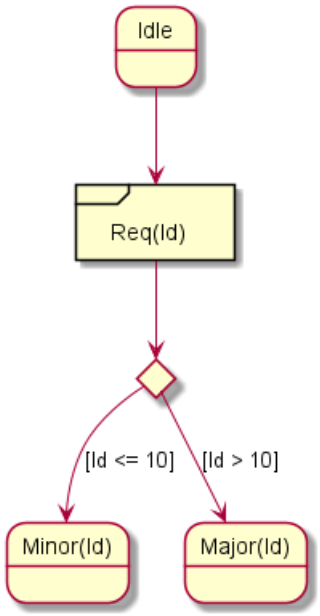
```
@startuml
[*] --> Active

state Active {
  [*] -> NumLockOff
  NumLockOff --> NumLockOn : EvNumLockPressed
  NumLockOn --> NumLockOff : EvNumLockPressed
  ||
  [*] -> CapsLockOff
  CapsLockOff --> CapsLockOn : EvCapsLockPressed
  CapsLockOn --> CapsLockOff : EvCapsLockPressed
  ||
  [*] -> ScrollLockOff
  ScrollLockOff --> ScrollLockOn : EvCapsLockPressed
  ScrollLockOn --> ScrollLockOff : EvCapsLockPressed
}

@enduml
```



9.8 Conditionnel [choice]

Le stéréotype `<<choice>>` peut être utilisé pour signifier des états conditionnels.

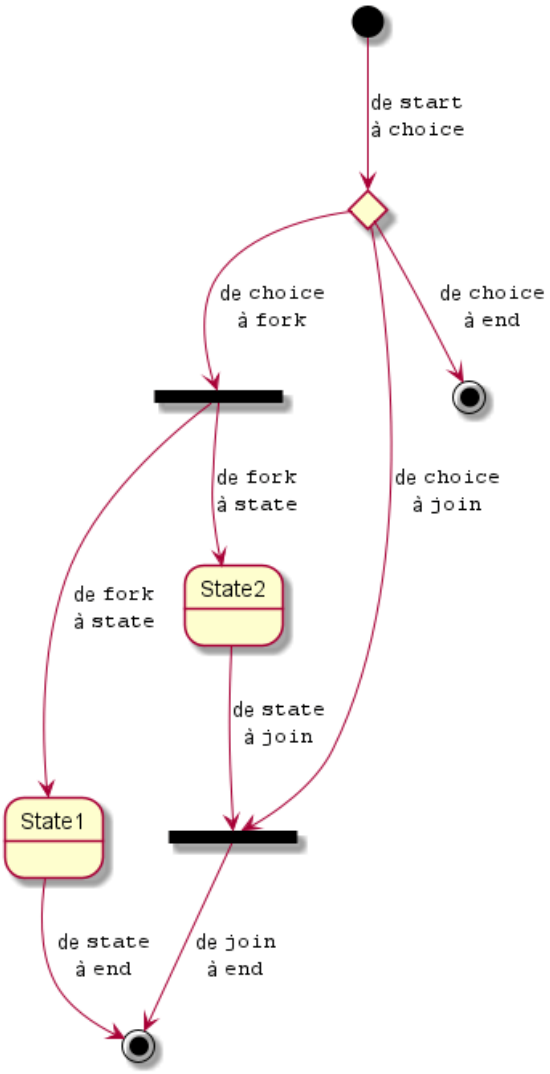
```
@startuml
state "Req(Id)" as ReqId <<sdlreceive>>
state "Minor(Id)" as MinorId
state "Major(Id)" as MajorId

state c <<choice>>

Idle --> ReqId
ReqId --> c
c --> MinorId : [Id <= 10]
c --> MajorId : [Id > 10]
@enduml
```



9.9 Exemple avec tous les stéréotypes [choice, fork, join, end]

```
@startuml
state choice1 <<choice>>
state fork1   <<fork>>
```

```
state join2   <<join>>
state end3    <<end>>

[*]     --> choice1 : de ""start""\nà ""choice""
choice1 --> fork1   : de ""choice""\nà ""fork""
choice1 --> join2   : de ""choice""\nà ""join""
choice1 --> end3    : de ""choice""\nà ""end""

fork1   ---> State1 : de ""fork""\nà ""state""
fork1   --> State2  : de ""fork""\nà ""state""

State2  --> join2   : de ""state""\nà ""join""
State1  --> [*]     : de ""state""\nà ""end""

join2   --> [*]     : de ""join""\nà ""end""
@enduml
```



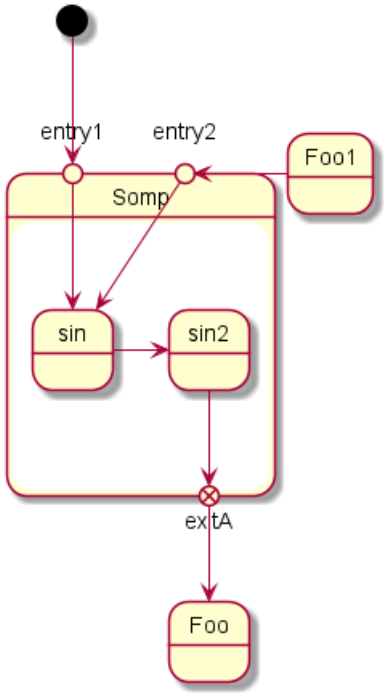
[Réf. QA-404 et QA-1159]

9.10 Petits cercles [entryPoint, exitPoint]

Vous pouvez ajouter de petits cercles *[point]* avec les stéréotypes `<<entryPoint>>` et `<<exitPoint>>` :

```
@startuml
state Somp {
```

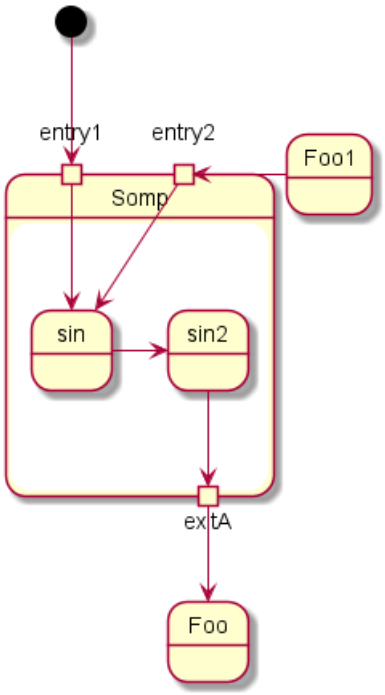
```
  state entry1 <<entryPoint>>
  state entry2 <<entryPoint>>
  state sin
  entry1 --> sin
  entry2 -> sin
  sin -> sin2
  sin2 --> exitA <<exitPoint>>
}

[*] --> entry1
exitA --> Foo
Foo1 -> entry2
@enduml
```



9.11 Petits carrés [inputPin, outputPin]

Vous pouvez ajouter de petits carrés *[pin]* avec les stéréotypes `<<inputPin>>` et `<<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
```

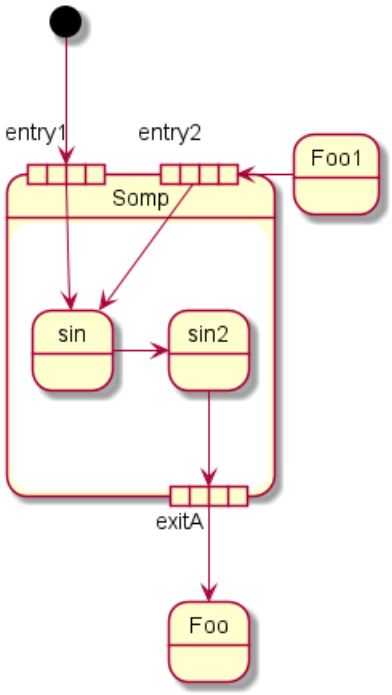


[Réf. QA-4309]

9.12 Multiples petits carrés [expansionInput, expansionOutput]

Vous pouvez ajouter de multiples petits carrés *[expansion]* avec les stéréotypes `<<expansionInput>>` et `<<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
```



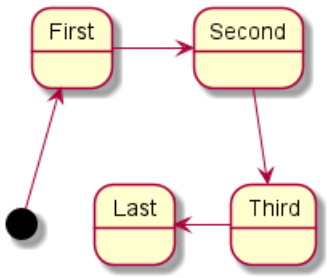
[Réf. QA-4309]

9.13 Direction des flèches

Vous pouvez utiliser `->` pour les flèches horizontales. Il est aussi possible de forcer la direction de la flèche avec la syntaxe suivante:

- `-down->` *(flèche par défaut)*
- `-right->` or `->`
- `-left->`
- `-up->`

```
@startuml

[*] -up-> First
First -right-> Second
Second --> Third
Third -left-> Last

@enduml
```



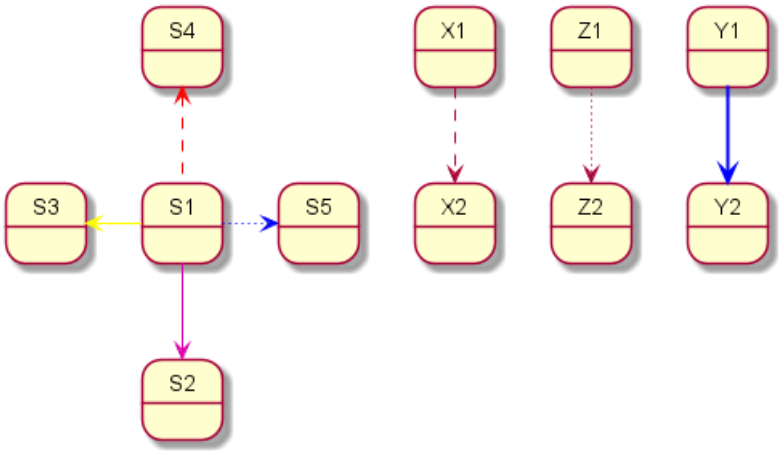
Vous pouvez aussi utiliser une notation abrégée, avec soit le premier caractère de la direction (par exemple `-d-` à la place de `-down-`) ou bien les deux premiers caractères (`-do-`).

Veuillez noter qu'il ne faut pas abuser de cette fonction : *Graphviz* donne généralement de bons résultats sans peaufinage.

9.14 Changer la couleur ou le style des flèches

Vous pouvez modifier la couleur et/ou le style des flèches.

```
@startuml
State S1
State S2
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
```



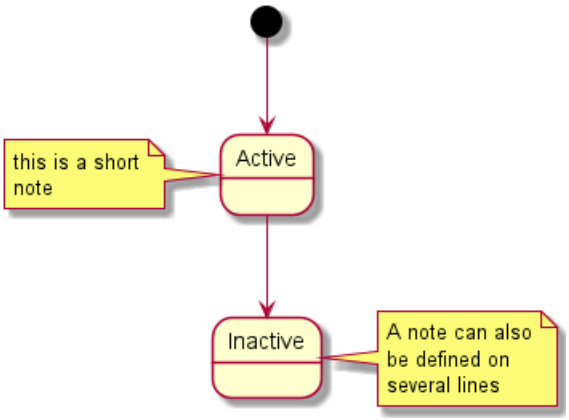
[Réf. Incubation: Change line color in state diagrams]

9.15 Note

Vous pouvez définir des notes avec les mots clés suivant: `note left of`, `note right of`, `note top of`, `note bottom of`

Vous pouvez aussi définir des notes sur plusieurs lignes.

```
@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
```



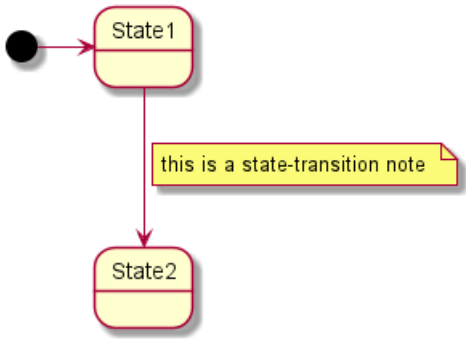
Vous pouvez aussi avoir des notes flottantes.

```
@startuml

state foo
note "This is a floating note" as N1

@enduml
```

9.16 Note sur un lien

Vous pouvez ajouter une note sur un lien entre états avec le mot clé `note on link`.

```
@startuml
[*] -> State1

State1 --> State2
note on link
  this is a state-transition note
end note
@enduml
```

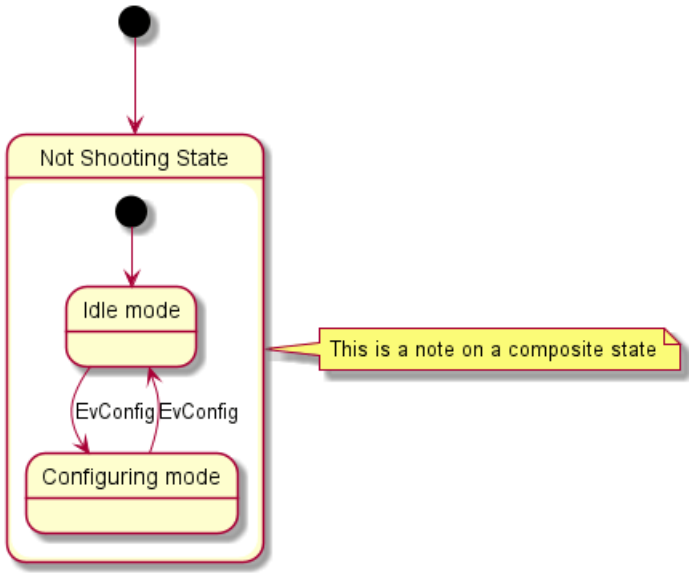


9.17 Plus de notes

Vous pouvez mettre des notes sur les états de composite

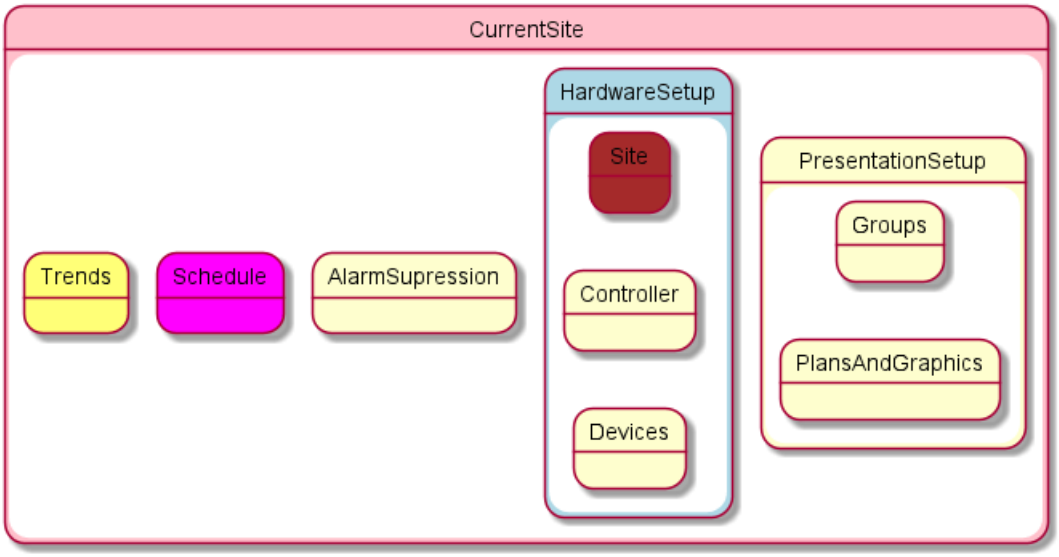
```
@startuml

[*] --> 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 Changer les couleurs localement [Inline color]

```
@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
```



[Réf. QA-1812]

9.19 Skinparam

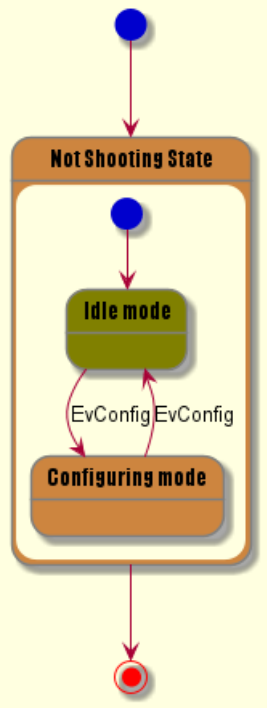
Utilisez la commande skinparam pour changer la couleur et la mise en forme du texte du schéma.

Vous pouvez utiliser cette commande :

- Dans la définition du diagramme, comme pour les autres commandes,
- Dans un fichier inclus,
- Dans un fichier de configuration, renseigné dans la ligne de commande ou la tâche ANT.

Vous pouvez définir une couleur spécifique et une police d'écriture pour les états stéréotypés.

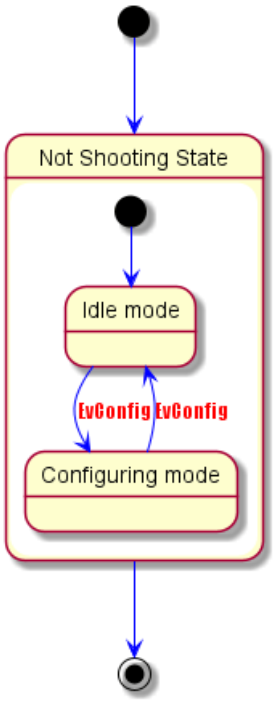
```
@startuml
skinparam backgroundColor LightYellow
skinparam state {
  StartColor MediumBlue
  EndColor Red
  BackgroundColor Peru
  BackgroundColor<<Warning>> Olive
  BorderColor Gray
  FontName Impact
}

[*] --> 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
```



9.20 Changing style

You can change style.

```
@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
```



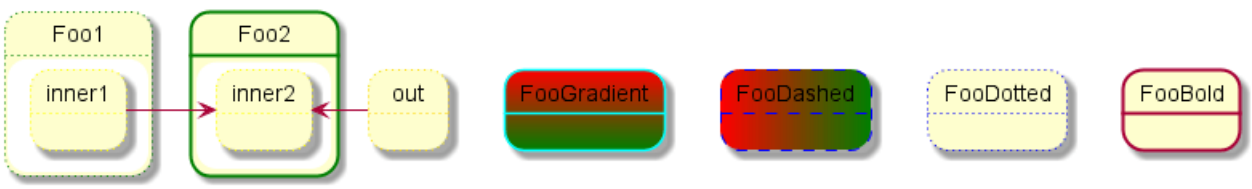
9.21 Change state color and style (inline style)

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
```

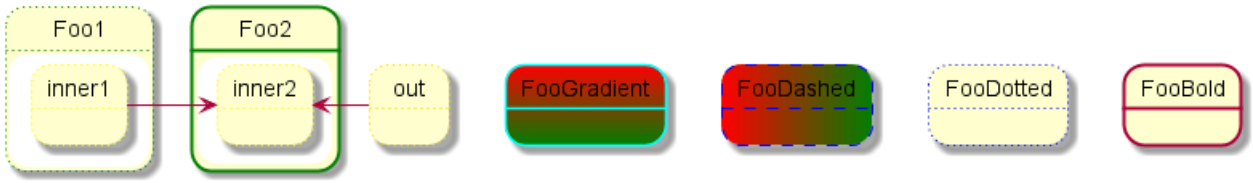


[Ref. QA-1487]

- `#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
```




```
@startuml
state s1 : s1 description
state s2 #pink;line:red;line.bold;text:red : s2 description
state s3 #palegreen;line:green;line.dashed;text:green : s3 description
state s4 #aliceblue;line:blue;line.dotted;text:blue   : s4 description
@enduml
```

[Adapted from QA-3770]

10 Diagramme de temps

C'est encore en développement. Vous pouvez proposer de nouvelles fonctionnalités si vous en avez besoin.

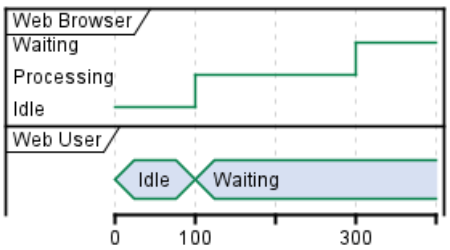
10.1 Définitions des participants

Les participants sont déclarés à l'aide des mots-clé `consise` ou `robust`, en fonction de la façon dont vous souhaitez les dessiner.

- `concise`: A simplified signal designed to show the movement of data (great for messages).
- `robust`: A complex line signal designed to show the transition from one state to another (can have many states).
- `clock`: A 'clocked' signal that repeatedly transitions from high to low
- `binary`: A specific signal restricted to only 2 states (binary).

Les changements d'état sont notifiés avec la notation `@` et le verbe `is`.

```
@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
```



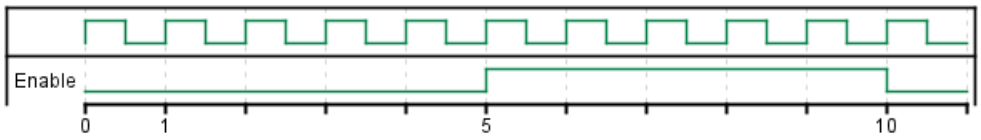
10.2 Horloge et signaux binaires

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
```

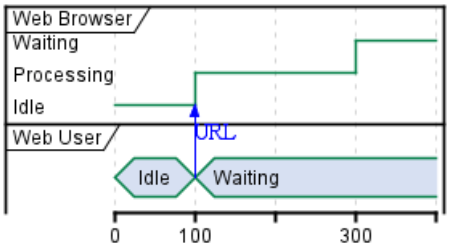
```
@10
EN is low
@enduml
```



10.3 Ajout de messages

Vous pouvez rajouter des messages à l'aide de la syntaxe suivante.

```
@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
```

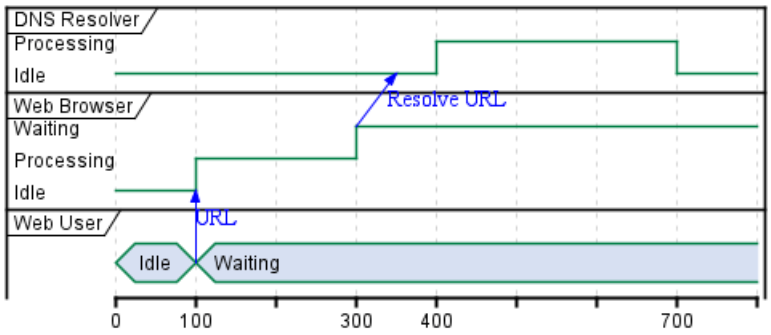


10.4 Référence relative de temps

Avec la notation `@`, il est possible d'utiliser une notation relative du temps.

```
@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
```

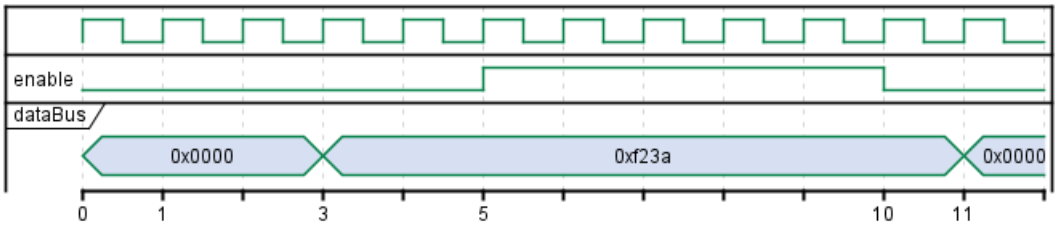


10.5 Anchor Points

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>
```

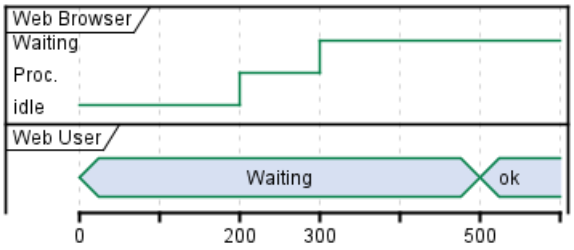
```
@startuml
clock clk with period 1
binary "enable" as EN
concise "dataBus" as db

@0 as :start
@5 as :en_high
@10 as :en_low


@:start
EN is low
db is "0x0000"

@:en_high
EN is high

@:en_low
EN is low

@:en_high-2
db is "0xf23a"

@:en_high+6
db is "0x0000"
@enduml
```



10.6 Définition participant par participant

Plutôt que de déclarer le diagramme dans l'ordre chronologique, il est possible de le définir participant par participant.

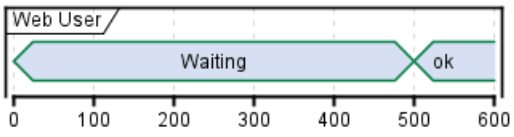
```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

@WB
0 is idle
+200 is Proc.
+100 is Waiting

@WU
0 is Waiting
+500 is ok
@enduml
```



10.7 Choix du zoom

Il est possible de choisir une échelle d'affichage précise.

```
@startuml
concise "Web User" as WU
scale 100 as 50 pixels

@WU
0 is Waiting
+500 is ok
@enduml
```

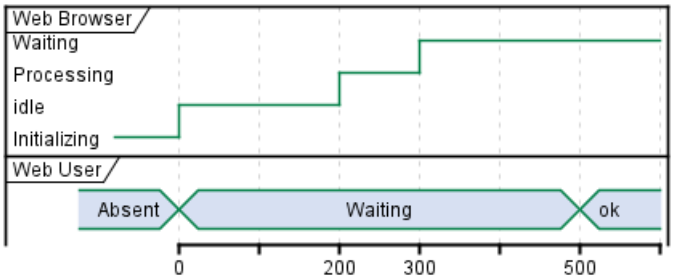


10.8 État initial

Vous pouvez également définir un état initial.

```
@startuml
```

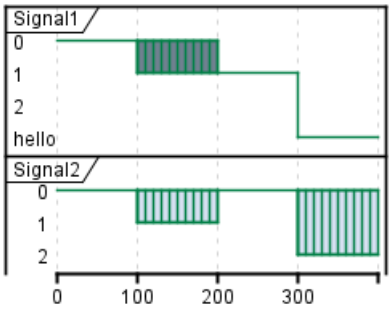
```
robust "Web Browser" as WB
concise "Web User" as WU

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.

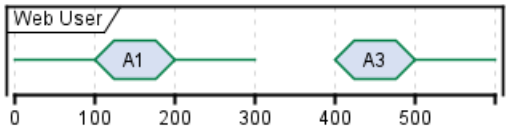
```
@startuml
robust "Signal1" as S1
robust "Signal2" as S2
S1 has 0,1,2,hello
S2 has 0,1,2
@0
S1 is 0
S2 is 0
@100
S1 is {0,1} #SlateGrey
S2 is {0,1}
@200
S1 is 1
S2 is 0
@300
S1 is hello
S2 is {0,2}
@enduml
```



10.10 Hidden state

It is also possible to hide some state.

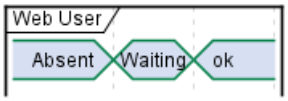
```
@startuml
concise "Web User" as WU

@0
WU is {-}

@100
WU is A1

@200
WU is {-}

@300
WU is {hidden}

@400
WU is A3

@500
WU is {-}
@enduml
```



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
```

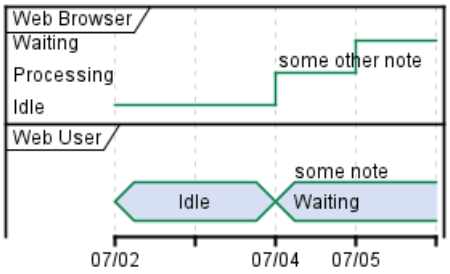


10.12 Using Time and Date

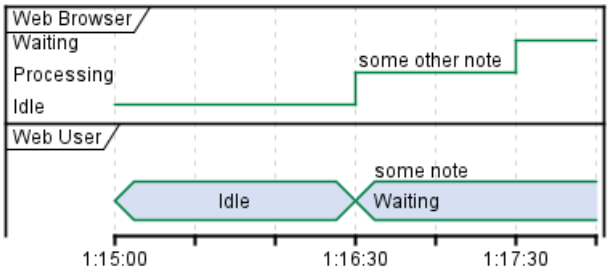
It is possible to use time or date.

```
@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
```

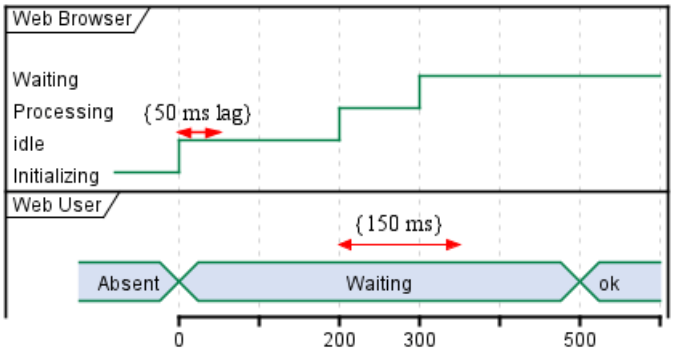


10.13 Ajout de contraintes

Il est possible d'afficher des contraintes de temps sur les diagrammes.

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

WB is Initializing
WU is Absent

@WB
0 is idle
+200 is Processing
```

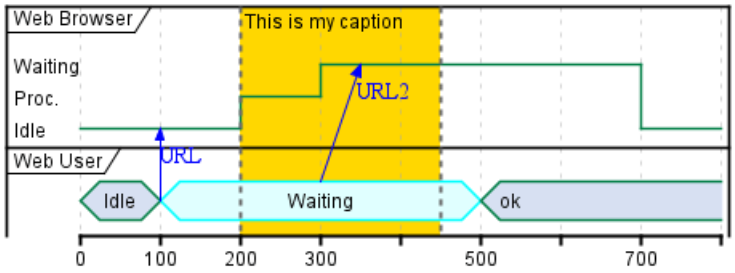
```
+100 is Waiting
WB@0 <-> @50 : {50 ms lag}

@WU
0 is Waiting
+500 is ok
@200 <-> @+150 : {150 ms}
@enduml
```



10.14 Highlighted period

You can higlight a part of diagram.

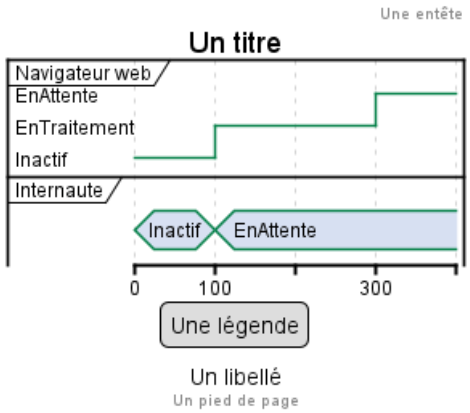
```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

@100
WU -> WB : URL
WU is Waiting #LightCyan;line:Aqua

@200
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
@enduml
```



10.15 Ajout de textes

Vous pouvez ajouter éventuellement un titre, une entête, un pied de page, une légende ou un libellé :

```
@startuml
Title Un titre
header: Une entête
footer: Un pied de page
legend
Une légende
end legend
caption Un libellé

robust "Navigateur web" as WB
concise "Internaute" as WU

@0
WU is Inactif
WB is Inactif

@100
WU is EnAttente
WB is EnTraitement

@300
WB is EnAttente
@enduml
```

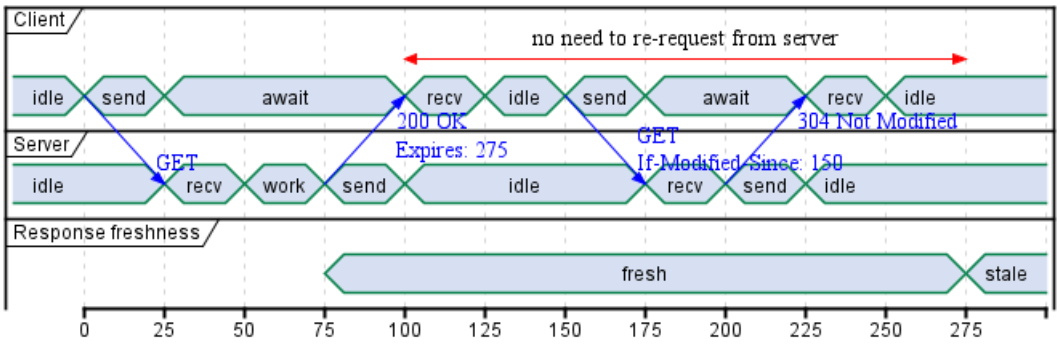


10.16 Complete example

Thanks to Adam Rosien for this example.

```
@startuml
concise "Client" as Client
concise "Server" as Server
```

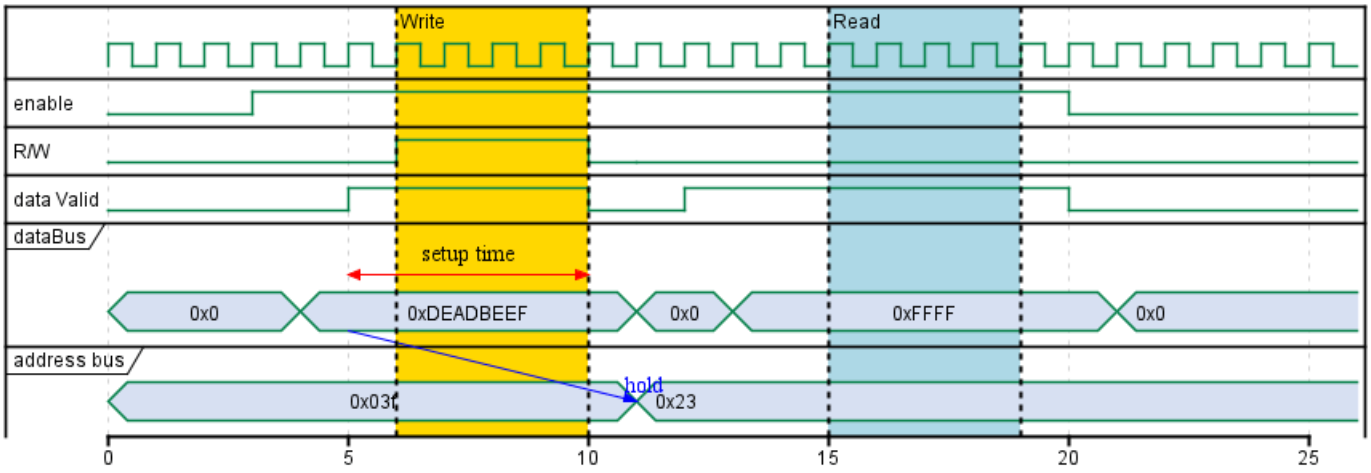
```
concise "Response freshness" as Cache

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
```



10.17 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
```

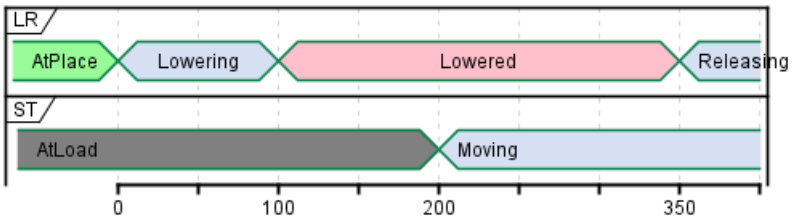
```
@6 as :write_beg
@10 as :write_end

@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
```



10.18 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]

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:

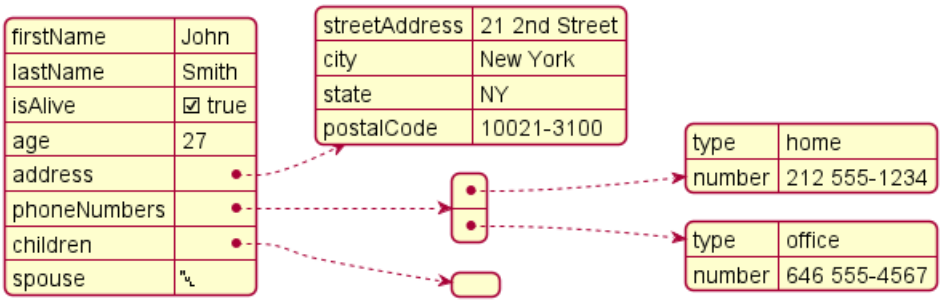
- begin with `@startjson` keyword
- end with `@endjson` keyword.

```
@startjson
{
   "fruit":"Apple",
   "size":"Large",
   "color":"Red"
}
@endjson
```

11.1 Complex example

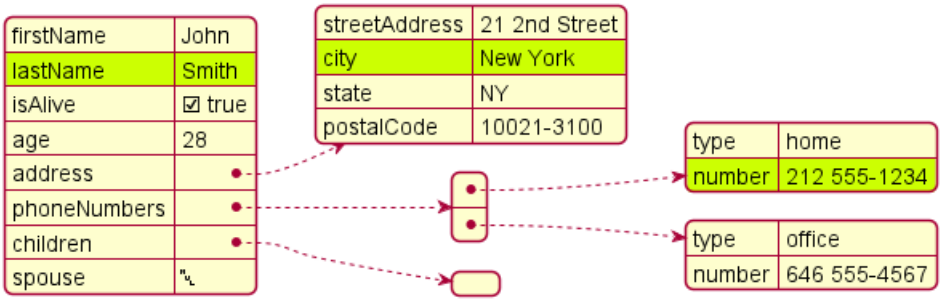
You can use complex JSON structure.

```
@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
```

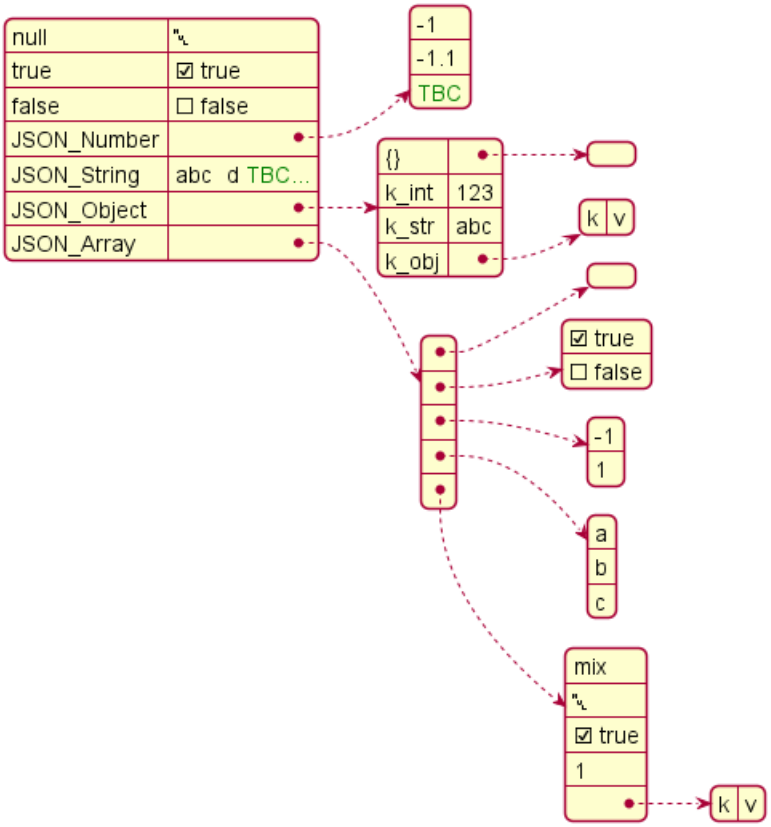


11.3 JSON basic element

11.3.1 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
```

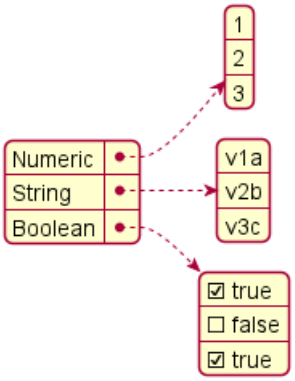


11.4 JSON array or table

11.4.1 Array type

```
@startjson
{
"Numeric": [1, 2, 3],
"String ": ["v1a", "v2b", "v3c"],
"Boolean": [true, false, true]
}
```

```
@endjson
```



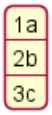
11.4.2 Minimal array or table

11.4.3 Number array

```
@startjson
[1, 2, 3]
@endjson
```

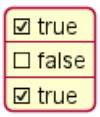
11.4.4 String array

```
@startjson
["1a", "2b", "3c"]
@endjson
```



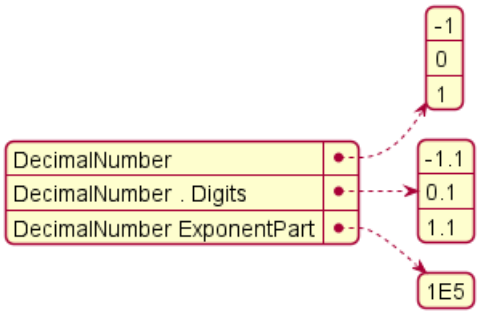
11.4.5 Boolean array

```
@startjson
[true, false, true]
@endjson
```



11.5 JSON numbers

```
@startjson
{
"DecimalNumber": [-1, 0, 1],
"DecimalNumber . Digits": [-1.1, 0.1, 1.1],
"DecimalNumber ExponentPart": [1E5]
}
@endjson
```



11.6 JSON strings

11.6.1 JSON Unicode

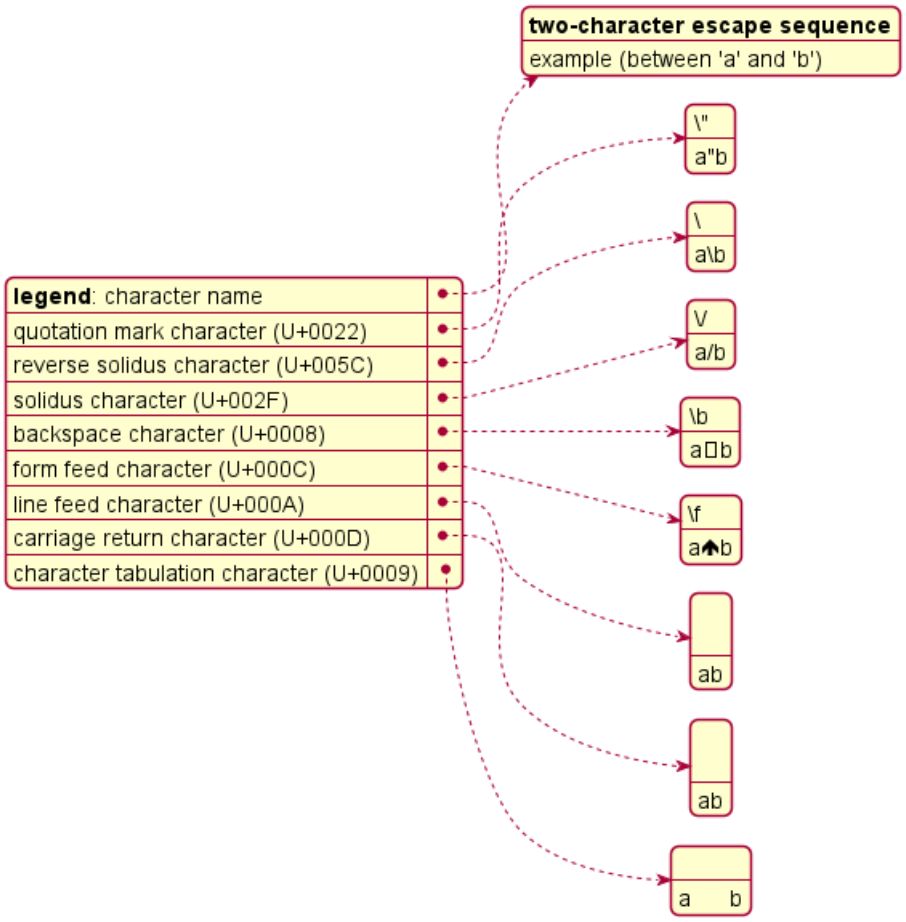
On JSON you can use Unicode directly or by using escaped form like .

```
@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	😐
😐	😐

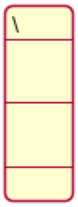
11.6.2 JSON two-character escape sequence

```
@startjson
{
 "**legend**: character name":                ["**two-character escape sequence**", "example (between
 "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
```



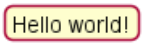
TODO: FIXME FIXME or not , on the same item as management in PlantUML **TODO:** FIXME

```
@startjson
[
"\\\\",
"\\n",
"\\r",
"\\t"
]
@endjson
```



11.7 Minimal JSON examples

```
@startjson
"Hello world!"
@endjson
```




```
@startjson
```

```
42
@endjson
```

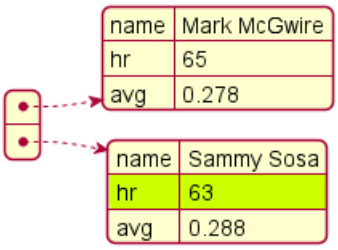

```
@startjson
true
@endjson
```

(Examples come from STD 90 - Examples)

11.8 Using (global) style

11.8.1 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
```

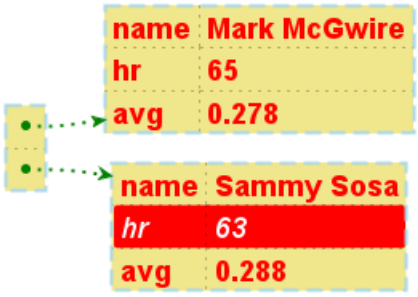


11.8.2 With style

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]

12 Display YAML Data

YAML format is widely used in software.

You can use PlantUML to visualize your data.

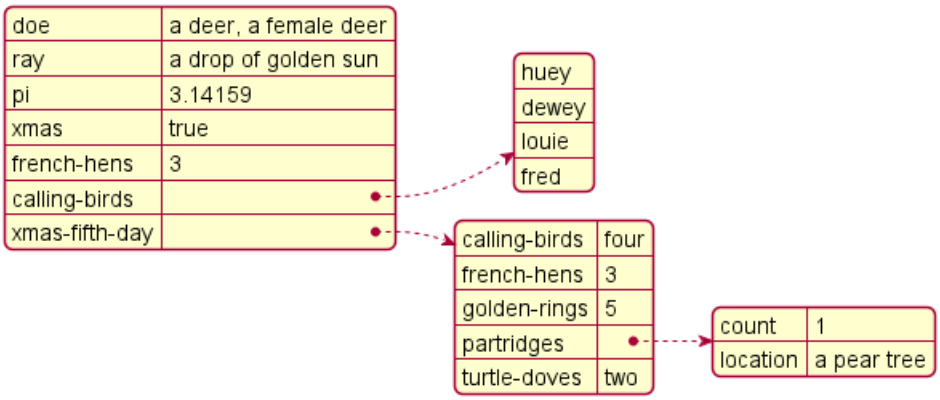
To activate this feature, the diagram must:

- begin with `@startyaml` keyword
- end with `@endyaml` keyword.

```
@startyaml
fruit: Apple
size: Large
color: Red
@endyaml
```

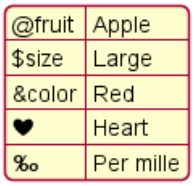
12.1 Complex example

```
@startyaml
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
```



12.2 Specific key (with symbols or unicode)

```
@startyaml
@fruit: Apple
$size: Large
&color: Red
 : Heart
%: Per mille
@endyaml
```

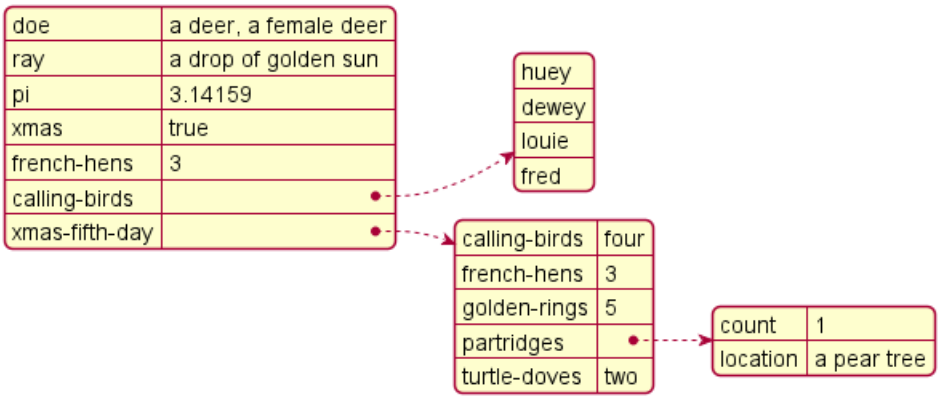


[Ref. QA-13376]

12.3 Highlight parts

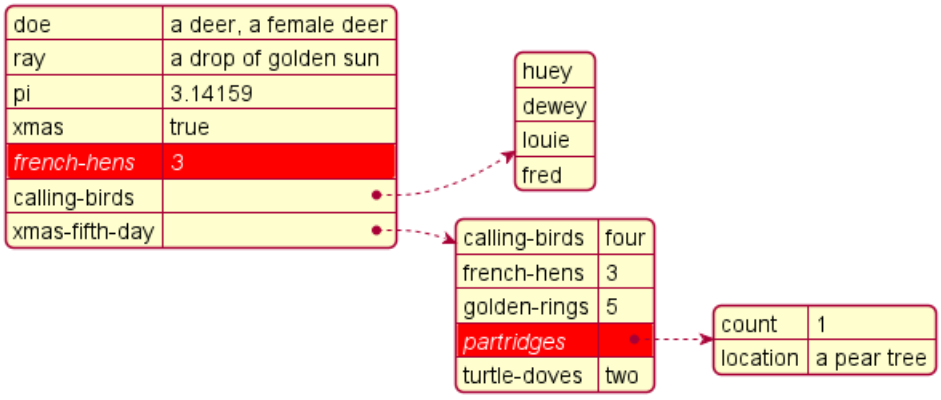
12.3.1 Normal style

```
@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
```



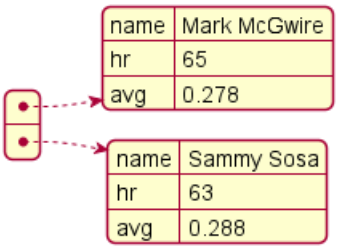
[Ref. QA-13288]

12.4 Using (global) style

12.4.1 Without style *(by default)*

```
@startyaml
  -
    name: Mark McGwire
    hr:   65
    avg:  0.278
```

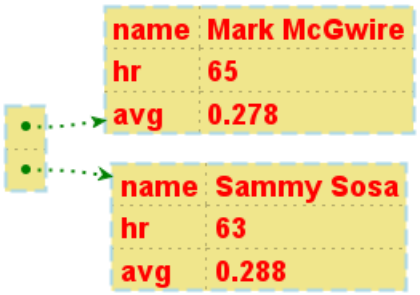
```
  -
    name: Sammy Sosa
    hr:   63
    avg:  0.288
@endyaml
```



12.4.2 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
```


name
Mark McGwire
hr
65
avg
0.278
name
Sammy Sosa
hr
63
avg
0.288

[Ref. QA-13123]

13 nwdiag

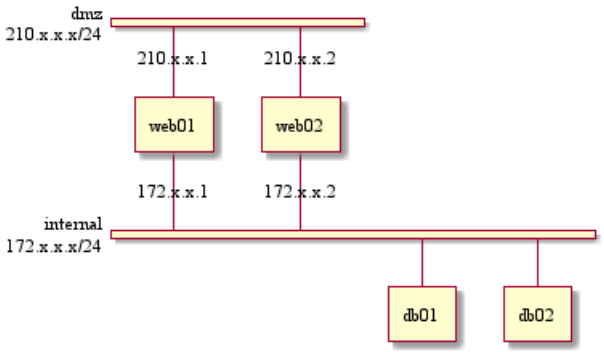
nwdiag a été écrit par Takeshi Komiya. Ce programme permet de décrire rapidement des diagrammes de réseaux. Merci beaucoup à lui pour sa création!

Puisque la syntaxe est simple et clair, celle-ci a été intégrée à PlantUML.

Nous réutilisons ici les exemples que Takeshi a utilisés.

13.1 Exemple simple

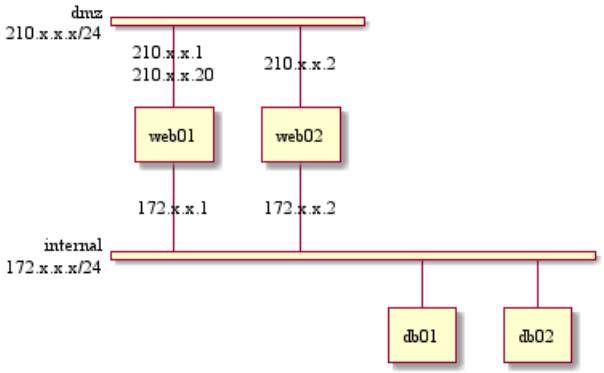
```
@startuml
nwdiag {
  network dmz {
      address = "210.x.x.x/24"

      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"];
```

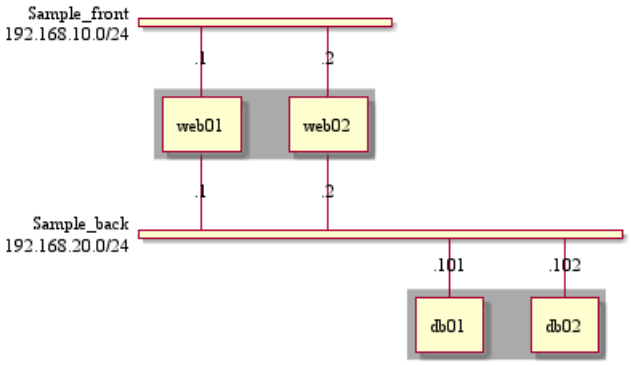
```
      db01;
      db02;
  }
}
@enduml
```



13.3 Grouping nodes

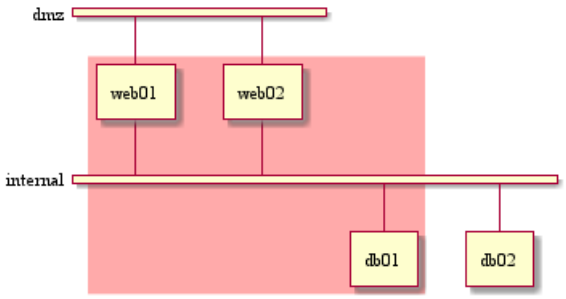
13.3.1 Define group inside network definitions

```
@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
```

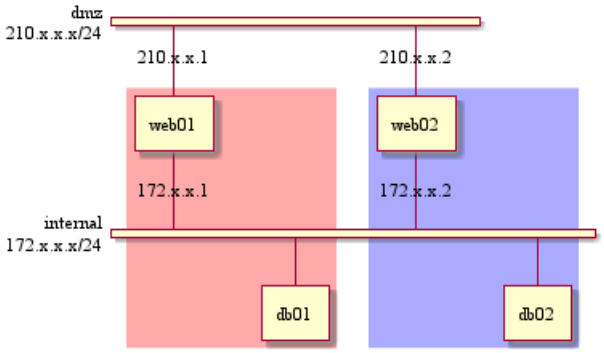


13.3.3 Define several groups on same network

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
```

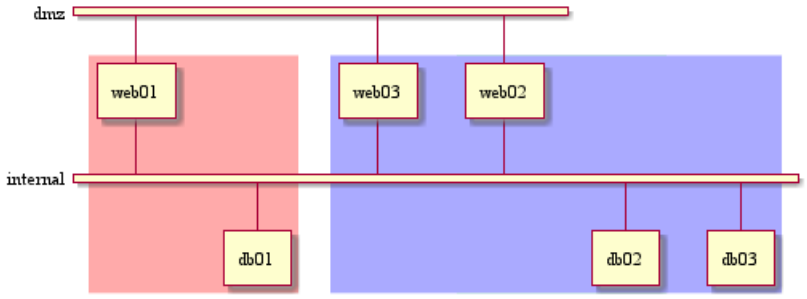


[Ref. QA-12663]

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)

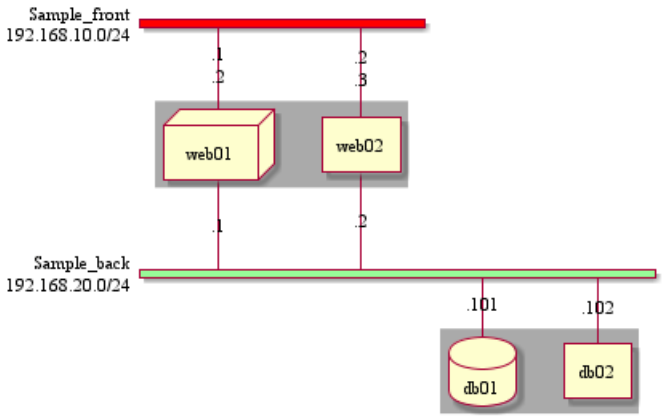
13.4.1 Network

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
```

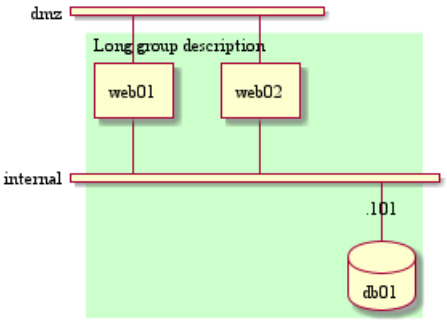


13.4.2 Group

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
```



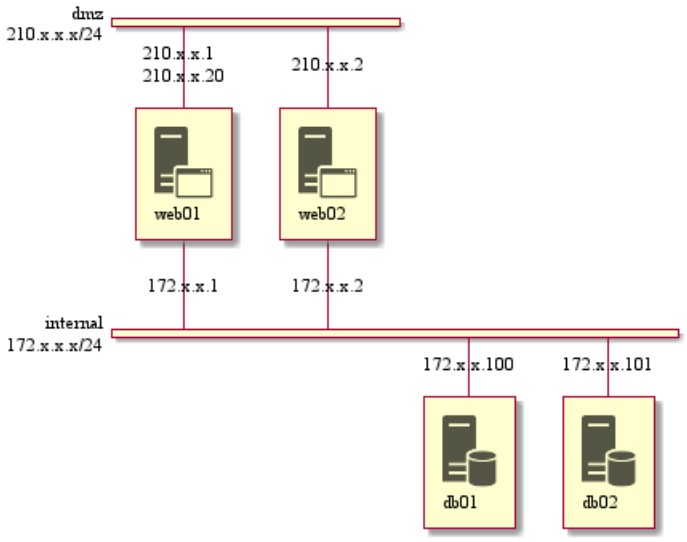
[Ref. QA-12056]

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, \n 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
```



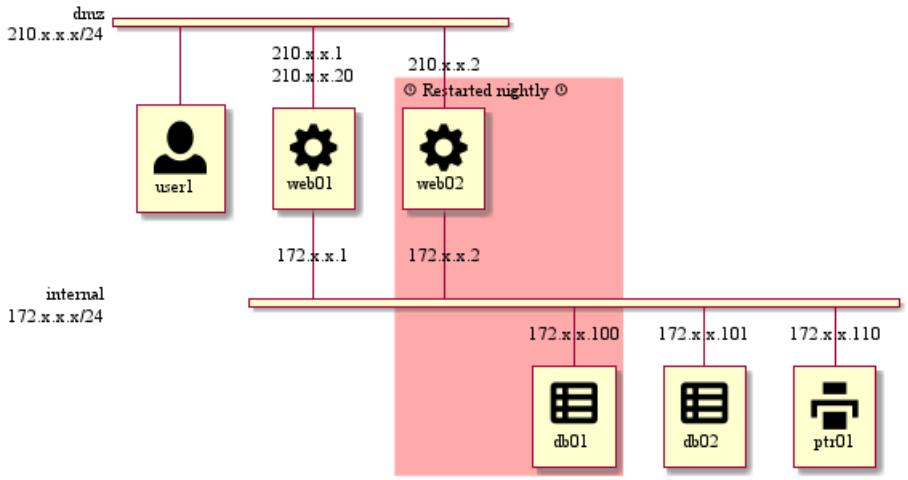
[Ref. QA-11862]

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 to make a newline:

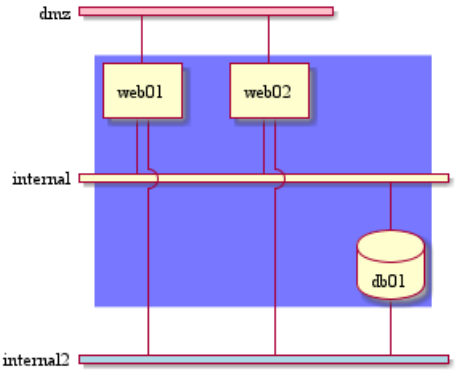
```
@startuml

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.

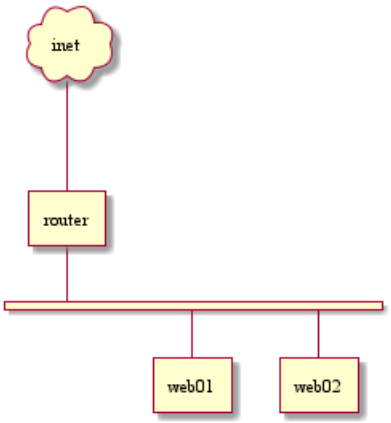
```
@startuml
nwdiag {
  // define group at outside network definitions
  group {
    color = "#7777FF";

    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
```

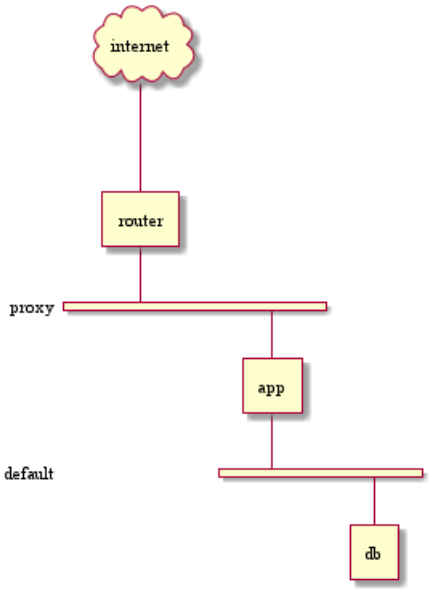


13.9 Peer networks and group

13.9.1 Without group

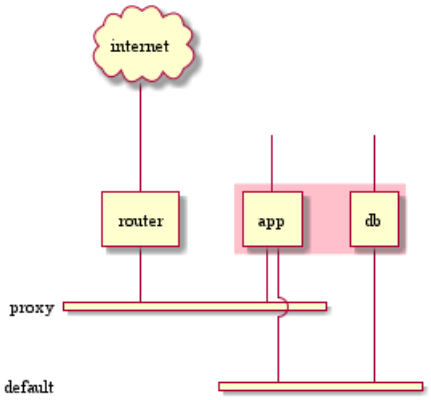
```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;
```

```
    network proxy {
        router;
        app;
    }
    network default {
     app;
        db;
    }
}
@enduml
```



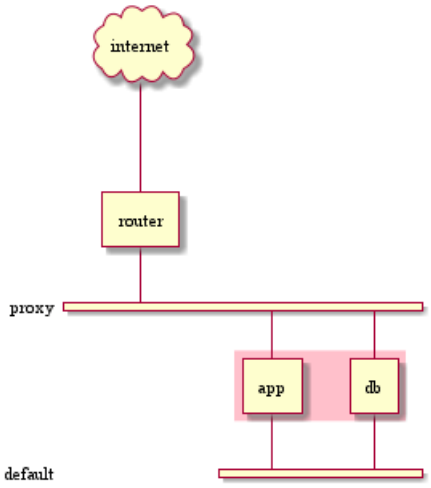
13.9.2 Group on first

```
@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

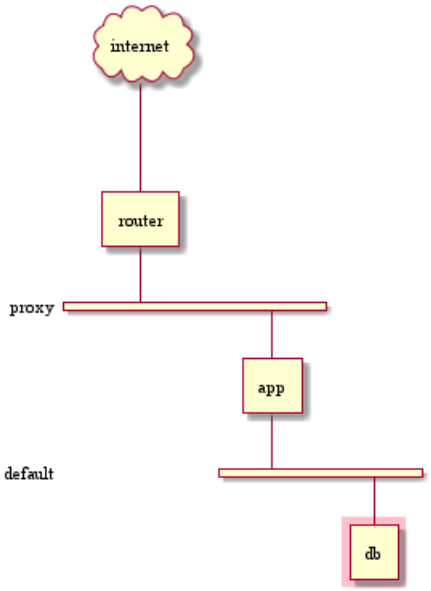
```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    network proxy {
        router;
        app;
    }

    group {
      color = "pink";
      app;
      db;
    }

    network default {
     app;
        db;
    }
}
@enduml
```



TODO: FIXME Why the line on proxy for 'db'? ('db' must be only on 'default network') [See example without group]

13.9.4 Group on third

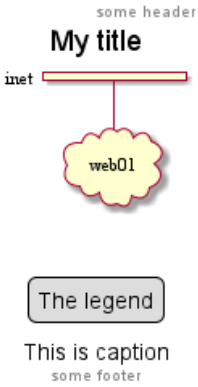
```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    network proxy {
        router;
        app;
    }
    network default {
     app;
        db;
    }
    group {
      color = "pink";
      app;
      db;
    }
}
@enduml
```



TODO: FIXME *[Ref. Issue#408 and QA-12655]* **TODO:** Not totally fixed

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]

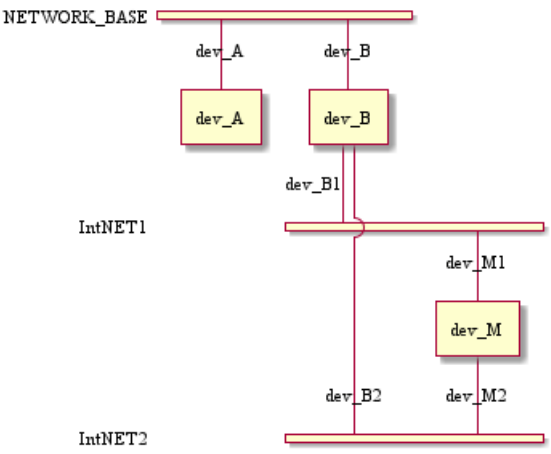
13.11 Change width of the networks

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

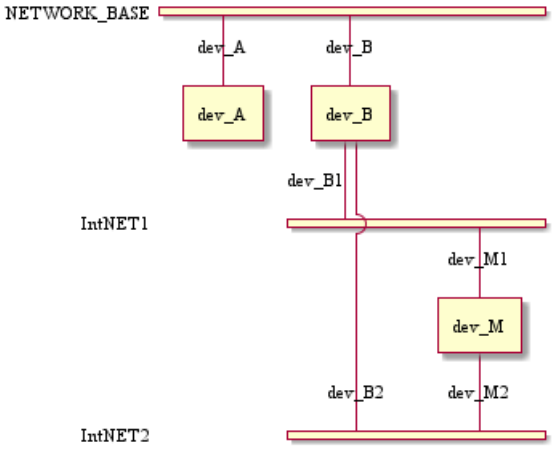
```
@startuml
nwdiag {
  network NETWORK_BASE {
   dev_A   [address = "dev_A" ]
   dev_B [address = "dev_B" ]
  }
  network IntNET1 {
   dev_B [address = "dev_B1" ]
   dev_M [address = "dev_M1" ]
  }
  network IntNET2 {
   dev_B [address = "dev_B2" ]
   dev_M [address = "dev_M2" ]
 }
}
@enduml
```



- only the first

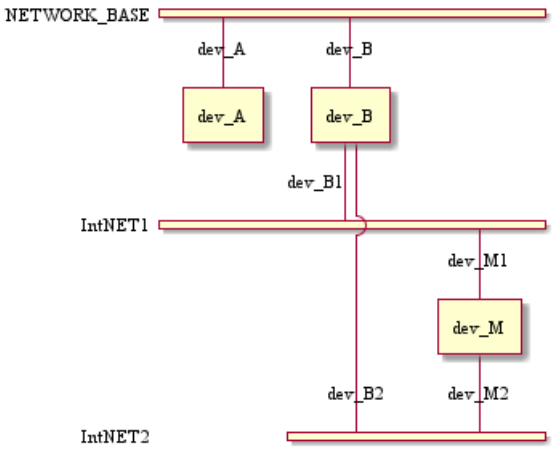
```
@startuml
```

```
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
```



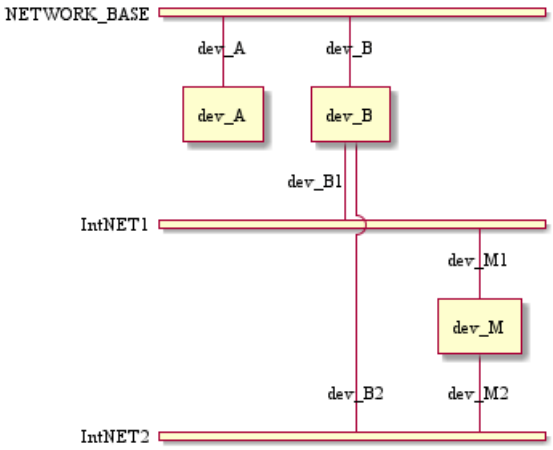
- the first and the second

```
@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
```

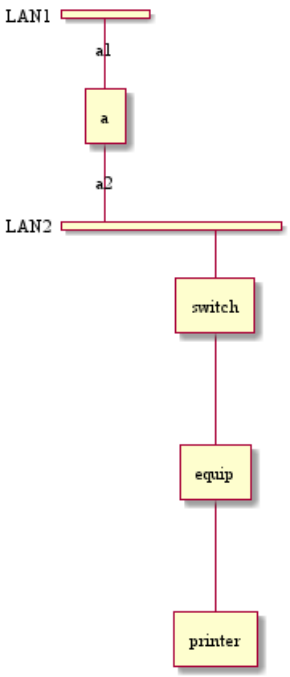


13.12 Other internal networks

You can define other internal networks (TCP/IP, USB, SERIAL,...).

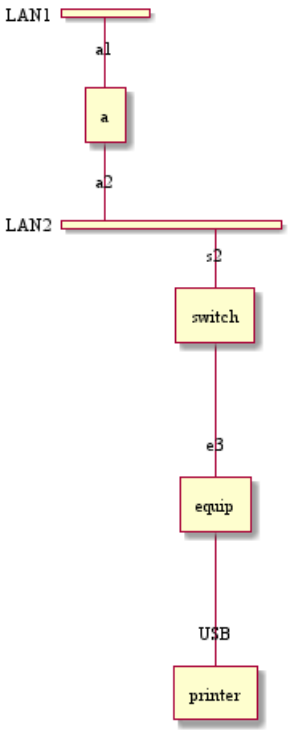
- Without adress or type

```
@startuml
nwdiag {
  network LAN1 {
     a [address = "a1"];
  }
  network LAN2 {
     a [address = "a2"];
     switch;
  }
  switch -- equip;
  equip -- printer;
}
@enduml
```



- With adress 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]

14 Salt (Wireframe)

Salt est un sous projet inclus dans PlantUML qui peut vous aider à modeler une interface graphique ou *une maquette fonctionnelle* ou un *Wireframe* (en anglais).

Le but de cet outil est de pouvoir contruire, échanger et discuter facilement sur des échantillons de fenêtres simples.

Vous pouvez utiliser soit le mot clé `@startsalt`, ou bien `@startuml` suivi par une ligne avec le mot clé `salt`.

14.1 Composants de base

Une fenêtre doit commencer et finir par une accolade.

Vous pouvez ensuite définir :

- un bouton en utilisant `[` et `]`,
- un bouton radio en utilisant `(` et `)`,
- une case à cocher en utilisant `[` et `]`,
- une zone de texte utilisateur en utilisant `"`,
- une liste déroulante en utilisant `^`.

```
@startsalt
{
  Just plain text
  [This is my button]
  ()  Unchecked radio
  (X) Checked radio
  []  Unchecked box
  [X] Checked box
  "Enter text here   "
  ^This is a droplist^
}
@endsalt
```

14.2 Utilisation de grille

Un tableau est créé automatiquement en utilisant une accolade ouvrante `{`.

Il faut utiliser `|` pour séparer les colomnes.

Par exemple:

```
@startsalt
{
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```

Tout de suite après l'accolade ouvrante, vous pouvez utiliser un caractère pour définir si vous voulez dessiner les lignes ou les colonnes de la grille :

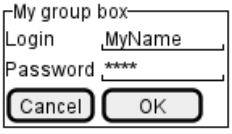
Symbol	Result
#	Pour afficher toutes les lignes verticales et horizontales
!	Pour afficher toutes les lignes verticales
-	Pour afficher toutes les lignes horizontales
+	Pour afficher les lignes extérieurs

```
@startsalt
{+
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```

14.3 Regroupement de champs

Plus d'information ici

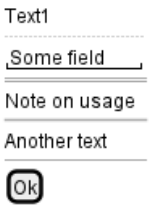
```
@startsalt
{^"My group box"
  Login    | "MyName   "
  Password | "****     "
  [Cancel] | [  OK   ]
}
@endsalt
```



14.4 Utilisation des séparateurs

Vous pouvez utiliser de nombreuses lignes horizontales en tant que séparateur.

```
@startsalt
{
  Text1
  ..
  "Some field"
  ==
  Note on usage
  ~~
  Another text
  --
  [Ok]
}
@endsalt
```



14.5 Arbre (structure arborescente) [T]

Pour faire un arbre ou une structure arborescente, vous devez commencer avec `{T` et utiliser `+` pour signaler la hiérarchie.

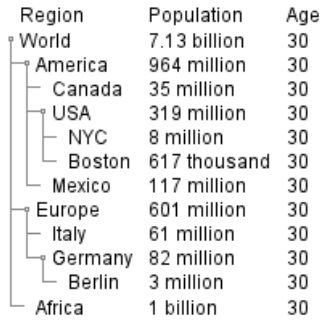
```
@startsalt
{
{T
 + World
 ++ America
 +++ Canada
 +++ USA
 ++++ New York
 ++++ Boston
 +++ Mexico
 ++ Europe
 +++ Italy
 +++ Germany
 ++++ Berlin
 ++ Africa
}
}
@endsalt
```

14.6 Tree table [T]

You can combine trees with tables.

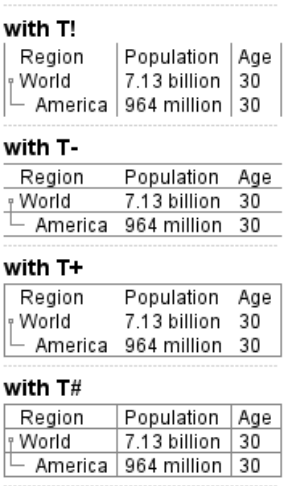
```
@startsalt
{
{T
+Region        | Population    | Age
+ World        | 7.13 billion  | 30
++ America     | 964 million   | 30
+++ Canada     | 35 million    | 30
+++ USA        | 319 million   | 30
++++ NYC       | 8 million     | 30
++++ Boston    | 617 thousand  | 30
+++ Mexico     | 117 million   | 30
++ Europe      | 601 million   | 30
+++ Italy      | 61 million    | 30
+++ Germany    | 82 million    | 30
```

```
++++ Berlin     | 3 million      | 30
++ Africa       | 1 billion      | 30
}
}
@endsalt
```



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
```

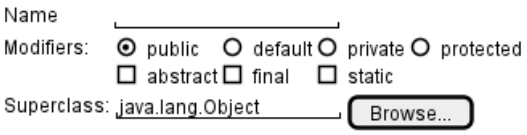


[Ref. QA-1265]

14.7 Accolades délimitantes [{, }]

Vous pouvez définir des sous-éléments en créant une accolade ouvrante.

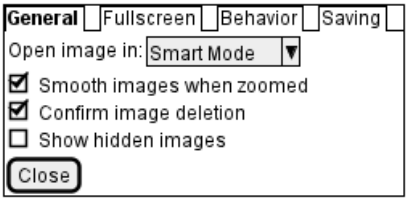
```
@startsalt
{
Name          | "                  "
Modifiers:    | { (X) public | () default | () private | () protected
                  [] abstract | [] final   | [] static }
Superclass:   | { "java.lang.Object " | [Browse...] }
}
@endsalt
```



14.8 Ajout d'onglet [/]

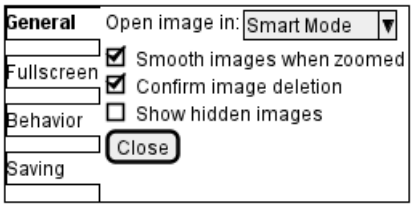
Vous pouvez ajouter des onglets avec la notation `{/`. Notez que vous pouvez utiliser du code HTML pour avoir un texte en gras.

```
@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
```



Les onglets peuvent également être orientés verticalement:

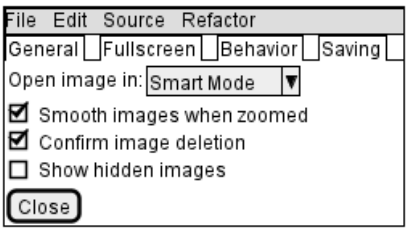
```
@startsalt
{+
{/ <b>General
Fullscreen
Behavior
Saving } |
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
[Close]
}
}
@endsalt
```



14.9 Utiliser les menus [*]

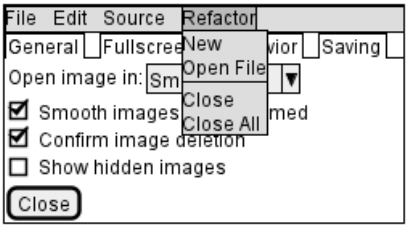
Vous pouvez ajouter un menu en utilisant la notation `{*`.

```
@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
```



Il est également possible d'ouvrir un 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
```



14.10 Tableaux avancés

Vous pouvez utiliser deux notations spéciales pour les tableaux :

- `*` pour indiquer que la cellule de gauche peut s'étendre sur l'actuelle
- `.` pour indiquer une cellule vide

```
@startsalt
{#
. | Column 2 | Column 3
Row header 1 | value 1 | value 2
Row header 2 | A long cell | *
}
@endsalt
```

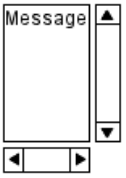
	Column 2	Column 3
Row header 1	value 1	value 2
Row header 2	A long cell	

14.11 Barres de défilement [S, SI, S-]

Vous pouvez utiliser la commande `{S` pour afficher les barres de défilement comme dans les exemples suivants :

- `{S` : barres de défilement verticale et horizontale

```
@startsalt
{S
Message
.
.
.
.
}
@endsalt
```



- {SI : barre de défilement verticale seulement

```
@startsalt
{SI
Message
.
.
.
.
}
@endsalt
```

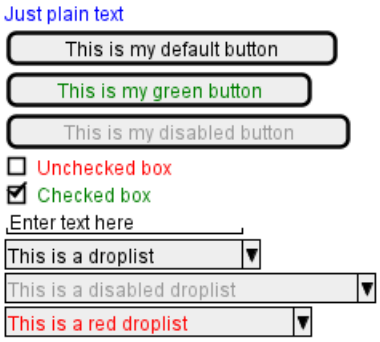
- {S- : barre de défilement horizontale seulement

```
@startsalt
{S-
Message
.
.
.
.
}
@endsalt
```

14.12 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 droplist^
  ^<color:#9a9a9a>This is a disabled droplist^
  ^<color:red>This is a red droplist^
}
@endsalt
```

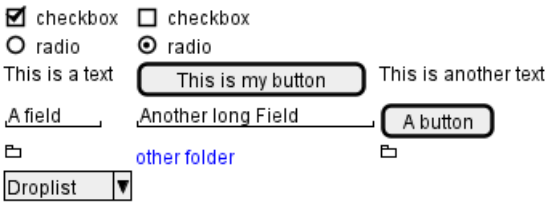


[Ref. QA-12177]

14.13 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
 ............
 .XXXXX......
 .X...X......
 .XXXXXXXXXX.
 .X........X.
 .X........X.
 .X........X.
 .X........X.
 .XXXXXXXXXX.
 ............
 >>|<color:blue>other folder|<<folder>>
^Droplist^
}
@endsalt
```



[Ref. QA-5849]

14.14 OpenIconic

OpenIconic is an 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
```

The complete list is available on OpenIconic Website, or you can use the following special diagram:

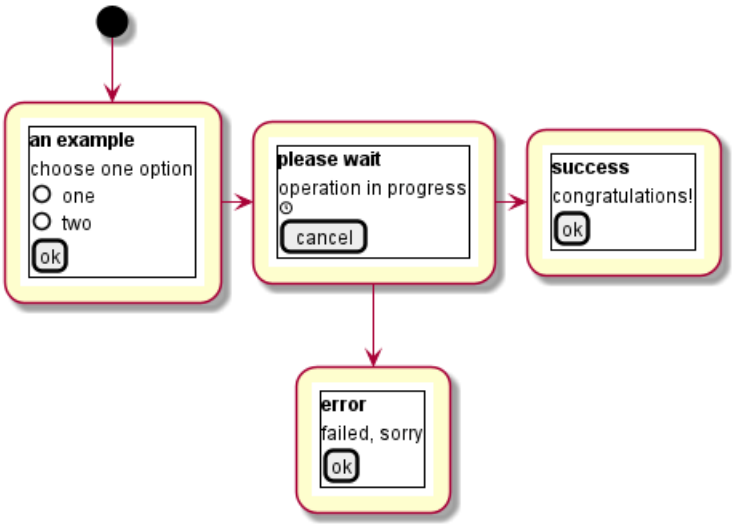
```
@startuml
listopeniconic
@enduml
```

14.15 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
{+
```

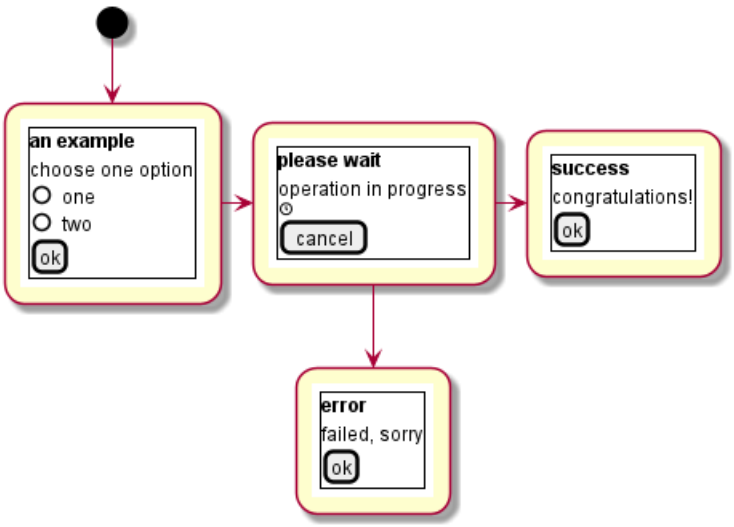
```
<b>please wait
operation in progress
<&clock>
[cancel]
}
}}
" as wait
wait -right-> "
{{
salt
{+
<b>success
congratulations!
[ok]
}
}}
" as success

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()
{+
```

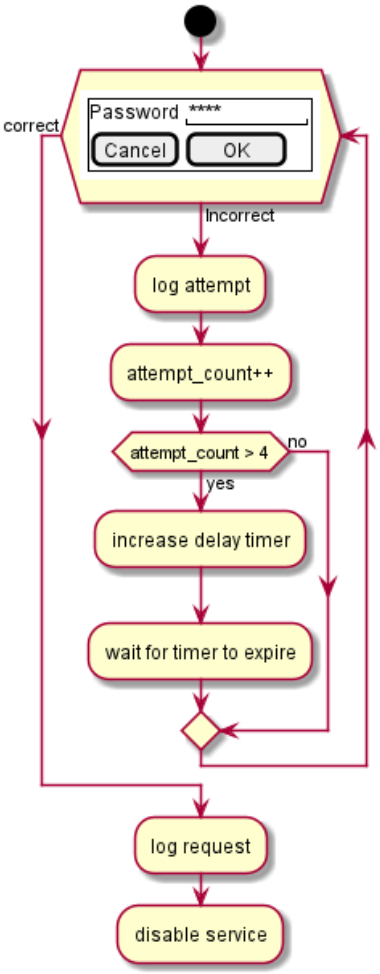
```
<b>an example
choose one option
()one
()two
[ok]
}
!endprocedure

!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
```



14.16 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}}\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]

15 Diagramme Archimate

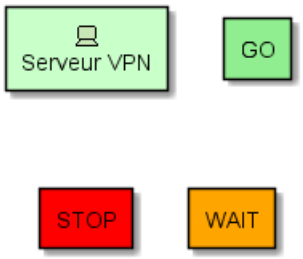
Ceci n'est qu'une proposition qui est susceptible d'évoluer.

Vous êtes invités à créer des discussions sur cette future syntaxe. Vos retours, vos idées et vos suggestions nous aideront à trouver la meilleure solution.

15.1 Mot-clé Archimate

Vous pouvez utiliser le mot-clé `archimate` pour définir un élément. De façon optionnelle, un stéréotype peut indiquer une icône à afficher. Certains noms de couleurs (`Business`, `Application`, `Motivation`, `Strategy`, `Technology`, `Physical`, `Implementation`) sont aussi disponibles.

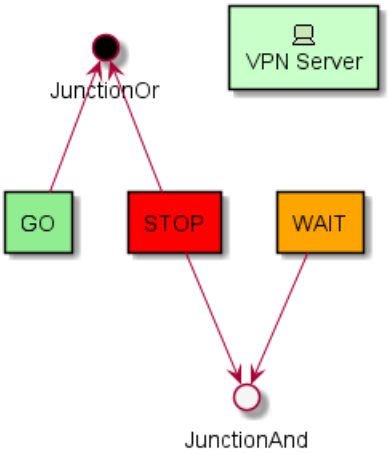
```
@startuml
archimate #Technology "Serveur VPN" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
@enduml
```



15.2 Jonctions Archimate

A l'aide du mot-clé `circle` et du préprocesseur, vous pouvez déclarer des jonctions.

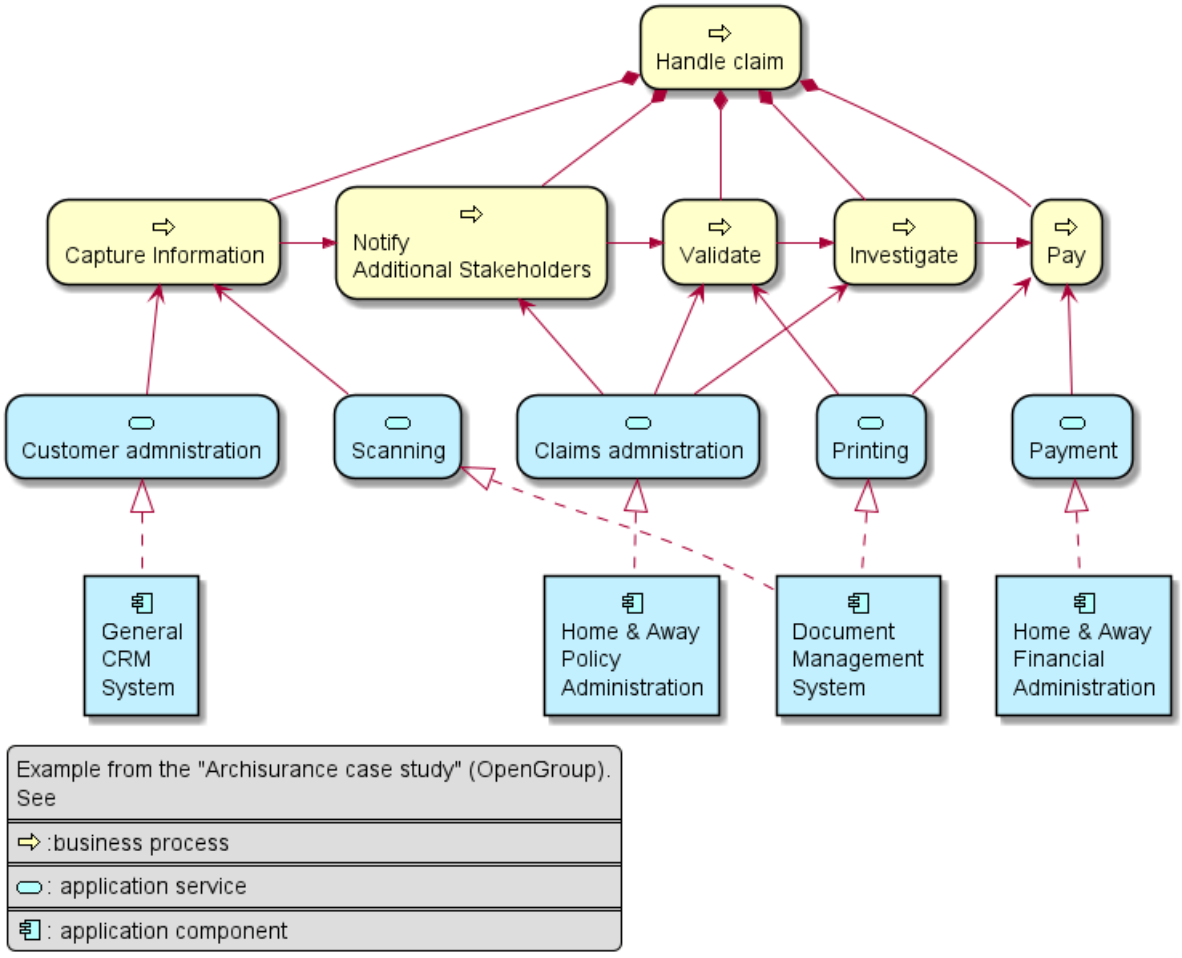
```
@startuml
!define Junction_Or circle #black
!define Junction_And circle #whitesmoke

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 Exemple 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\nManagement\nSystem" as DMS <<$aComponent>> #Application
rectangle "General\nCRM\nSystem" as CRM <<$aComponent>>  #Application
rectangle "Home & Away\nPolicy\nAdministration" as HAPA <<$aComponent>> #Application
rectangle "Home & Away\nFinancial\nAdministration" as HFPA <<$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
```



15.4 Exemple 2

```
@startuml
```

```
skinparam roundcorner 25
rectangle "Capture Information"  as CI <<$archimate/business-process>> #Business
@enduml
```

15.5 Liste des sprites possibles

Vous pouvez afficher tous les sprites disponibles pour Archimate à l'aide du diagramme suivant:

```
@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(S "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

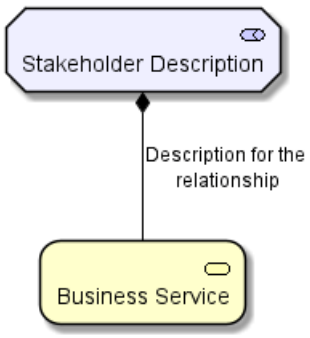
For example:

- 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")
```

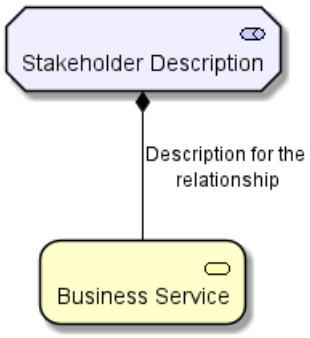
```
Business_Service(BService, "Business Service")
Rel_Composition(StakeholderElement, BService, "Description for the relationship")
@enduml
```



- Unordered List ItemTo orient the two elements in top - down position, the syntax will be

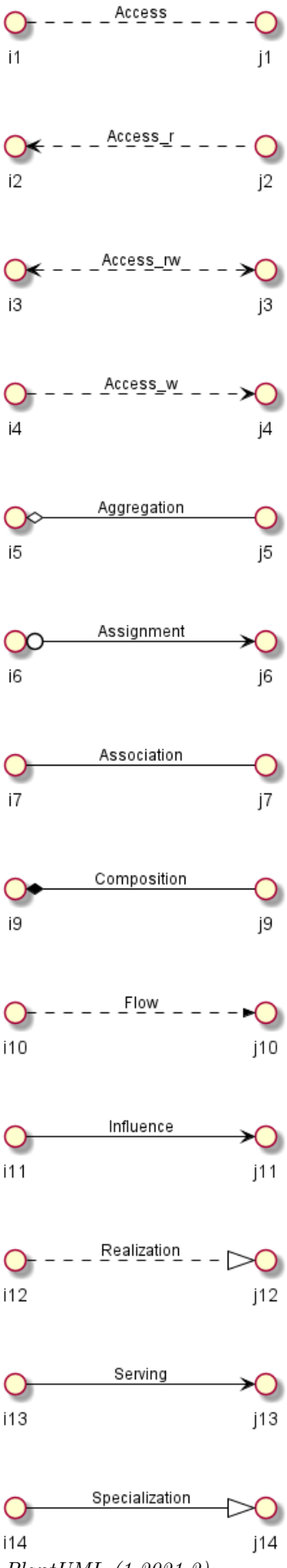
```
Rel_Composition_Down(StakeholderElement, BService, "Description for the relationship")
```

```
@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 Appendice: Examples of all Archimate RelationTypes

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


Access
i1
j1
Access_r
i2
j2
Access_rw
i3
j3
Access_w
i4
j4
Aggregation
i5
j5
Assignment
i6
j6
Association
i7
j7
Composition
i9
j9
Flow
i10
j10
Influence
i11
j11
Realization
i12
j12
Serving
i13
j13
Specialization
i14
j14
Triggering

16 Diagramme de Gantt

Le Gantt doit être décrit en anglais, à l'aide de phrase très simple (sujet-verbe-complément).

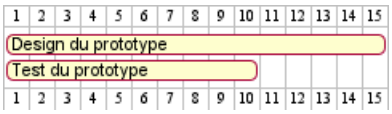
16.1 Définir des tâches (avec les verbes : last, start, end)

Les tâches sont définies à l'aide des crochets.

16.1.1 Durée

Leur durée est définie à l'aide du verbe `last`.

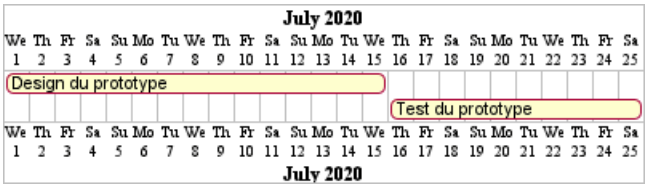
```
@startuml
[Design du prototype] lasts 15 days
[Test du prototype] lasts 10 days
@enduml
```



16.1.2 Début

Leur début est définie à l'aide du verbe `start`.

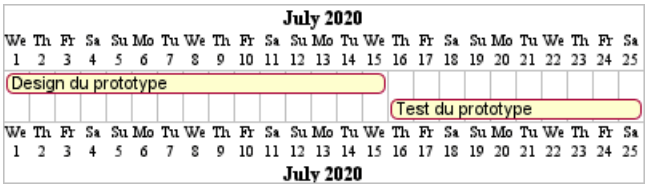
```
@startuml
[Design du prototype] lasts 15 days
[Test du prototype] lasts 10 days

Project starts 2020-07-01
[Design du prototype] starts 2020-07-01
[Test du prototype] starts 2020-07-16

@enduml
```



16.1.3 Fin

Leur fin est définie à l'aide du verbe `end`.

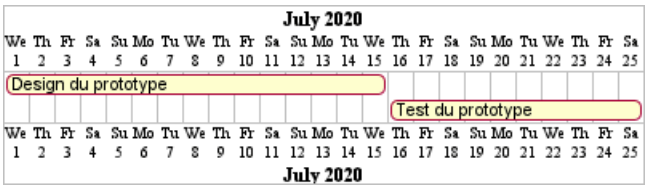
```
@startuml
[Design du prototype] lasts 15 days
[Test du prototype] lasts 10 days

Project starts 2020-07-01
[Design du prototype] ends 2020-07-15
[Test du prototype] ends 2020-07-25

@enduml
```



16.1.4 Début/Fin

Il est possible de définir les deux de façon absolu, en précisant des dates.

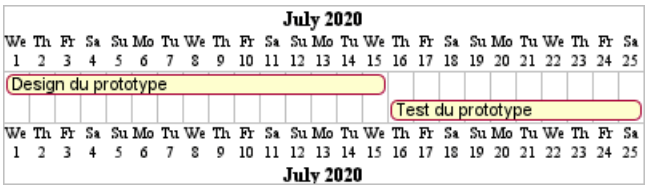
```
@startuml

Project starts 2020-07-01
[Design du prototype] starts 2020-07-01
[Test du prototype] starts 2020-07-16
[Design du prototype] ends 2020-07-15
[Test du prototype] ends 2020-07-25

@enduml
```



16.2 Déclaration sur une ligne (avec la conjonction 'and')

Il est possible de combiner les déclarations sur une ligne en utilisant la conjonction `and`.

```
@startuml
Project starts 2020-07-01
[Design du prototype] starts 2020-07-01 and ends 2020-07-15
[Test du prototype] starts 2020-07-16 and lasts 10 days
@enduml
```

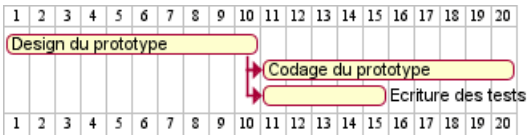


16.3 Ajout de contraintes

Il est possible de rajouter des contraintes entre les tâches.

```
@startuml
[Design du prototype] lasts 15 days
[Test du prototype] lasts 10 days
[Test du prototype] starts at [Design du prototype]'s end
@enduml
```

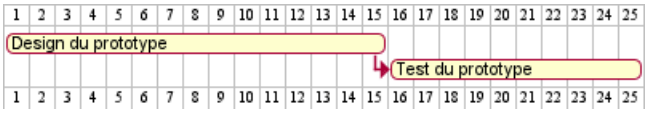

```
@startuml
[Design du prototype] lasts 10 days
[Codage du prototype] lasts 10 days
[Ecriture des tests] lasts 5 days
[Codage du prototype] starts at [Design du prototype]'s end
[Ecriture des tests] starts at [Codage du prototype]'s start
@enduml
```



16.4 Noms courts (avec : as)

Un nom court peut être utilisé pour les tâches à l'aide de l'instruction `as`.

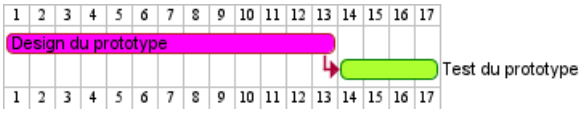
```
@startuml
[Design du prototype] as [D] lasts 15 days
[Test du prototype] as [T] lasts 10 days
[T] starts at [D]'s end
@enduml
```



16.5 Choix des couleurs

Il est possible de changer les couleurs des tâches avec les mots clés `is colored in`.

```
@startuml
[Design du prototype] lasts 13 days
[Test du prototype] lasts 4 days
[Test du prototype] starts at [Design du prototype]'s end
[Design du prototype] is colored in Fuchsia/FireBrick
[Test du prototype] is colored in GreenYellow/Green
@enduml
```



16.6 Complétion d'une tâche

Il est possible d'attribuer un pourcentage d'avancement à une tâche avec les mots clés `is xx% completed`.

```
@startgantt
[foo] lasts 21 days
[foo] is 40% completed
[bar] lasts 30 days and is 10% complete
@endgantt
```

16.7 Jalon (avec le verbe : happen)

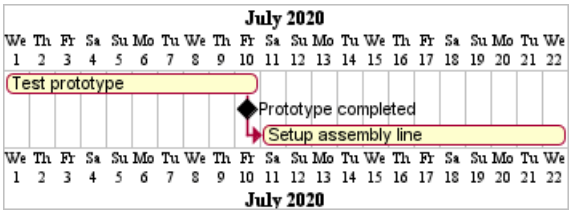
Vous pouvez définir des jalons à l'aide du verbe `happen`.

16.7.1 Jalon relatif (avec utilisation de contraintes sur des tâches)

```
@startuml
[Test du prototype] lasts 10 days
[Prototype terminé] happens at [Test du prototype]'s end
[Mise en place production] lasts 12 days
[Mise en place production] starts at [Test du prototype]'s end
@enduml
```

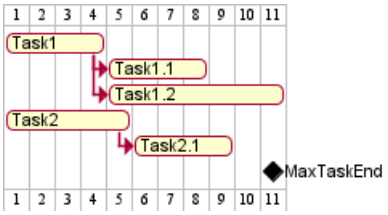
16.7.2 Jalon absolu (avec utilisation d'un date fixe)

```
@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 Jalon de fin maximale de tâches

```
@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
```

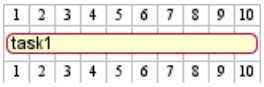


[Ref. QA-10764]

16.8 Lien hypertexte

On peut rajouter des liens aux tâches avec le mot clé `links to`.

```
@startgantt
[task1] lasts 10 days
[task1] links to [[http://plantuml.com]]
@endgantt
```



16.9 Calendrier

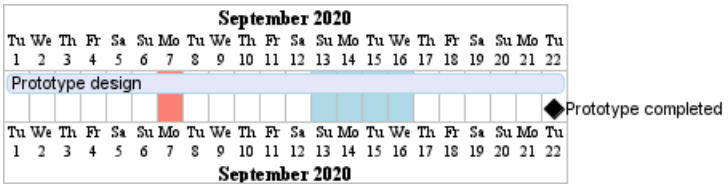
Vous pouvez définir une date de début pour l'ensemble du projet. Par défault, la première tâche commence à cette date.

```
@startuml
Project starts the 20th of september 2017
[Design du prototype] as [TASK1] lasts 13 days
[TASK1] is colored in Lavender/LightBlue
@enduml
```

16.10 Colorer des jours

Il est possible d'ajouter des couleurs sur certains jours.

```
@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
```



16.11 Zoom

Il est possible de changer l'échelle d'affichage pour les projets qui sont très longs, avec l'un des paramètres suivant :

- printscale
- ganttscale
- projectscale

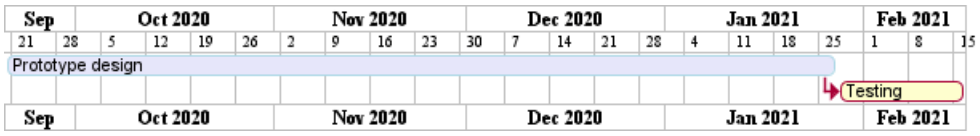
et une des valeurs suivantes :

- weekly *(par semaine)*
- monthly *(par mois)*

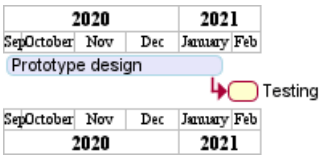
(See QA-11272, QA-9041 and QA-10948)

```
@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]
```

```
@endgantt
```



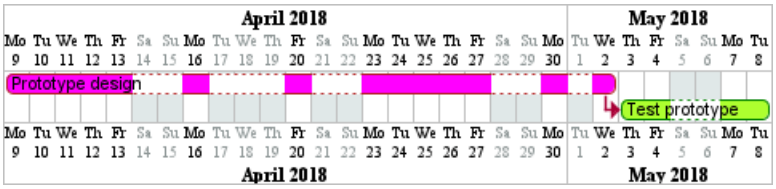

```
@startgantt
projectscale monthly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
@endgantt
```



16.12 Jours non travaillés

Il est possible de déclarer certains jours comme non travaillés.

```
@startuml
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
@enduml
```



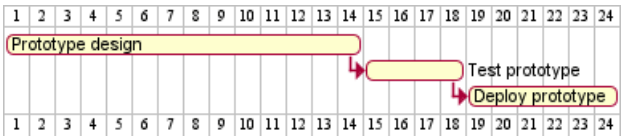
Puis il est possible de déclarer certains jours non travaillés, comme finalement, travaillés.

```
@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
```

16.13 Succession des tâches simplifiée et dépendances

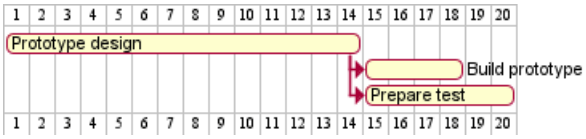
Il est possible d'utiliser le mot-clés `then` pour définir des tâches consécutives.

```
@startuml
[Prototype design] lasts 14 days
then [Test prototype] lasts 4 days
then [Deploy prototype] lasts 6 days
@enduml
```



Vous pouvez aussi utiliser une flèche `->`.

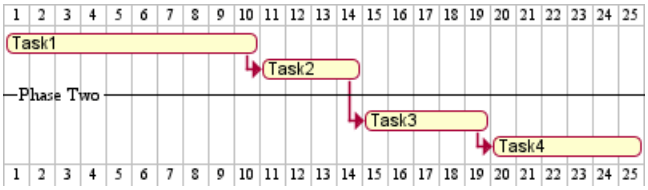
```
@startuml
[Prototype design] lasts 14 days
[Build prototype] lasts 4 days
[Prepare test] lasts 6 days
[Prototype design] -> [Build prototype]
[Prototype design] -> [Prepare test]
@enduml
```



16.14 Séparateur

Il est possible d'utiliser `--` pour séparer ou grouper des tâches ensemble.

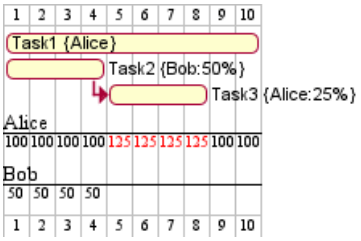
```
@startuml
[Task1] lasts 10 days
then [Task2] lasts 4 days
-- Phase Two --
then [Task3] lasts 5 days
then [Task4] lasts 6 days
@enduml
```



16.15 Travailler avec des ressources

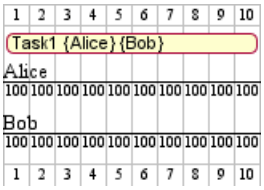
Vous pouvez affecter des ressources aux tâches à l'aide du mot-clé `on` et en mettant le nom de la ressource entre accolades. Vous pouvez aussi rajouter un taux de charge en pourcentage.

```
@startgantt
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
@endgantt
```



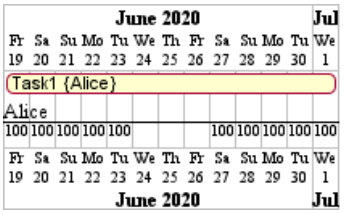
Plusieurs ressources peuvent être affectées à une tâche :

```
@startgantt
[Task1] on {Alice} {Bob} lasts 20 days
@endgantt
```



Les ressources peuvent être marquées comme inopérantes certains jours:

```
@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
```

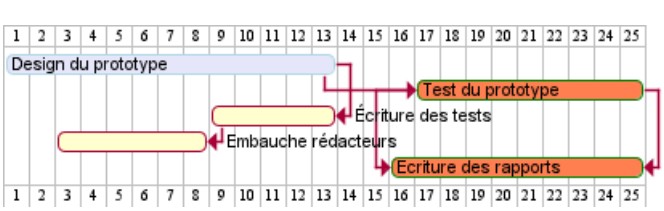


16.16 Exemple plus complexe

On peut se servir de la conjonction de coordination `and`.

Il est aussi possible de spécifier un délai (avec `before` ou `after`) dans les contraintes.

```
@startuml
[Design du prototype] lasts 13 days and is colored in Lavender/LightBlue
[Test du prototype] lasts 9 days and is colored in Coral/Green and starts 3 days after [Design du pr
[Écriture des tests] lasts 5 days and ends at [Design du prototype]'s end
[Embauche rédacteurs] lasts 6 days and ends at [Écriture des tests]'s start
[Ecriture des rapports] is colored in Coral/Green
[Ecriture des rapports] starts 1 day before [Test du prototype]'s start and ends at [Test du prototy
@enduml
```

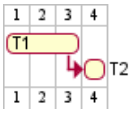


16.17 Comments

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` *')*

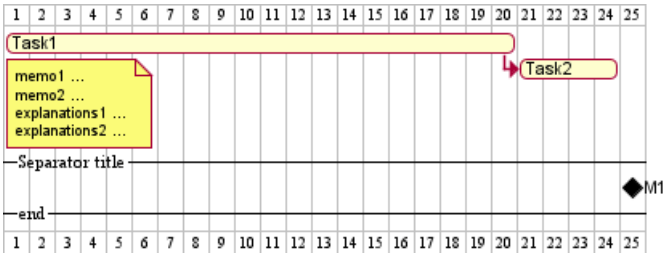
```
@startgantt
' This is a comment

[T1] lasts 3 days

/' this comment
is on several lines '/

[T2] starts at [T1]'s end and lasts 1 day
@endgantt
```



16.18 Using style

16.18.1 Without style (by default)

```
@startuml
[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 --
@enduml
```

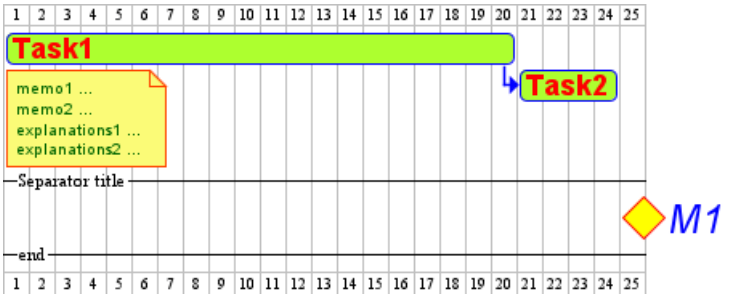


16.18.2 With style

You can use style to change rendering of elements.

```
@startuml
<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 --
@enduml
```



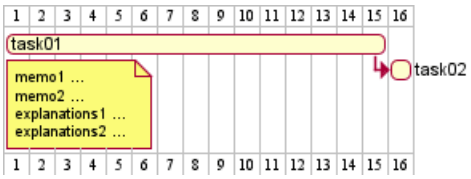
[Ref. QA-10835, QA-12045, QA-11877 and PR-438]

TODO: TODO *Awaiting style for Separator and all style for Arrow (thickness...)*

16.19 Add notes

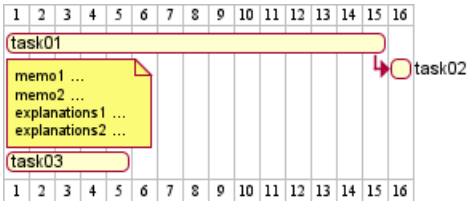
```
@startgantt
```

```
[task01] lasts 15 days
note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note

[task01] -> [task02]

@endgantt
```



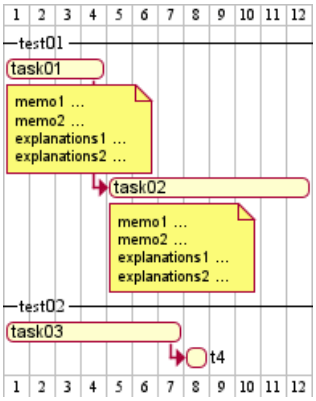
Example with overlap.

```
@startgantt
[task01] lasts 15 days
note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note

[task01] -> [task02]
[task03] lasts 5 days

@endgantt
```



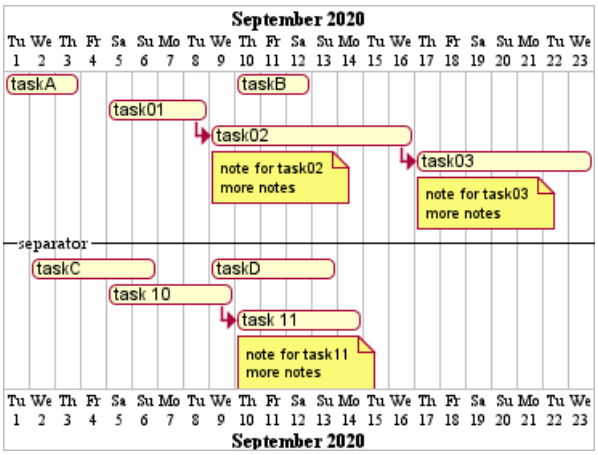

```
@startgantt

-- 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
```



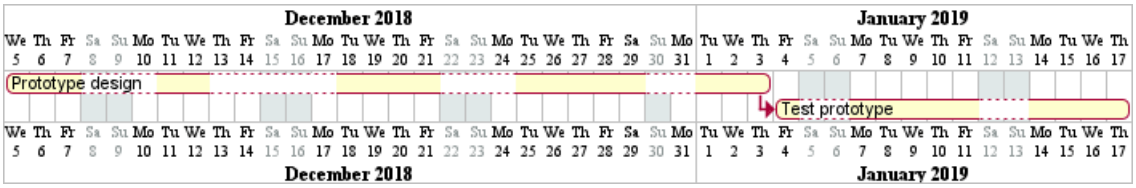
TODO: DONE *Thanks for correction (of #386 on v1.2020.18) when overlapping*

```
@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
```



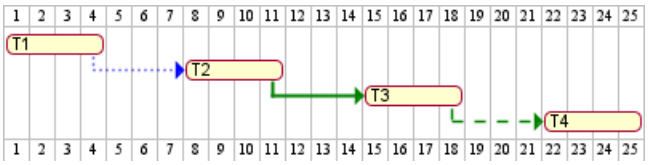
16.20 Pause tasks

```
@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
```

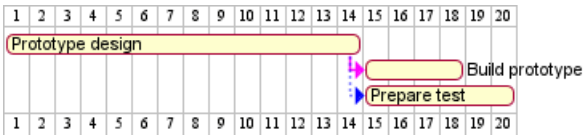


16.21 Change link colors

```
@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
```

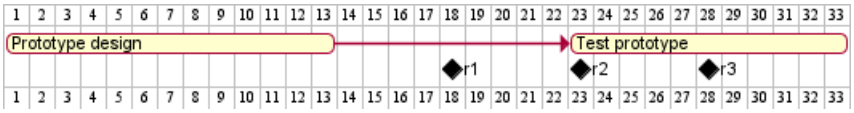



```
@startuml
Links are colored in blue
[Prototype design] lasts 14 days
[Build prototype] lasts 4 days
[Prototype design] -[#FF00FF]-> [Build prototype]
[Prototype design] -[dotted]-> [Prepare test]
@enduml
```



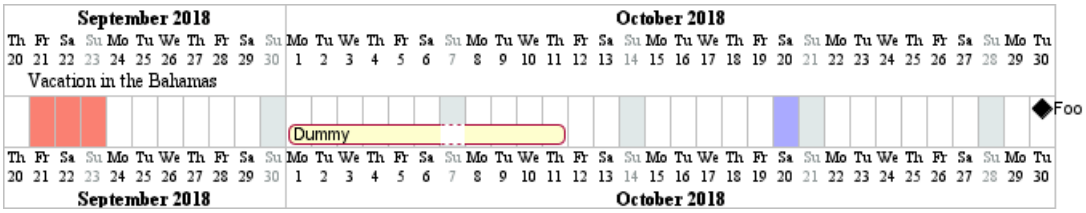
16.22 Tasks or Milestones on the same line

```
@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
```



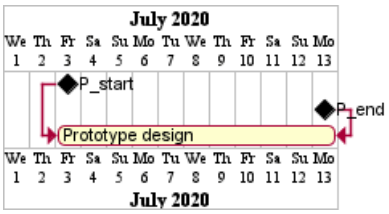
16.23 Highlight today

```
@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
```



16.24 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
```



16.25 Grammar and verbal form

Verbal form	Example
[*T*] starts	
[*M*] happens	

16.26 Add title, header, footer, caption or legend on gantt diagram

```
@startuml

header some header

footer some footer

title My title

[Prototype design] lasts 13 days

legend
The legend
end legend

caption This is caption

@enduml
```

(See also: Common commands)

16.27 Removing Foot Boxes

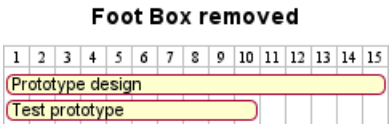
You can use the `hide footbox` keywords to remove the foot boxes of the gantt diagram *(as for sequence diagram)*.

Examples on:

- daily scale *(without project start)*

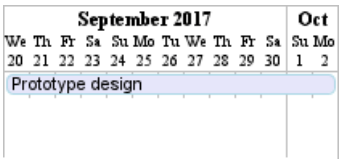
```
@startgantt

hide footbox
title Foot Box removed

[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
```

```
@endgantt
```



- 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
```


2020
2021
Sep
October
Nov
Dec
January
Feb
Prototype design
Testing

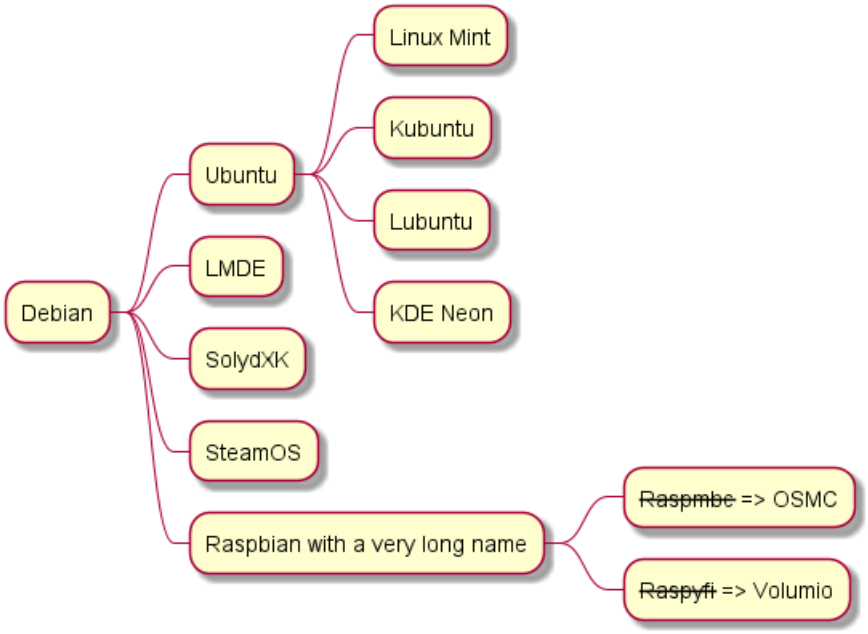
17 MindMap

Les cartes MindMap sont encore en beta : la syntaxe peut être amenée à évoluer.

17.1 Syntaxe OrgMode

La syntaxe OrgMode est supportée.

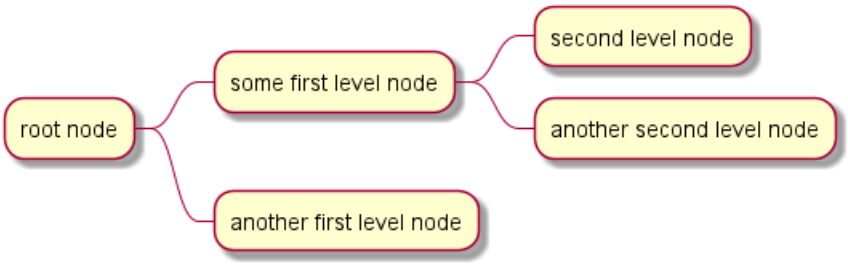
```
@startmindmap
* Debian
** Ubuntu
*** Linux Mint
*** Kubuntu
*** Lubuntu
*** KDE Neon
** LMDE
** SolydXK
** SteamOS
** Raspbian with a very long name
*** <s>Raspmbc</s> => OSMC
*** <s>Raspyfi</s> => Volumio
@endmindmap
```



17.2 Syntaxe Markdown

La syntaxe Markdown est supportée.

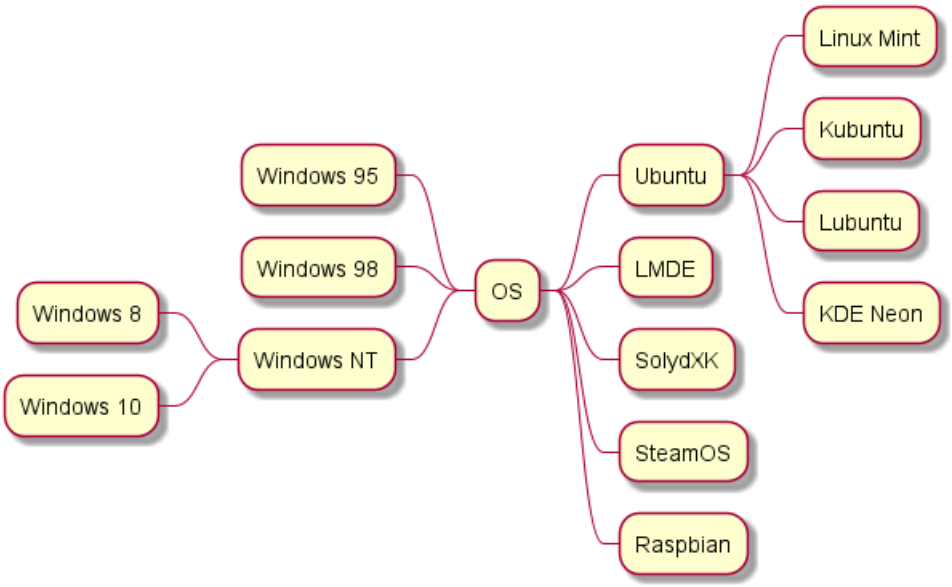
```
@startmindmap
* root node
* some first level node
* second level node
* another second level node
* another first level node
@endmindmap
```



17.3 Notation arithmétique [+, -]

Vous pouvez utiliser la notation suivante pour orienter votre diagramme.

```
@startmindmap
+ OS
++ Ubuntu
+++ Linux Mint
+++ Kubuntu
+++ Lubuntu
+++ KDE Neon
++ LMDE
++ SolydXK
++ SteamOS
++ Raspbian
-- Windows 95
-- Windows 98
-- Windows NT
--- Windows 8
--- Windows 10
@endmindmap
```

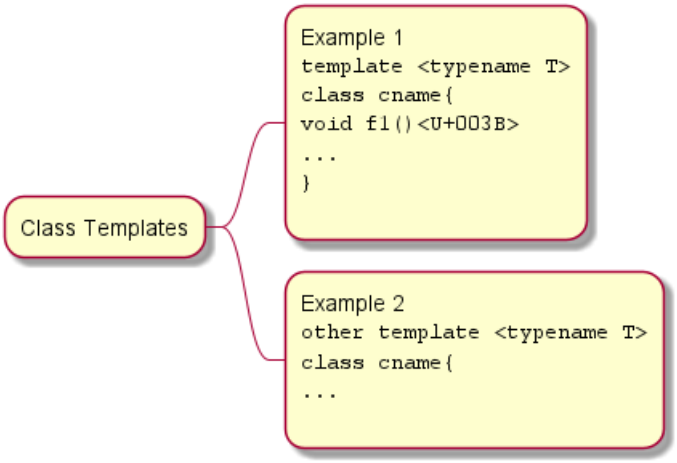


17.4 Multilignes

Le contenu multiligne des boîtes commence avec : et finisse avec ;.

```
@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
```

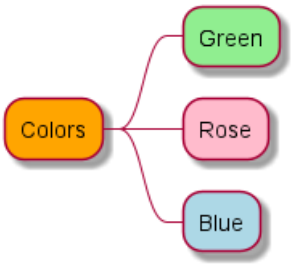


(Penser à échapper le `;`, s'il apparait en fin de ligne intermédiaire dans le contenu, par exemple par son correspondant unicode `<U+003B>`)

17.5 Couleurs

Il est possible de changer la couleur des boîtes.

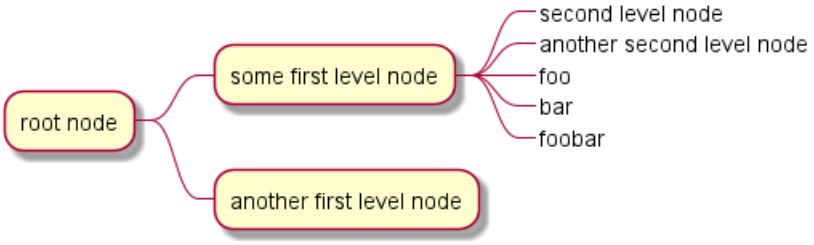
```
@startmindmap
*[#Orange] Colors
**[#lightgreen] Green
**[#FFBBCC] Rose
**[#lightblue] Blue
@endmindmap
```



17.6 Masquer les bordures [_]

Vous pouvez enlever les contours des boîtes en utilisant le caractère tiret bas (_), comme pour les diagrammes de type WBS.

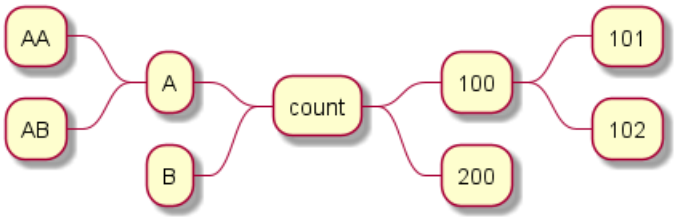
```
@startmindmap
* root node
** some first level node
***_ second level node
***_ another second level node
***_ foo
***_ bar
***_ foobar
** another first level node
@endmindmap
```



17.7 Diagramme multi-directionnel

Il est possible d'utiliser les deux côtés du diagramme.

```
@startmindmap
* count
** 100
*** 101
*** 102
** 200

left side

** A
*** AA
*** AB
** B
@endmindmap
```



17.8 Exemple complet

```
@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
```

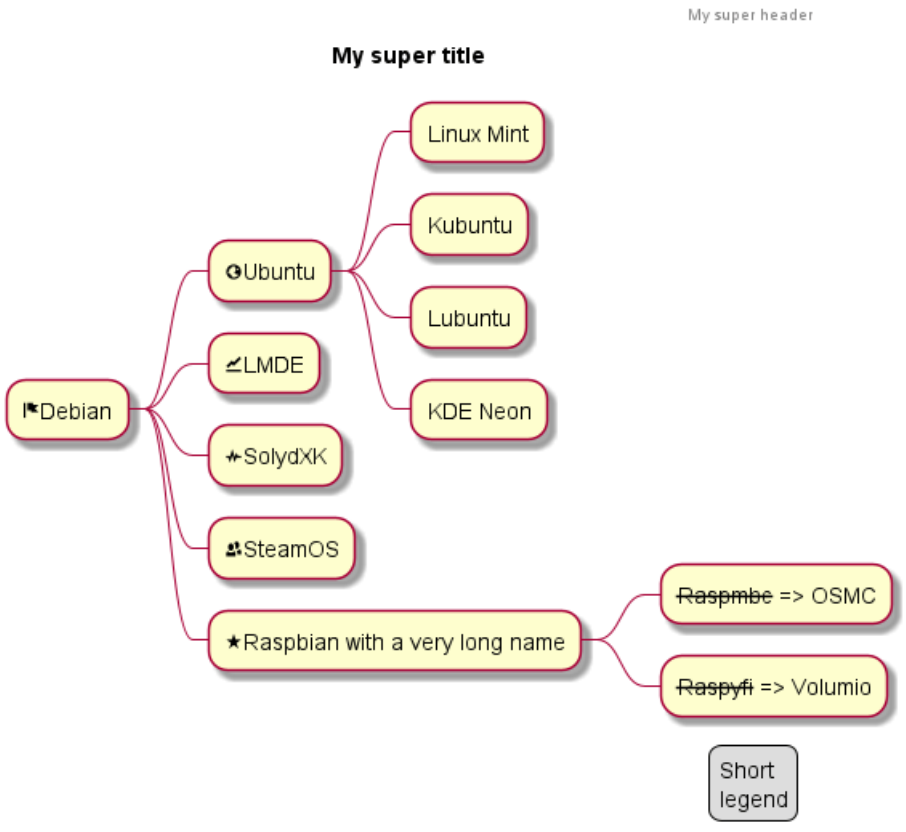


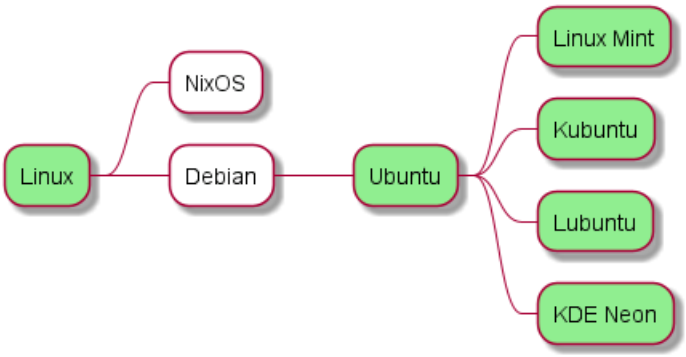
figure 1

My super footer

17.9 Changing style

17.9.1 node, depth

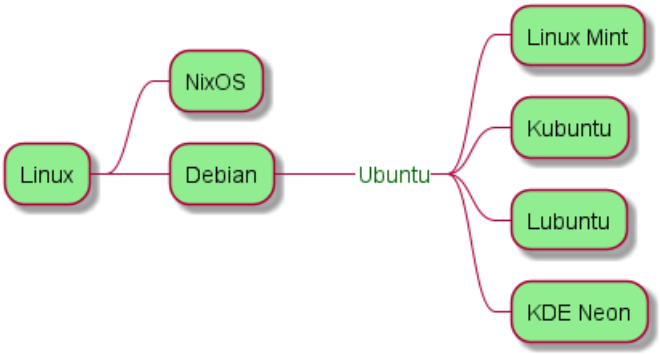
```
@startmindmap
<style>
mindmapDiagram {
    node {
        BackgroundColor lightGreen
    }
    :depth(1) {
      BackGroundColor white
    }
}
</style>
* Linux
** NixOS
** Debian
*** Ubuntu
**** Linux Mint
**** Kubuntu
**** Lubuntu
**** KDE Neon
@endmindmap
```



17.9.2 boxless

```
@startmindmap
<style>
mindmapDiagram {
  node {
    BackgroundColor lightGreen
  }
  boxless {
    FontColor darkgreen
  }
}
</style>
* Linux
** NixOS
** Debian
***_ Ubuntu
**** Linux Mint
**** Kubuntu
**** Lubuntu
**** KDE Neon
```

```
@endmindmap
```

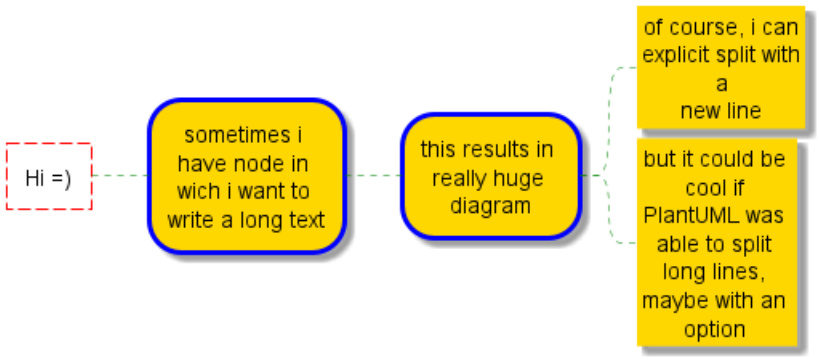


17.10 Word Wrap

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 wich 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
```



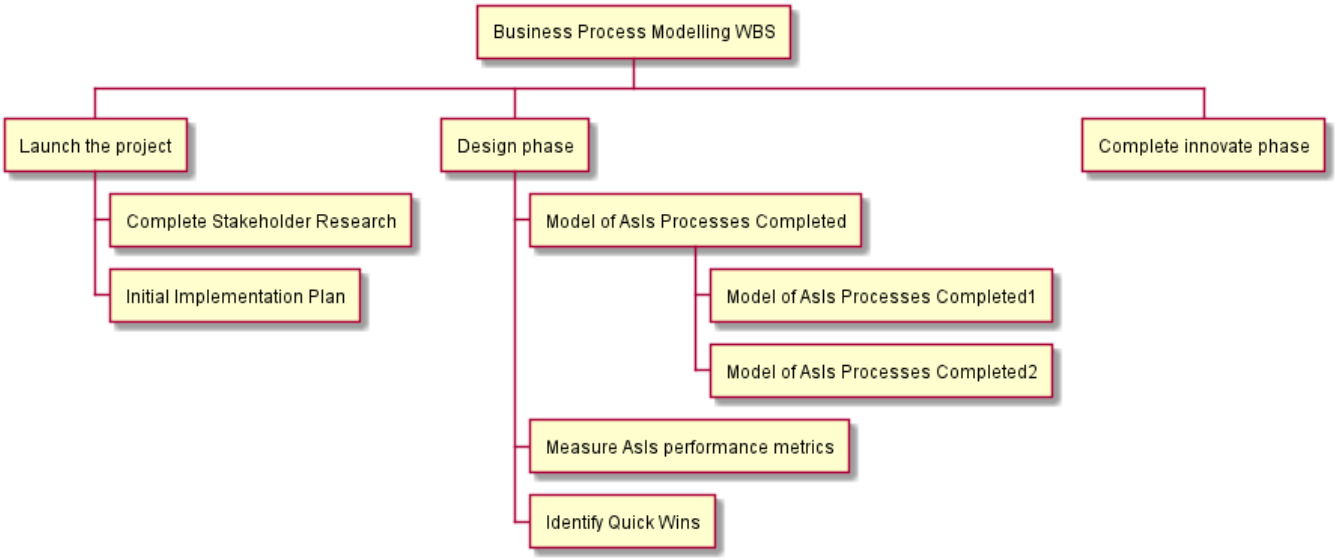
18 Work Breakdown Structure (WBS)

Les diagrammes de type *organigramme des tâches du projet (OTP)* ou *structure de découpage du projet (SDP)*, ou encore *Work Breakdown Structure (WBS)* (en anglais) sont encore en phase de test : leur syntaxe peut évoluer sans aucune garantie.

18.1 Syntaxe OrgMode

La syntaxe est compatible avec celle de 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
```



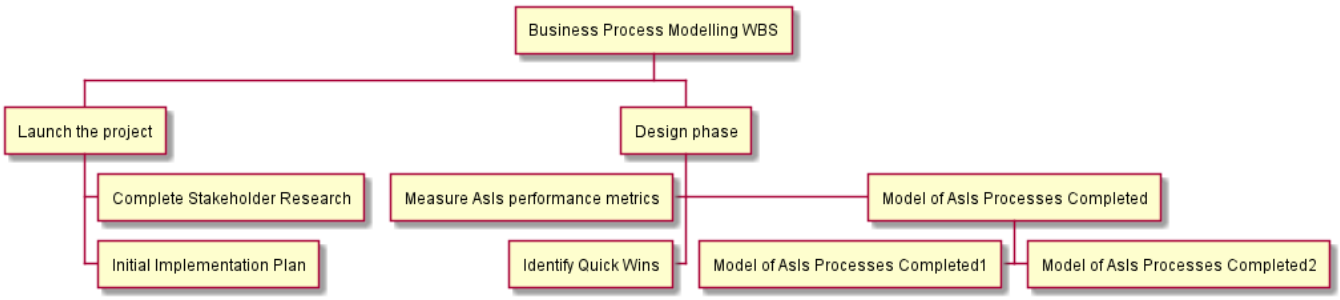
18.2 Changement de direction [<, >]

Vous pouvez changer de direction en utilisant :

- <
- >

```
@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
```

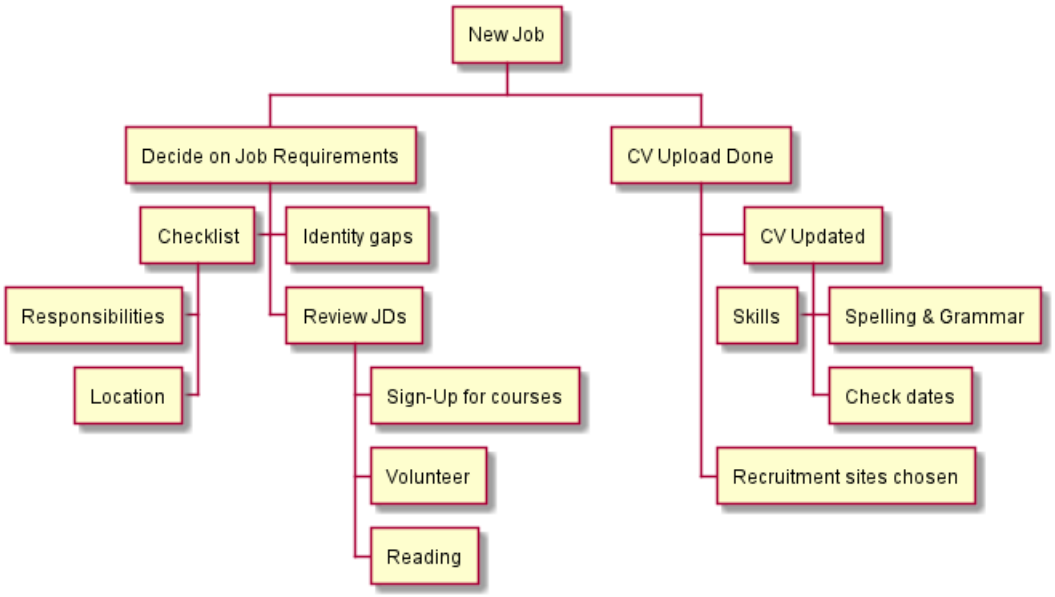
```
***< Measure AsIs performance metrics
***< Identify Quick Wins
@endwbs
```



18.3 Notation arithmétique [+, -]

Vous pouvez utiliser la notation suivante (avec des + ou des -) pour choisir le côté du diagramme.

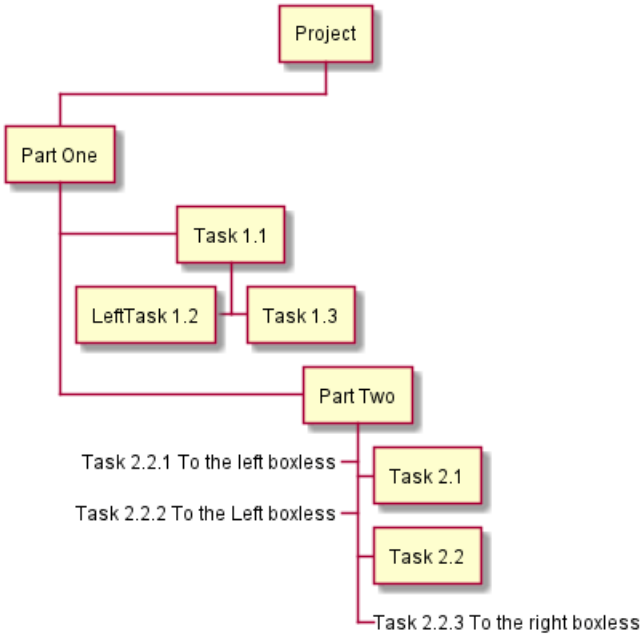
```
@startwbs
+ New Job
++ Decide on Job Requirements
+++ Identity gaps
+++ Review JDs
++++ Sign-Up for courses
++++ Volunteer
++++ Reading
++- Checklist
+++- Responsibilities
+++- Location
++ CV Upload Done
+++ CV Updated
++++ Spelling & Grammar
++++ Check dates
---- Skills
+++ Recruitment sites chosen
@endwbs
```



18.4 Masquer les bordures [_]

Vous pouvez enlever les contours des boîtes en utilisant le caractère tiret bas (_), comme pour les cartes MindMap.

```
@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
```



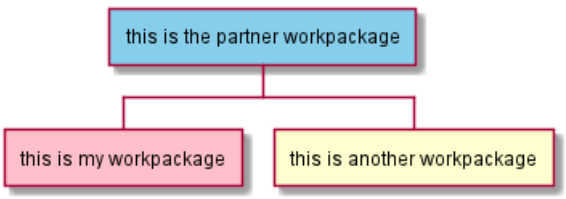
[Ref. QA-13297] [Ref. QA-13355]

18.5 Colors (with inline or style color)

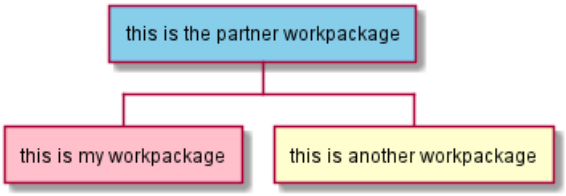
It is possible to change node color:

- with inline color

```
@startwbs
*[#SkyBlue] this is the partner workpackage
**[#pink] this is my workpackage
** this is another workpackage
@endwbs
```



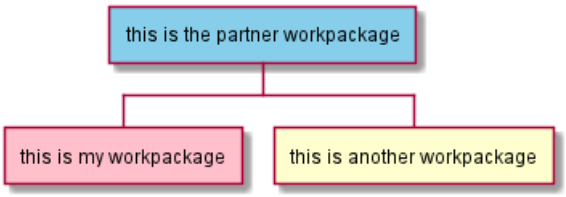
```
@startwbs
+[#SkyBlue] this is the partner workpackage
++[#pink] this is my workpackage
++ this is another workpackage
@endwbs
```



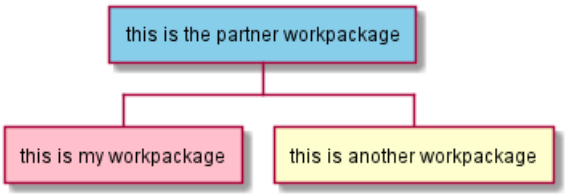
[Ref. QA-12374, only from v1.2020.20]

- 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
```

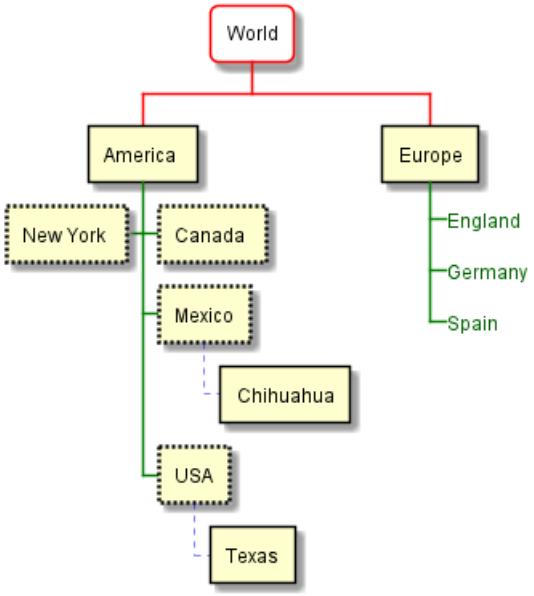


18.6 Using style

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
```

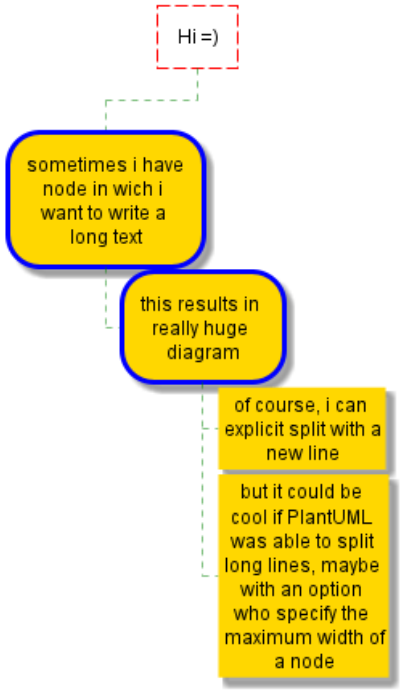


18.7 Word Wrap

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
}
```

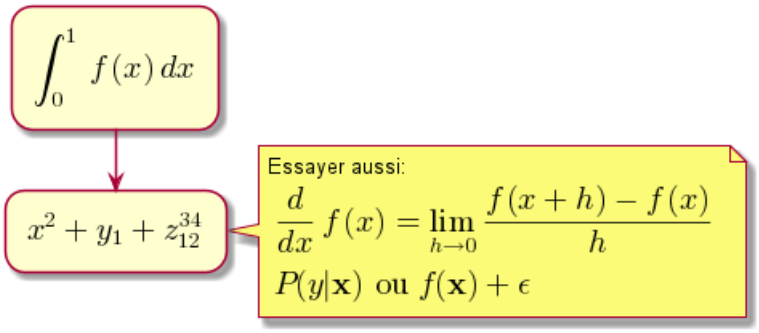
```
arrow {
    LineStyle 4
    LineThickness 0.5
    LineColor green
}
</style>

* Hi =)
** sometimes i have node in wich 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
```



19 Mathématiques

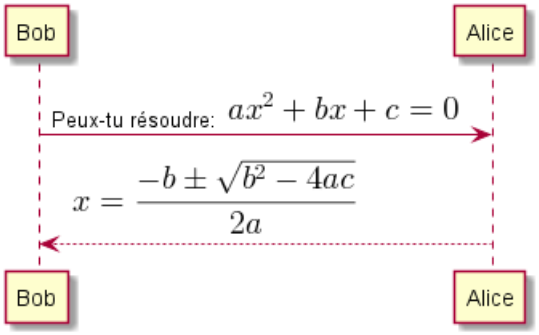
Vous pouvez utiliser les notations AsciiMath ou JLaTeXMath dans PlantUML:

```
@startuml
:<math>int_0^1f(x)dx</math>;
:<math>x^2+y_1+z_12^34</math>;
note right
Essayer aussi:
<math>d/dxf(x)=lim_(h->0)(f(x+h)-f(x))/h</math>
<latex>P(y|\mathbf{x}) \mbox{ ou } f(\mathbf{x})+\epsilon</latex>
end note
@enduml
```



ou encore:

```
@startuml
Bob -> Alice : Peux-tu résoudre: <math>ax^2+bx+c=0</math>
Alice --> Bob: <math>x = (-b+-sqrt(b^2-4ac))/(2a)</math>
@enduml
```



19.1 Diagramme indépendant

Il est possible d'utiliser `@startmath`/`@endmath` pour créer des formules AsciiMath.

```
@startmath
f(t)=(a_0)/2 + sum_(n=1)^ooa_ncos((npit)/L)+sum_(n=1)^oo b_n\ sin((npit)/L)
@endmath
```

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi t}{L}\right) + \sum_{n=1}^{\infty} b_n \sin\left(\frac{n\pi t}{L}\right)$$

Ou bien utiliser `@startlatex`/`@endlatex` pour créer des formules JLaTeXMath.

```
@startlatex
\sum_{i=0}^{n-1} (a_i + b_i^2)
@endlatex
```

$$\sum_{i=0}^{n-1}(a_i + b_i^2)$$

19.2 Comment cela fonctionne ?

Pour dessiner ces formules, PlantUML utilise deux projets OpenSource:

- AsciiMath qui convertit la notation AsciiMath vers une expression LaTeX.
- JLatexMath qui dessine une formule mathématique écrite en LaTeX. JLaTeXMath est le meilleur projet Java pour dessiner du code LaTeX.

ASCIIMathTeXImg.js est suffisamment petit pour être intégré dans la distribution standard de PlantUML.

PlantUML relies on the Java Scripting API (specifically: `new ScriptEngineManager().getEngineByName("JavaScript"` to load a JavaScript engine and execute JavaScript code. Java 8 includes a JavaScript engine called Nashorn but it was deprecated in Java 11.

If you are using AsciiMath in Java 11 you see the following warnings:

```
Warning: Nashorn engine is planned to be removed from a future JDK release
```

Nashorn was removed in Java 15. Fortunately, you can use the GraalVM JavaScript Engine instead by adding the following dependencies:

```
<dependency>
  <groupId>org.graalvm.js</groupId>
  <artifactId>js</artifactId>
  <version>20.2.0</version>
</dependency>
<dependency>
  <groupId>org.graalvm.js</groupId>
  <artifactId>js-scriptengine</artifactId>
  <version>20.2.0</version>
</dependency>
```

You can even use the GraalVM JavaScript Engine in Java 11 to get rid of the warning messages.

Comme JLatexMath est plus gros, vous devez le télécharger séparément, puis extraire les 4 fichiers (*batik-all-1.7.jar*, *jlatexmath-minimal-1.0.3.jar*, *jlm_cyrillic.jar* et *jlm_greek.jar*) dans le même répertoire que PlantUML.jar.

20 Entity Relationship Diagram

Based on the Information Engineering notation.

This is an extension to the existing Class Diagram. This extension adds:

- Additional relations for the Information Engineering notation.
- An `entity` alias that maps to the class diagram `class`.
- An additional visibility modifier `*` to identify mandatory attributes.

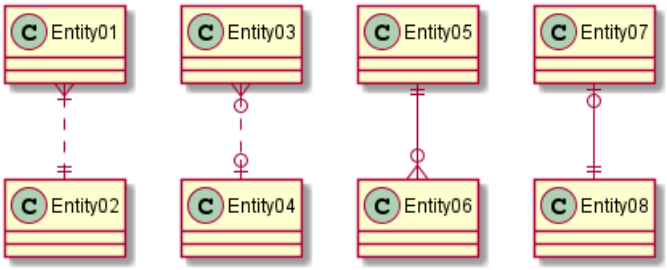
Otherwise, the syntax for drawing diagrams is the same as for class diagrams. All other features of class diagrams are also supported.

20.1 Information Engineering Relations

Type	Symbol
Zero or One	`|o--`
Exactly One	`||--`
Zero or Many	`}o--`
One or Many	`}|--`

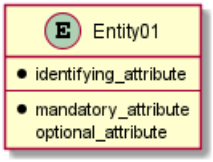
Examples:

```
@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
```



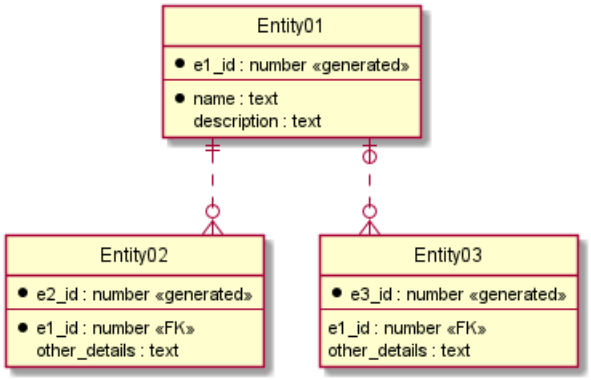
Again, this is the normal class diagram syntax (aside from use of `entity` instead of `class`). Anything that you can do in a class diagram can be done here.

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 Complete Example

```
@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
```



Currently the crows feet do not look very good when the relationship is drawn at an angle to the entity. This can be avoided by using the `linetype ortho` skinparam.

21 Common commands

21.1 Comments

Everything that starts with `simple quote '` is a comment.

You can also put comments on several lines using `/'` to start and `'/` to end.

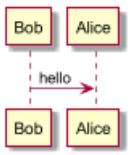
21.2 Zoom

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
```

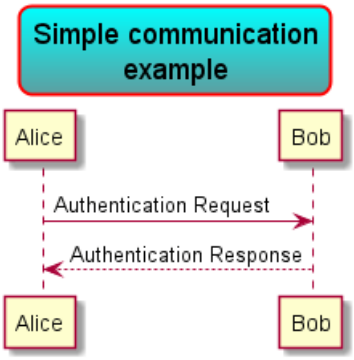


21.3 Title

The `title` keywords is used to put a title. You can add newline using in the title description.

Some skinparam settings are available to put borders on the title.

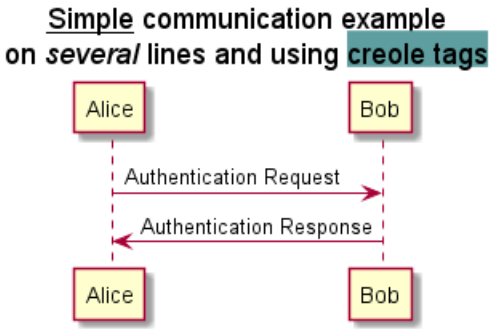
```
@startuml
skinparam titleBorderRoundCorner 15
skinparam titleBorderThickness 2
skinparam titleBorderColor red
skinparam titleBackgroundColor Aqua-CadetBlue

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.

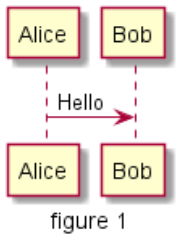
```
@startuml

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
```



21.4 Caption

There is also a `caption` keyword to put a caption under the diagram.

```
@startuml

caption figure 1
Alice -> Bob: Hello

@enduml
```



21.5 Footer and header

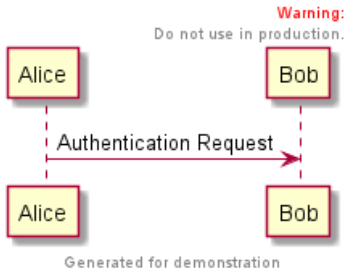
You can use the commands `header` or `footer` to add a footer or a header on any generated diagram.

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
```

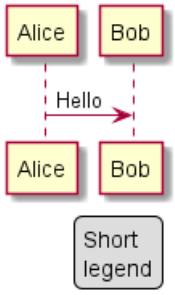


21.6 Legend the diagram

The `legend` and `end legend` are keywords is used to put a legend.

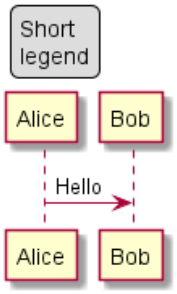
You can optionally specify to have `left`, `right`, `top`, `bottom` or `center` alignment for the legend.

```
@startuml
Alice -> Bob : Hello
legend right
  Short
  legend
endlegend
@enduml
```




```
@startuml
Alice -> Bob : Hello
legend top left
  Short
  legend
endlegend
```

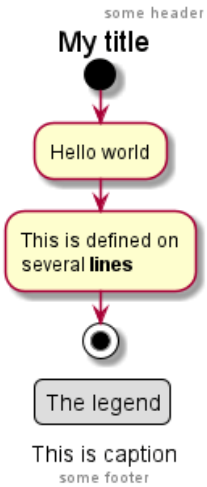
```
@enduml
```



21.7 Appendix: Examples on all diagram

21.7.1 Activity

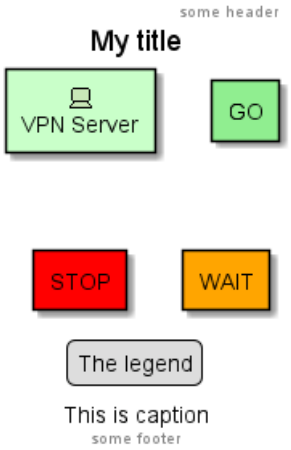
```
@startuml
header some header

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

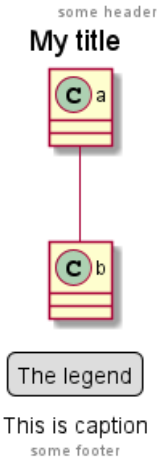
```
@startuml
header some header
```

```
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
```



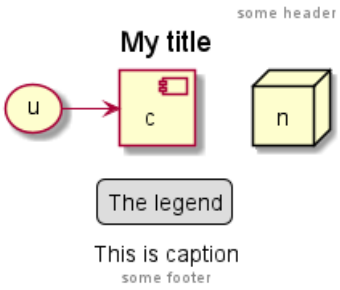
21.7.3 Class

```
@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

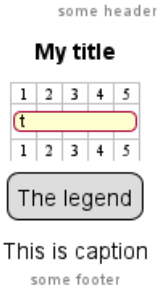
```
@startuml
header some header

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

```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
```

```
end legend


[t] lasts 5 days

@enduml
```



TODO: DONE *[(Header, footer) corrected on V1.2020.18]*

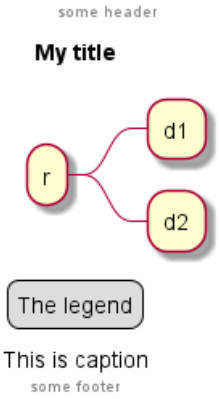
21.7.6 Object

```
@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
```

21.7.7 MindMap

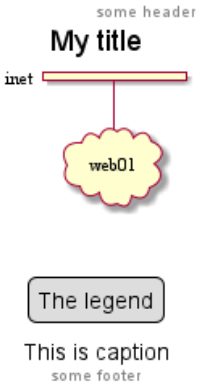
```
@startmindmap
header some header

footer some footer

title My title
```

```
caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endmindmap
```



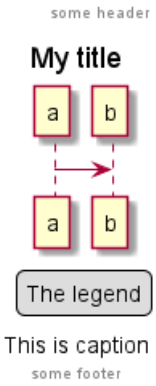
21.7.8 Network (nwdiag)

```
@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

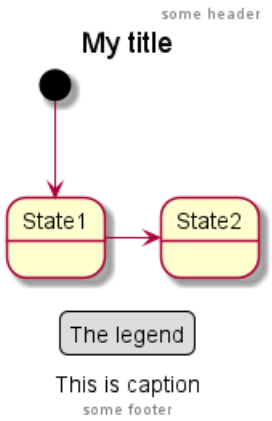
```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

a->b
@enduml
```



21.7.10 State

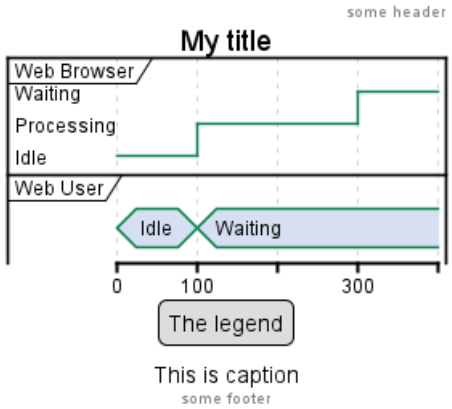
```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend
```

```
[*] --> State1
State1 -> State2

@enduml
```



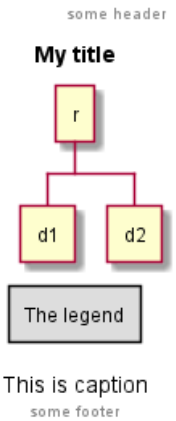
21.7.11 Timing

```
@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
```



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
```



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
```

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:

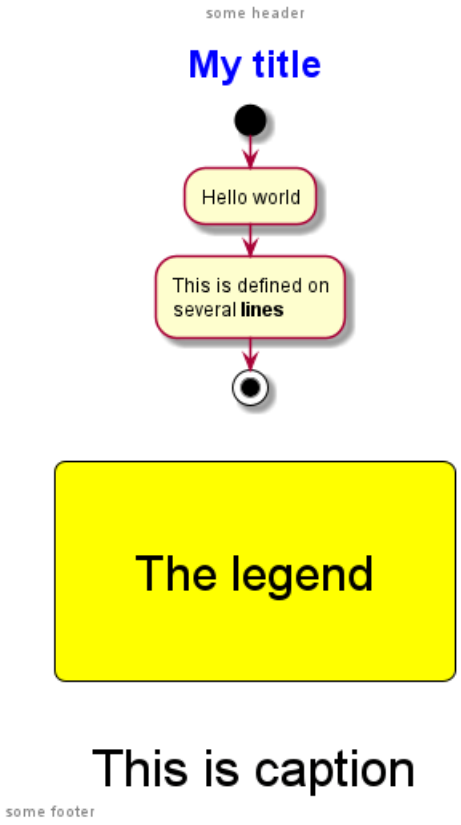
```
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

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
```

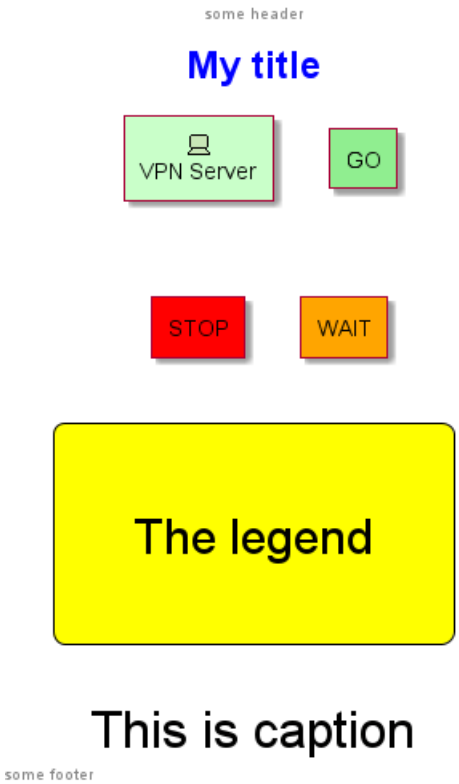
```
start
:Hello world;
:This is defined on
several **lines**;
stop

@enduml
```



21.8.2 Archimate

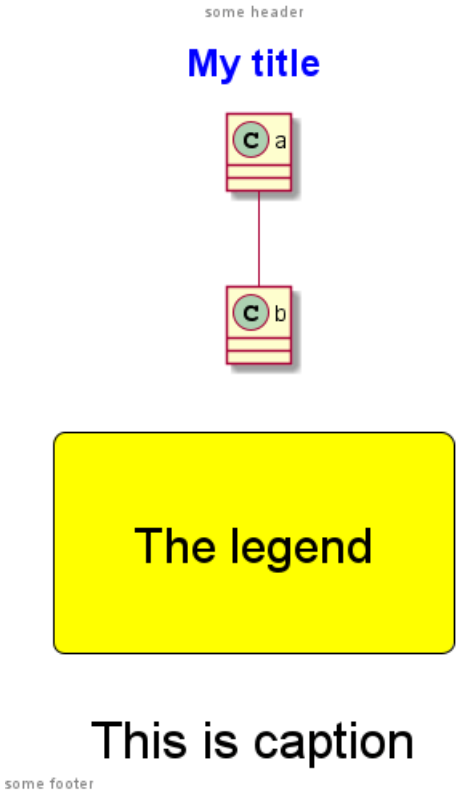
```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

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
```



21.8.3 Class

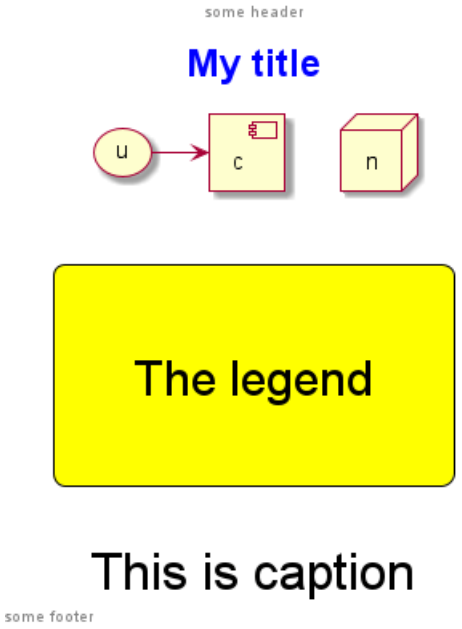
```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

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
}
```

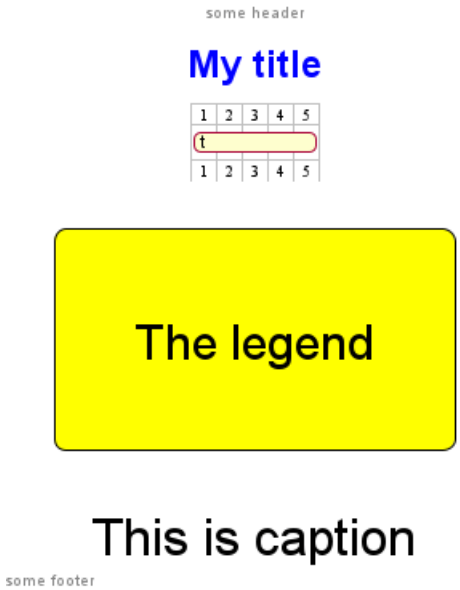
```
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

node n
(u) -> [c]

@enduml
```



21.8.5 Gantt project planning

```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
```

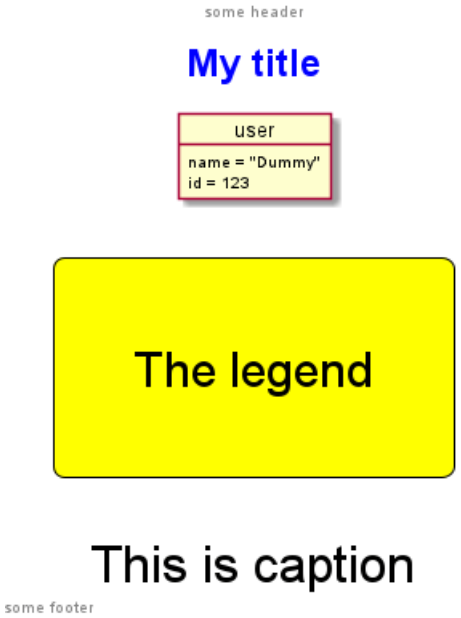
```
}

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


[t] lasts 5 days

@enduml
```



21.8.6 Object

```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}
```

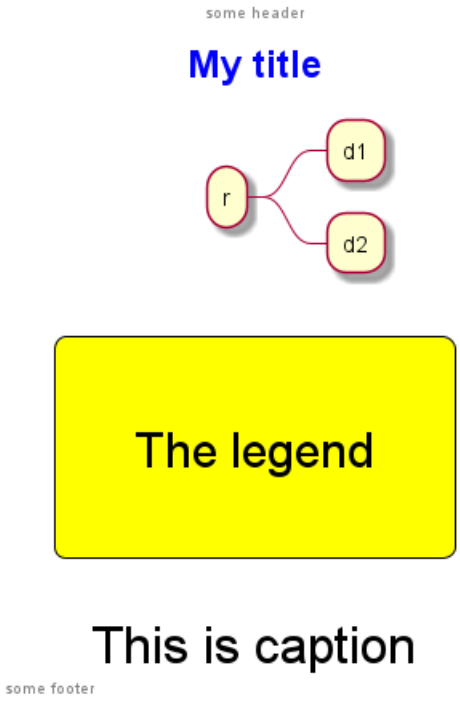
```
header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

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
```



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
```

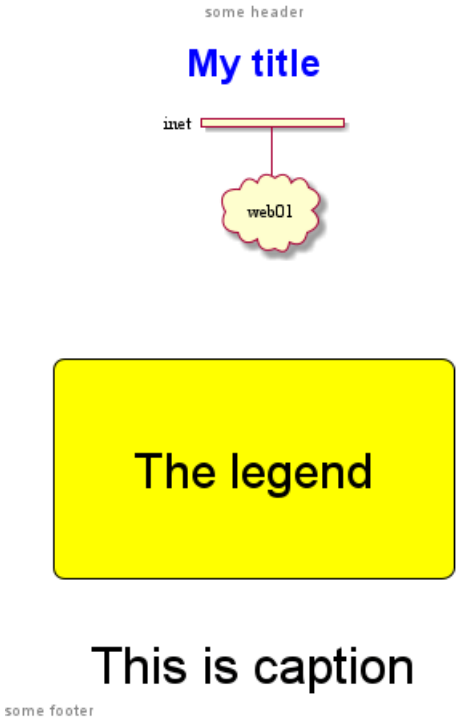
```
caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endmindmap
```



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
}
```

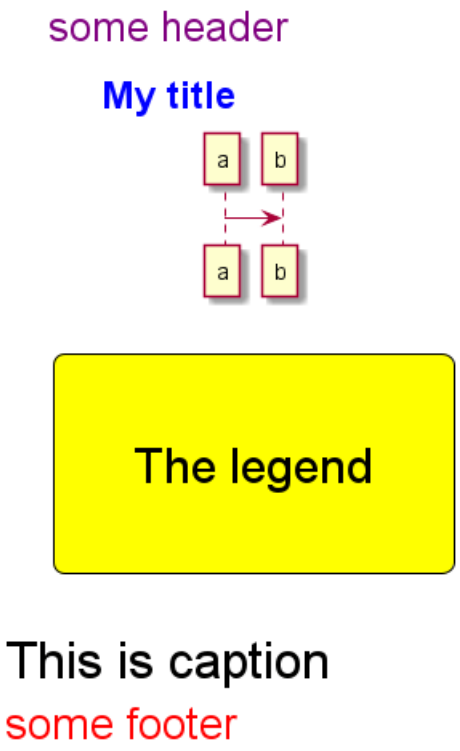
```
legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

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 {
```

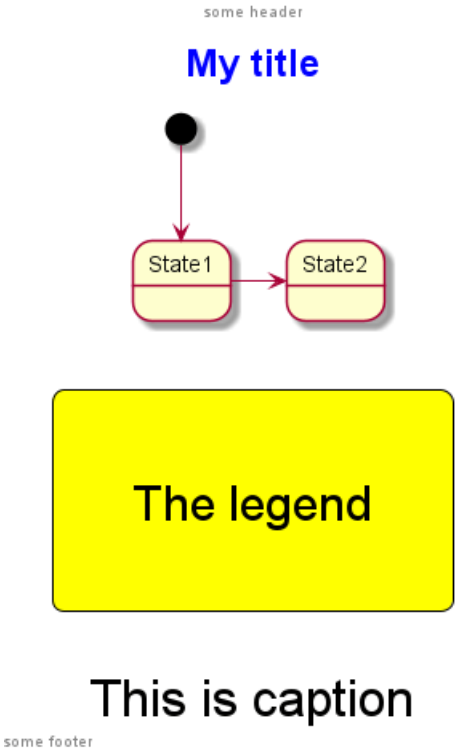
```
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

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
```



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>
```

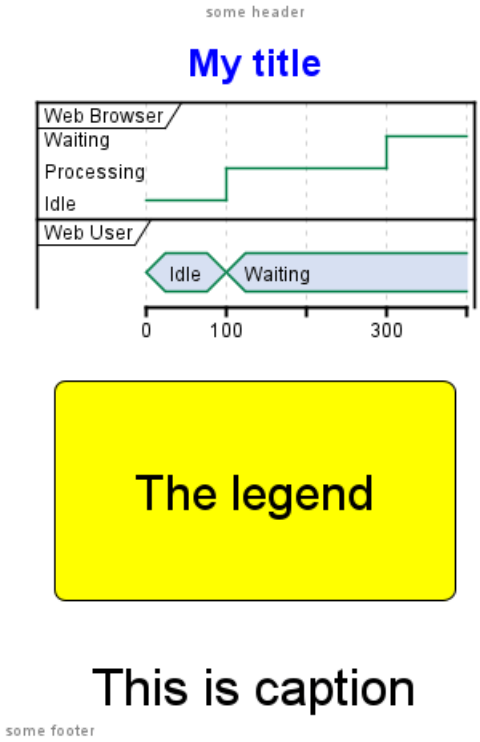
```
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

[*] --> State1
State1 -> State2

@enduml
```



21.8.11 Timing

```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
```

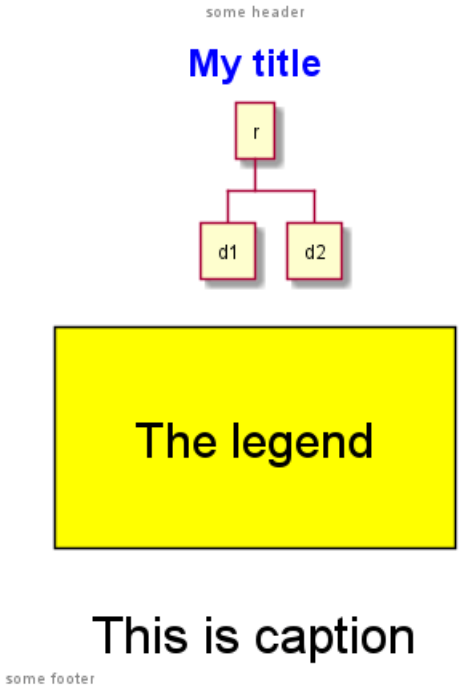
```
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

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
```



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
```



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
```

22 Créole

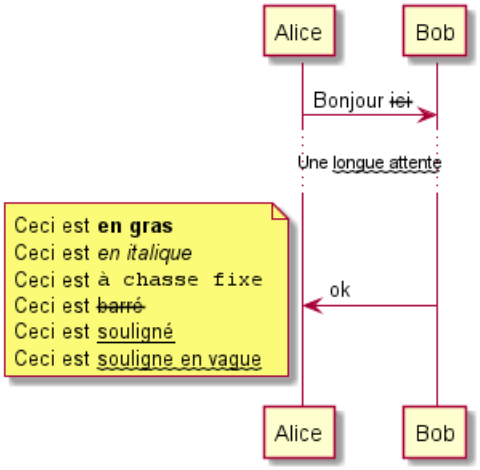
Un petit moteur Créole ou Wikicréole a été intégré à PlantUML pour pouvoir formater les textes de façon standardisé.

Tous les diagrammes intègrent cette syntaxe.

Notez qu'une compatibilité ascendante avec la syntaxe HTML a été conservée. Note that compatibility with HTML syntax is preserved.

22.1 Formatage de texte

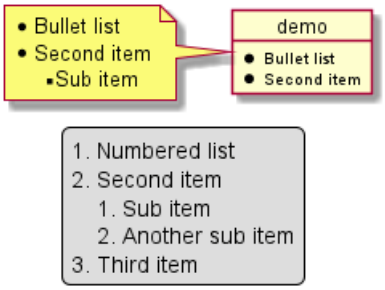
```
@startuml
Alice -> Bob : Bonjour --ici--
... Une ~~longue attente~~ ...
Bob -> Alice : ok
note left
  Ceci est **en gras**
  Ceci est //en italique//
  Ceci est ""à chasse fixe""
  Ceci est --barré--
  Ceci est __souligné__
  Ceci est ~~souligne en vague~~
end note
@enduml
```



22.2 Listes

```
@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
```

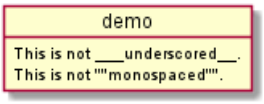
```
  # Third item
end legend
@enduml
```



22.3 Caractère d'échappement

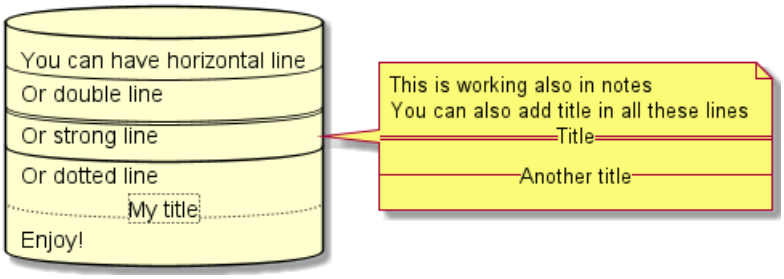
Vous pouvez utiliser le tilde ~ pour échapper les caractères Créoles spéciaux.

```
@startuml
object demo {
  This is not ~___underscored__.
  This is not ~""monospaced"".
}
@enduml
```



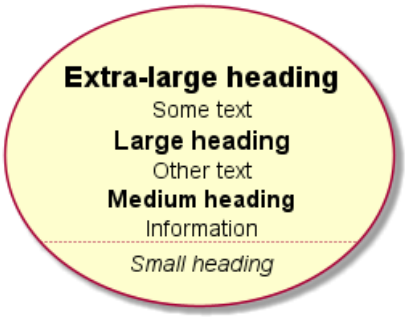
22.4 Lignes horizontales

```
@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
```



22.5 Entêtes

```
@startuml
usecase UC1 as "
= Extra-large heading
Some text
== Large heading
Other text
=== Medium heading
Information
....
==== Small heading"
@enduml
```



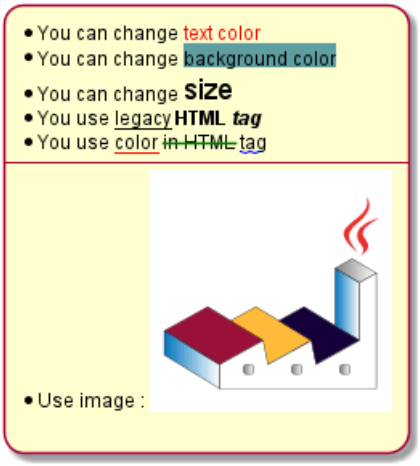
22.6 Tag HTML

Certains tag HTML sont encore fonctionnels:

- `<b>` pour du texte en gras
- `<u>` ou `<u:#AAAAAA>` ou `<u:[[color|colorName]]>` pour souligner
- `<i>` pour de l'italique
- `<s>` ou `<s:#AAAAAA>` ou `<s:[[color|colorName]]>` pour barrer du texte
- `<w>` ou `<w:#AAAAAA>` ou `<w:[[color|colorName]]>` pour souligner en vague
- `<color:#AAAAAA>` ou `<color:[[color|colorName]]>` pour la couleur
- `<back:#AAAAAA>` ou `<back:[[color|colorName]]>` pour la couleur de fond
- `<size:nn>` pour changer la taille des caractères
- `<img:file>` : le fichier doit être accessible sur le système de fichier
- `<img:http://plantuml.com/logo3.png>` : l'URL doit être accessible

```
@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>
```

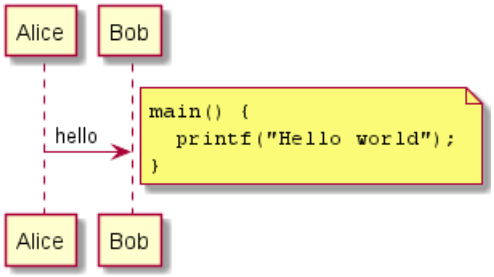
```
* You use <u:red>color</u> <s:green>in HTML</s> <w:#0000FF>tag</w>
----
* Use image : <img:http://plantuml.com/logo3.png>
;
@enduml
```



22.7 Code

You can use <code> to display some programming code in your diagram (sorry, syntax highlighting is not yet supported).

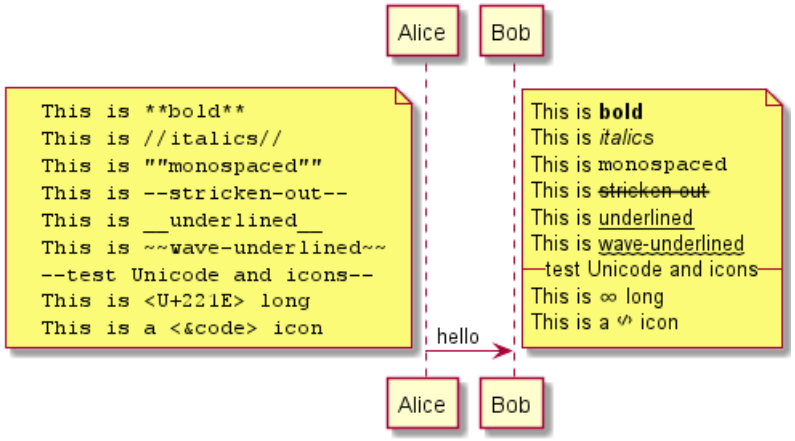
```
@startuml
Alice -> Bob : hello
note right
<code>
main() {
  printf("Hello world");
}
</code>
end note
@enduml
```



This is especially useful to illustrate some PlantUML code and the resulting rendering:

```
@startuml
Alice -> Bob : hello
note left
<code>
  This is **bold**
  This is //italics//
  This is ""monospaced""
  This is --stricken-out--
  This is __underlined__
  This is ~~wave-underlined~~
```

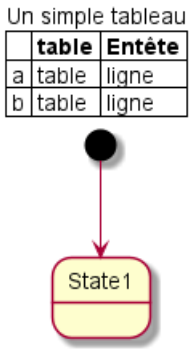
```
  --test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
</code>
end note
note right
  This is **bold**
  This is //italics//
  This is ""monospaced""
  This is --stricken-out--
  This is __underlined__
  This is ~~wave-underlined~~
  --test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
end note
@enduml
```



22.8 Tableau

Il est possible de construire des tableaux.

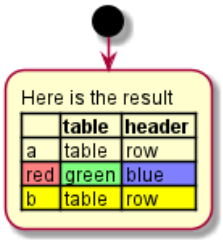
```
@startuml
skinparam titleFontSize 14
title
  Un simple tableau
  |= |= table |= Entête |
  | a | table | ligne |
  | b | table | ligne |
end title
[*] --> State1
@enduml
```



22.8.1 Add color on cells or lines

Il est possible de changer la couleur de fond des cellules et des lignes.

```
@startuml
start
:Here is the result
|= |= table |= header |
| a | table | row |
|<#FF8080> red |<#80FF80> green |<#8080FF> blue |
<#yellow>| b | table | row |;
@enduml
```



[Ref. QA-7184]

22.8.2 Add color on border

You can also specify background colors and colors for border.

```
@startuml
title
<#lightblue,#red>|=  Step  |=  Date  |=  Name  |=  Status  |=  Link  |
<#lightgreen>|  1.1  |  TBD  | plantuml news |<#Navy><color:OrangeRed><b> Unknown  | [[https://plantu
end title
@enduml
```

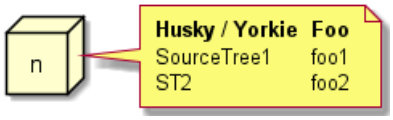
Step	Date	Name	Status	Link
1.1	TBD	plantuml news	Unknown	plantuml news

[Ref. QA-7184]

22.8.3 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
```

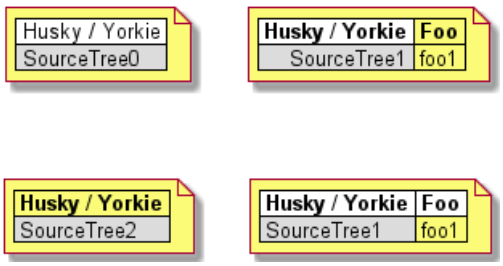


[Ref. QA-12448]

22.8.4 Bold header or not

Yan can have a bold header 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
```

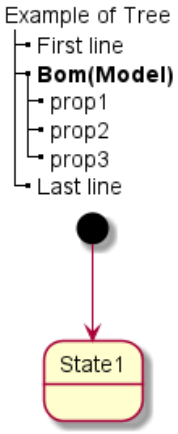


[Ref. QA-10923]

22.9 Arbre (structure arborescente)

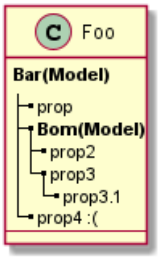
Vous pouvez utiliser les caractères suivants |_ pour construire un arbre (ou une structure arborescente).

```
@startuml
skinparam titleFontSize 14
title
  Example of Tree
  |_ First line
  |_ **Bom(Model)**
    |_ prop1
    |_ prop2
    |_ prop3
  |_ Last line
end title
[*] --> State1
@enduml
```



On Class diagram:

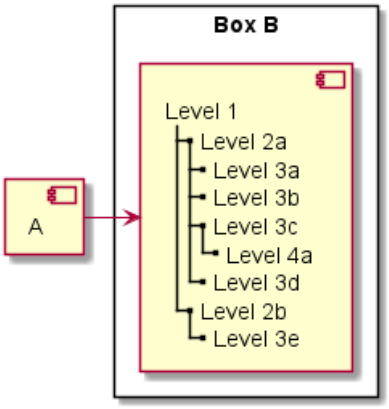
```
@startuml
class Foo{
**Bar(Model)**
|_ prop
|_ **Bom(Model)**
  |_ prop2
  |_ prop3
    |_ prop3.1
|_ prop4 :(
--
}
@enduml
```



[Ref. QA-3448]

And on *component or deployement diagram*:

```
@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]

22.10 Caractères spéciaux

Il est possible d'utiliser des caractères unicodes avec les syntaxes suivantes `&#xxxx;` ou `<U+XXXX>`

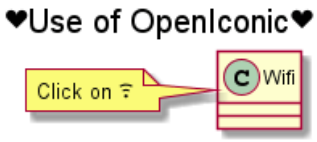
```
@startuml
usecase foo as "this is &#8734; long"
usecase bar as "this is also <U+221E> long"
@enduml
```

22.11 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
```



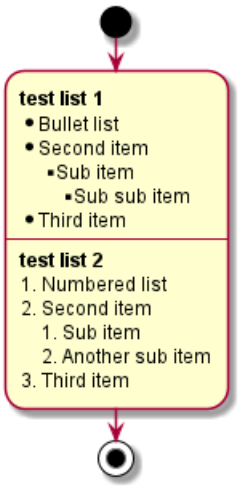
The complete list is available at the OpenIconic Website, or you can use the following special command to list them:

```
@startuml
listopeniconic
@enduml
```

22.12 Appendix: Examples of "Creole List" on all diagrams

22.12.1 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
```



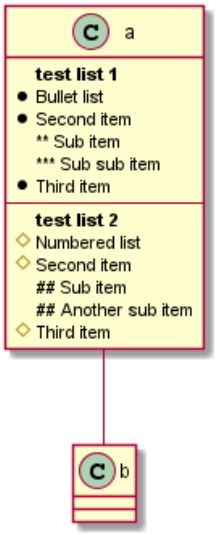
22.12.2 Class

TODO: FIXME

- *Sub item*
- *Sub sub item*

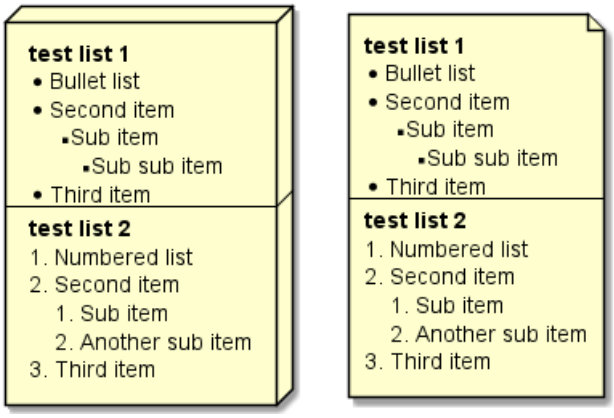
TODO: FIXME

```
@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
```



22.12.3 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]*

22.12.4 Gantt project planning

N/A

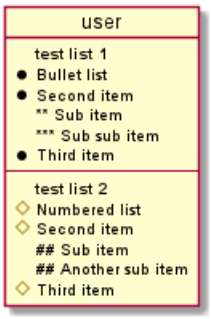
22.12.5 Object

TODO: FIXME

- *Sub item*
- *Sub sub item*

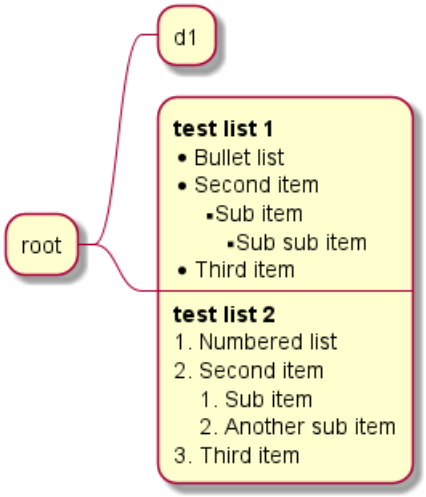
TODO: FIXME

```
@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
```



22.12.6 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
```



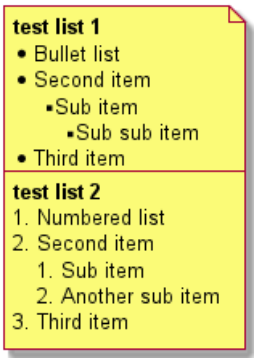
22.12.7 Network (nwdiag)

N/A

22.12.8 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
```



22.12.9 Sequence

N/A *(or on note or common commands)*

22.12.10 State

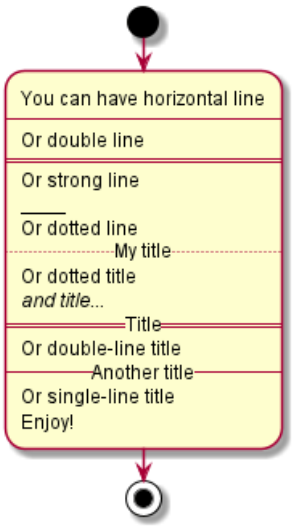
N/A *(or on note or common commands)*

22.13 Appendix: Examples of "Creole horizontal lines" on all diagrams

22.13.1 Activity

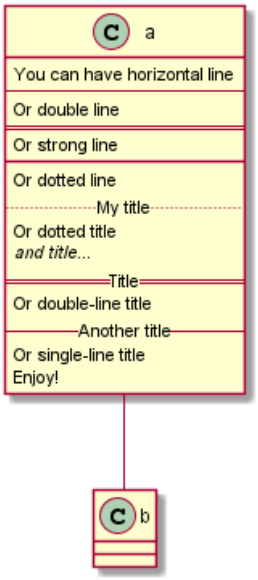
TODO: FIXME strong line ____ **TODO:** FIXME

```
@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
```



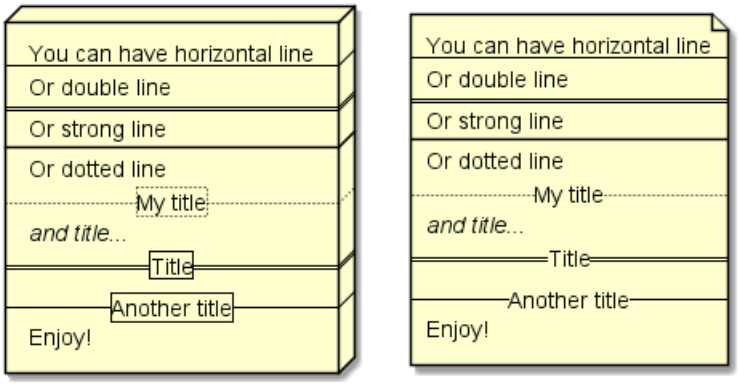
22.13.2 Class

```
@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
```



22.13.3 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!
"
@enduml
```

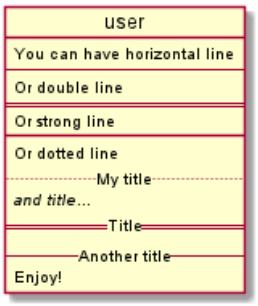


22.13.4 Gantt project planning

N/A

22.13.5 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
```



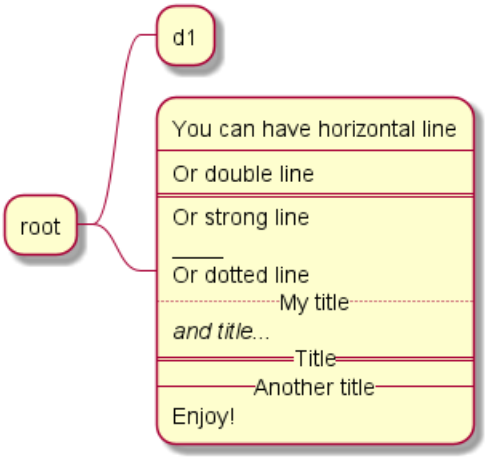
TODO: DONE *[Corrected on V1.2020.18]*

22.13.6 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
```

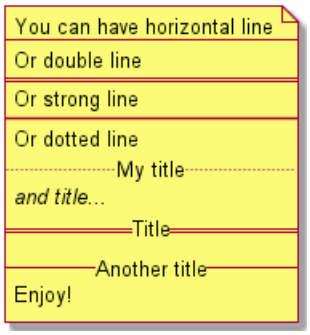


22.13.7 Network (nwdiag)

N/A

22.13.8 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
```



22.13.9 Sequence

N/A *(or on note or common commands)*

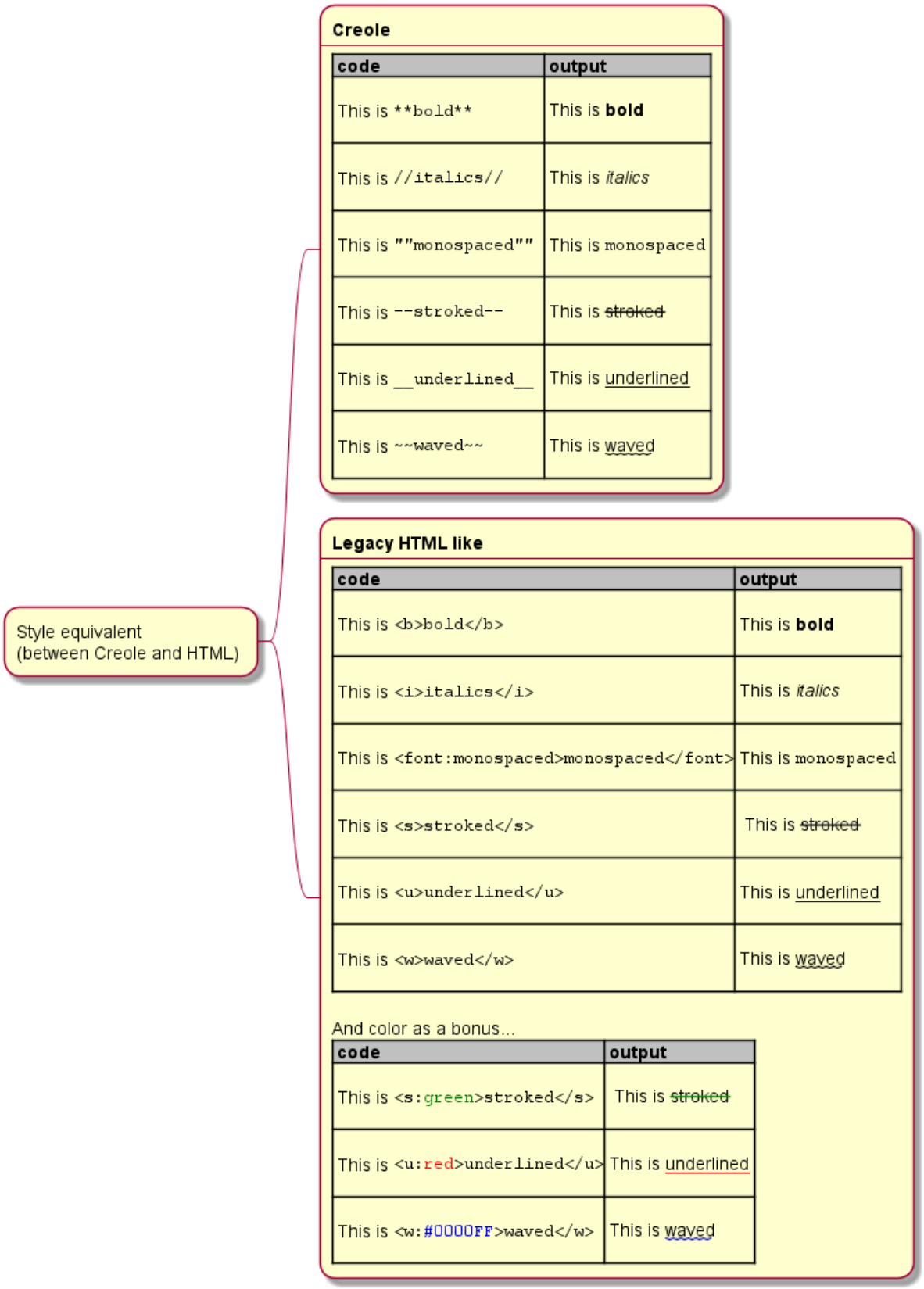
22.13.10 State

N/A *(or on note or common commands)*

22.14 Style equivalent (between Creole and HTML)

Style	Creole	Legacy HTML like
bold	This is `**bold**`	This is `<b>bold</b>`
italics	This is `//italics//`	This is `<i>italics</i>`
`monospaced`	This is `""monospaced""`	This is `<font:monospaced>monospaced</font>`
~~stroked~~	This is `--stroked--`	This is `<s>stroked</s>`
underlined	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
| \n This is ""~<u:""<color:red>""red""</color>"">underlined</u>""\n | \n This is <u:red>underlined</
| \n This is ""~<w:""<color:#0000FF>""#0000FF""</color>"">waved</w>""\n | \n This is <w:#0000FF>wave
@endmindmap
```


Style equivalent
(between Creole and HTML)
Creole
code
output
This is **bold**
This is bold
This is //italics//
This is italics
This is ""monospaced""
This is monospaced
This is --stroked--
This is stroked
This is __underlined__
This is underlined
This is ~~waved~~
This is waved
Legacy HTML like
code
output
This is bold
This is bold
This is italics
This is italics
This is <font:monospaced>monospaced
This is monospaced
This is <s>stroked</s>
This is stroked
This is <u>underlined</u>
This is underlined
This is <w>waved</w>
This is waved
And color as a bonus...
code
output
This is <s:green>stroked</s>
This is stroked
This is <u:red>underlined</u>
This is underlined
This is <w:#0000FF>waved</w>
This is waved

23 Defining and using sprites

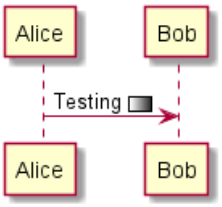
A *Sprite* is a small graphic element that can be used in diagrams.

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.

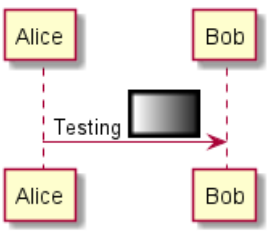
Then you can use the sprite using `<$XXX>` where XXX is the name of the sprite.

```
@startuml
sprite $foo1 {
  FFFFFFFFFFFFFFFF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  FFFFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1>
@enduml
```



You can scale the sprite.

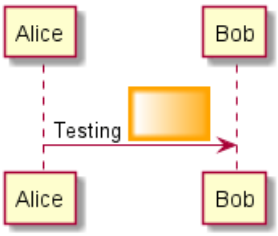
```
@startuml
sprite $foo1 {
  FFFFFFFFFFFFFFFF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  FFFFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1{scale=3}>
@enduml
```



23.1 Changing colors

Although sprites are monochrome, it's possible to change their color.

```
@startuml
sprite $foo1 {
  FFFFFFFFFFFFFFF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  FFFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1,scale=3.4,color=orange>
@enduml
```



23.2 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.

23.3 Importing Sprite

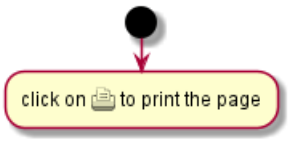
You can also launch the GUI to generate a sprite from an existing image.

Click in the menubar then on `File/Open Sprite Window`.

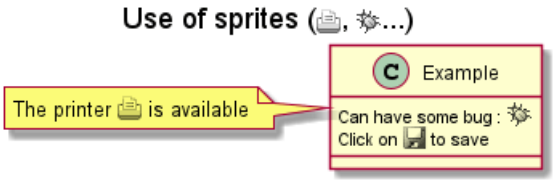
After copying an image into you clipboard, several possible definitions of the corresponding sprite will be displayed : you will just have to pickup the one you want.

23.4 Examples

```
@startuml
sprite $printer [15x15/8z] NOtH3W0W208HxFz_kMAhj7lHWpa1XC716szOPq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwvl
start
:click on <$printer> to print the page;
@enduml
```



```
@startuml
 sprite $bug [15x15/16z] PKzR2i0m2BFMi15p__FEjQEqB1z27aeqCqixa8S4OT7C53cKpsHpaYPDJY_12MHM-BLRyywPhrr
 sprite $printer [15x15/8z] NOtH3W0W208HxFz_kMAhj7lHWpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXw
 sprite $disk {
   444445566677881
   436000000009991
   43600000000ACA1
   53700000001A7A1
   53700000012B8A1
   53800000123B8A1
   63800001233C9A1
   634999AABBC99B1
   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
```



23.5 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

23.6 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

listsprites
@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:

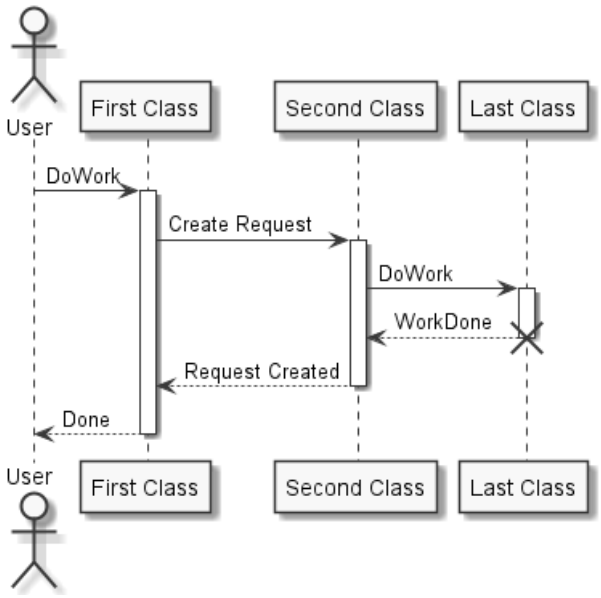
```
skinparam xxxx {
    Param1 value1
    Param2 value2
    Param3 value3
    Param4 value4
}
```

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
```

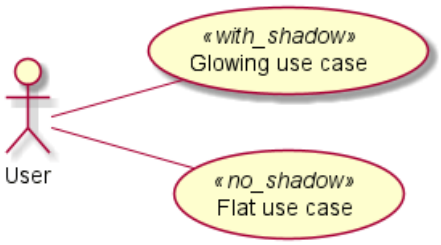
```
A --> User: Done
deactivate A

@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
```

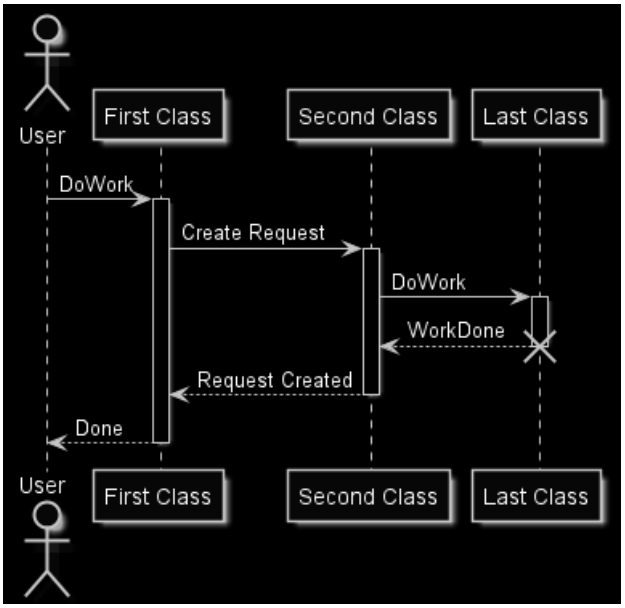


24.5 Reverse colors

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 system.

A lot of parameters are available. You can list them using the following command:

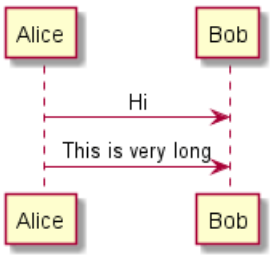
```
java -jar plantuml.jar -language
```

24.8 Text Alignment

Text alignment can be set up to `left`, `right` or `center`. You can also use `direction` or `reverseDirection` values for `sequenceMessageAlign` which align text depending on arrow direction.

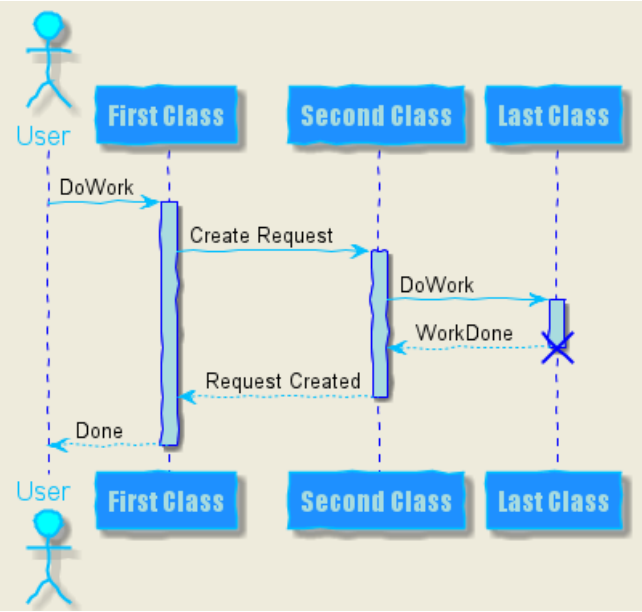
Param name	Default value	Comment
sequenceMessageAlign	left	Used for messages in sequence diagrams
sequenceReferenceAlign	center	Used for `ref over` in sequence diagrams

```
@startuml
skinparam sequenceMessageAlign center
Alice -> Bob : Hi
Alice -> Bob : This is very long
@enduml
```

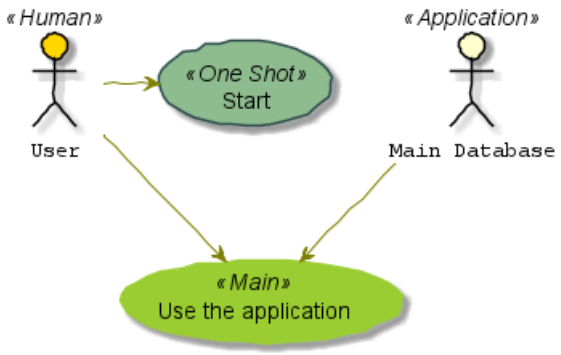


24.9 Examples

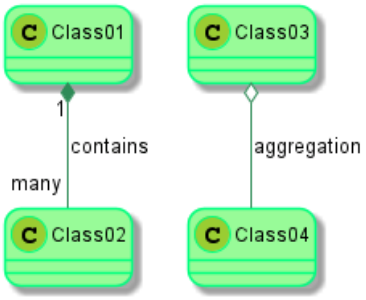
```
@startuml
skinparam backgroundColor #EEEBDC
skinparam handwritten true

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

ParticipantBorderColor DeepSkyBlue
ParticipantBackgroundColor DodgerBlue
ParticipantFontName Impact
ParticipantFontSize 17
ParticipantFontColor #A9DCDF

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
```

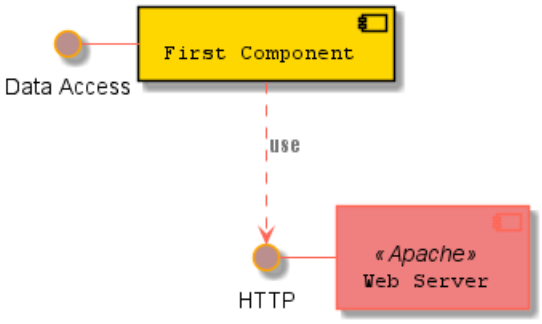



```
@startuml
skinparam roundcorner 20
skinparam class {
BackgroundColor PaleGreen
ArrowColor SeaGreen
BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen

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

Class03 o-- Class04 : aggregation
@enduml
```

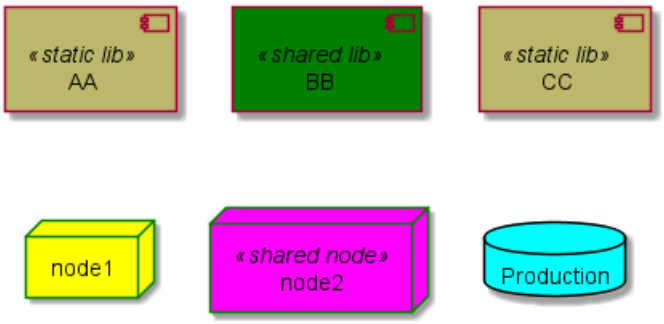



```
@startuml
skinparam interface {
  backgroundColor RosyBrown
  borderColor orange
}

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

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
```

Or you can generate a ”diagram” with a list of all the skinparam parameters using `help skinparams`.

That will give you the following result, from this page *(code of this command)*:

- CommandHelpSkinparam.java

```
@startuml
```

```
help skinparams
@enduml
```

Help on skinparam

The code of this command is located in *net.sourceforge.plantuml.help* package.
You may improve it on *https://github.com/plantuml/plantuml/tree/master/src/net/sourceforge/plantuml/help*

The possible skinparam are :

- ActivityBackgroundColor
- ActivityBarColor
- ActivityBorderColor
- ActivityBorderThickness
- ActivityDiamondBackgroundColor
- ActivityDiamondBorderColor
- ActivityDiamondFontColor
- ActivityDiamondFontName
- ActivityDiamondFontSize
- ActivityDiamondFontStyle
- ActivityEndColor
- ActivityFontColor
- ActivityFontName
- ActivityFontSize
- ActivityFontStyle
- ActivityStartColor
- ActorBackgroundColor
- ActorBorderColor
- ActorFontColor
- ActorFontName
- ActorFontSize
- ActorFontStyle
- ActorStereotypeFontColor
- ActorStereotypeFontName
- ActorStereotypeFontSize
- ActorStereotypeFontStyle
- AgentBackgroundColor
- AgentBorderColor
- AgentBorderThickness
- AgentFontColor
- AgentFontName
- AgentFontSize
- AgentFontStyle
- AgentStereotypeFontColor
- AgentStereotypeFontName
- AgentStereotypeFontSize
- AgentStereotypeFontStyle
- ArchimateBackgroundColor
- ArchimateBorderColor
- ArchimateBorderThickness
- ArchimateFontColor
- ArchimateFontName
- ArchimateFontSize
- ArchimateFontStyle
- ArchimateStereotypeFontColor
- ArchimateStereotypeFontName
- ArchimateStereotypeFontSize
- ArchimateStereotypeFontStyle
- ArrowColor
- ArrowFontColor
- ArrowFontName
- ArrowFontSize
- ArrowFontStyle
- ArrowHeadColor
- ArrowLollipopColor
- ArrowMessageAlignment
- ArrowThickness

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 Preprocesseur

Des fonctionnalités de préprocessing ont été incluses dans **PlantUML** et sont disponibles pour *tous* les diagrammes.

Ces fonctionnalités sont assez proches du préprocesseur du language C, à la différence pour le caractère `#` a été remplacé par le point d'exclamation `!`.

25.1 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.

25.2 Variable definition

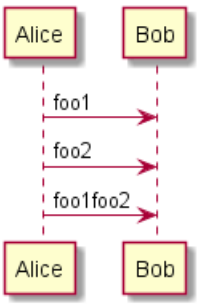
Although this is not mandatory, we highly suggest that variable names start with a `$`.

There are two types of data:

- **Integer number** *(int)*;
- **String** *(str)* - these must be surrounded by single quote or double quote.

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
!$ab = "foo1"
!$cd = "foo2"
!$ef = $ab + $cd

Alice -> Bob : $ab
Alice -> Bob : $cd
Alice -> Bob : $ef
@enduml
```



25.3 Boolean expression

25.3.1 Boolean represention [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]

25.3.2 Boolean operation and operator [&&, ||, ()]

You can use boolean expression, in the test, with :

- *parenthesis* `()`;
- *and operator* `&&`;
- *or operator* `||`.

(See next example, within `if` test.)

25.3.3 Boolean builtin functions [%false(), %true(), %not(<exp>)]

For convenience, you can use those boolean builtin functions:

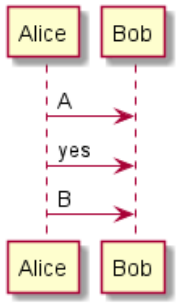
- `%false()`
- `%true()`
- `%not(<exp>)`

[See also Builtin functions]

25.4 Conditions [!if, !else, !elseif, !endif]

- You can use expression in condition.
- *else* and *elseif* are also implemented

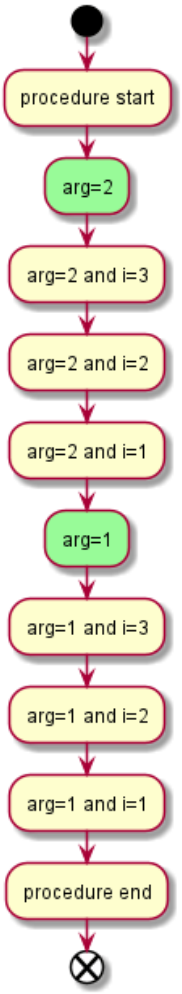
```
@startuml
!$a = 10
!$ijk = "foo"
Alice -> Bob : A
!if ($ijk == "foo") && ($a+10>=4)
Alice -> Bob : yes
!else
Alice -> Bob : This should not appear
!endif
Alice -> Bob : B
@enduml
```



25.5 While loop [!while, !endwhile]

You can use `!while` and `!endwhile` keywords to have repeat loops.

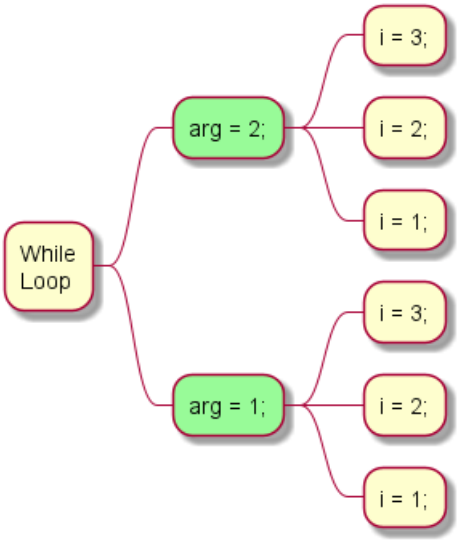
```
@startuml
!procedure $foo($arg)
  :procedure start;
  !while $arg!=0
    !$i=3
    #palegreen:arg=$arg;
    !while $i!=0
      :arg=$arg and i=$i;
      !$i = $i - 1
    !endwhile
    !$arg = $arg - 1
  !endwhile
  :procedure end;
!endprocedure

start
$foo(2)
end
@enduml
```



[Adapted from QA-10838]

```
@startmindmap
!procedure $foo($arg)
```

```
  !while $arg!=0
    !$i=3
    **[#palegreen] arg = $arg;
    !while $i!=0
      *** i = $i;
      !$i = $i - 1
    !endwhile
    !$arg = $arg - 1
  !endwhile
!endprocedure

*:While
Loop;
$foo(2)
@endmindmap
```



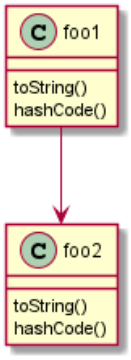
25.6 Procedure [!procedure, !endprocedure]

- 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
```



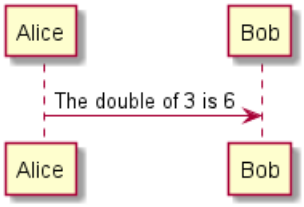
Variables defined in procedures are **local**. It means that the variable is destroyed when the procedure ends.

25.7 Return function [!function, !endfunction]

A return function does not output any text. It just define a function that you can call:

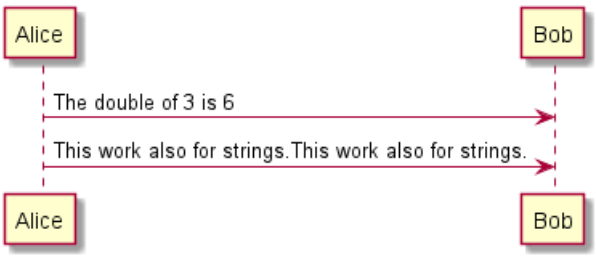
- directly in variable definition or in diagram text
- from other return functions
- from procedures
- Function name *should* start with a `$`
- Argument names *should* start with a `$`

```
@startuml
!function $double($a)
!return $a + $a
!endfunction

Alice -> Bob : The double of 3 is $double(3)
@enduml
```



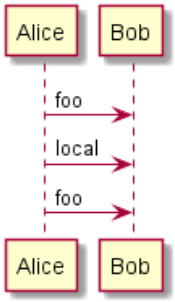
It is possible to shorten simple function definition in one line:

```
@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.

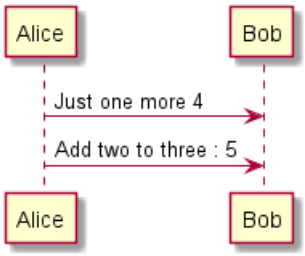
```
@startuml
!function $dummy()
!local $ijk = "local"
!return "Alice -> Bob : " + $ijk
!endfunction

!global $ijk = "foo"

Alice -> Bob : $ijk
$dummy()
Alice -> Bob : $ijk
@enduml
```



25.8 Default argument value

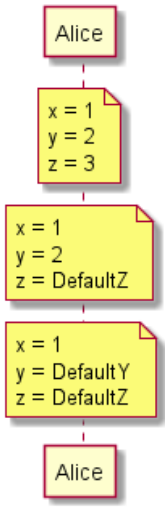
In both procedure and return functions, you can define default values for arguments.

```
@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.

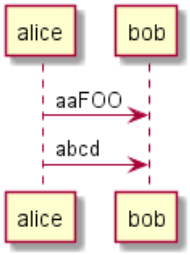
```
@startuml
!procedure defaulttest($x, $y="DefaultY", $z="DefaultZ")
note over Alice
  x = $x
  y = $y
  z = $z
end note
!endprocedure

defaulttest(1, 2, 3)
defaulttest(1, 2)
defaulttest(1)
@enduml
```



25.9 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
```

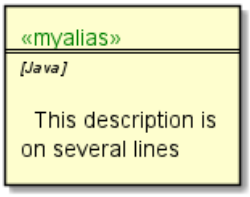


25.10 Keywords arguments

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
```

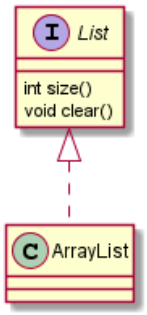


25.11 Including files or URL [!include, !include_many, !include_once]

Use the `!include` directive to include file in your diagram. Using URL, you can also include file from Internet/Intranet.

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.

25.12 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
```

25.13 Builtin functions [%]

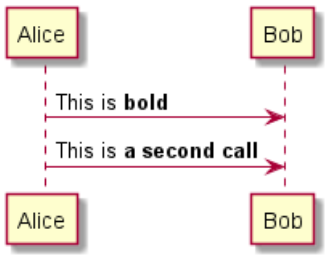
Some functions are defined by default. Their name starts by `%`

Name	Description	Example
%date	Retrieve current date. You can provide an optional format for the date	%date("yyyy.MM.d
%dirpath	Retrieve current dirpath	%dirpath()
%false	Return always false	%false()
%file_exists	Check if a file exists on the local filesystem	%file_exists("c:
%filename	Retrieve current filename	%filename()
%function_exists	Check if a function exists	%function_exists
%get_variable_value	Retrieve some variable value	%get_variable_va
%getenv	Retrieve environment variable value	%getenv("OS")
%intval	Convert a String to Int	%intval("42")
%lower	Return a lowercase string	%lower("Hello")
%newline	Return a newline	%newline()
%not	Return the logical negation of an expression	%not(2+2==4)
%set_variable_value	Set a global variable	%set_variable_va
%string	Convert an expression to String	%string(1 + 2)
%strlen	Calculate the length of a String	%strlen("foo")
%strpos	Search a substring in a string	%strpos("abcdef"
%substr	Extract a substring. Takes 2 or 3 arguments	%substr("abcdef"
%true	Return always true	%true()
%upper	Return an uppercase string	%upper("Hello")
%variable_exists	Check if a variable exists	%variable_exists
%version	Return PlantUML current version	%version()

25.14 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
```

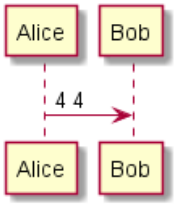


25.15 Memory dump [!memory_dump]

You can use `!memory_dump` to dump the full content of the memory when generating the diagram. An optional string can be put after `!memory_dump`. 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
```

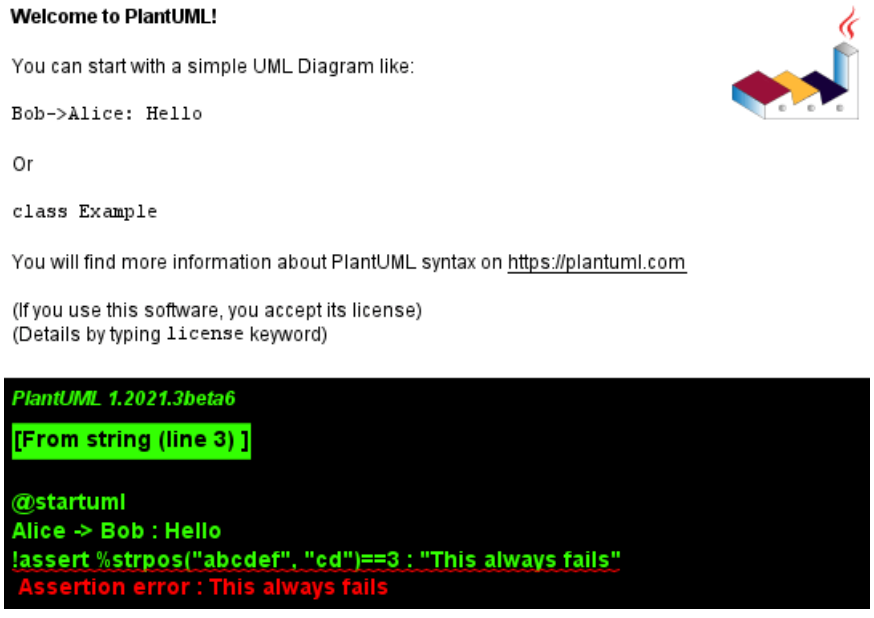
```
!dump_memory
!return $val+1
!endfunction

Alice -> Bob : 4 $inc("3")
!unused = "foo"
!dump_memory EOF
@enduml
```



25.16 Assertion [!assert]

You can put assertions in your diagram.

```
@startuml
Alice -> Bob : Hello
!assert %strpos("abcdef", "cd")==3 : "This always fails"
@enduml
```



25.17 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

...
```

25.18 Search path

You can specify the java property `plantuml.include.path` in the command line.

For example:

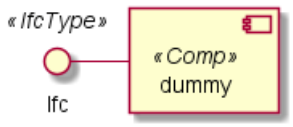
```
java -Dplantuml.include.path="c:/mydir" -jar plantuml.jar atest1.txt
```

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.

25.19 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
```

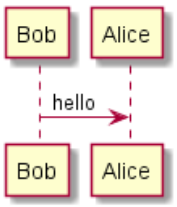


25.20 Dynamic invocation [%invoke_procedure(), %call_user_func()]

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.

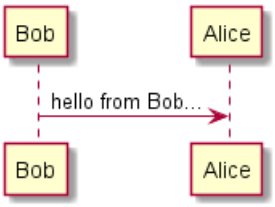
For example, you can have:

```
@startuml
!procedure $go()
  Bob -> Alice : hello
!endprocedure

!$wrapper = "$go"

%invoke_procedure($wrapper)
@enduml
```



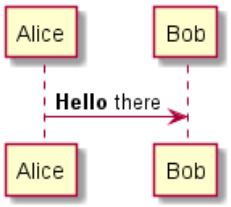

```
@startuml
!procedure $go($txt)
  Bob -> Alice : $txt
```

```
!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
```



25.21 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

25.22 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)

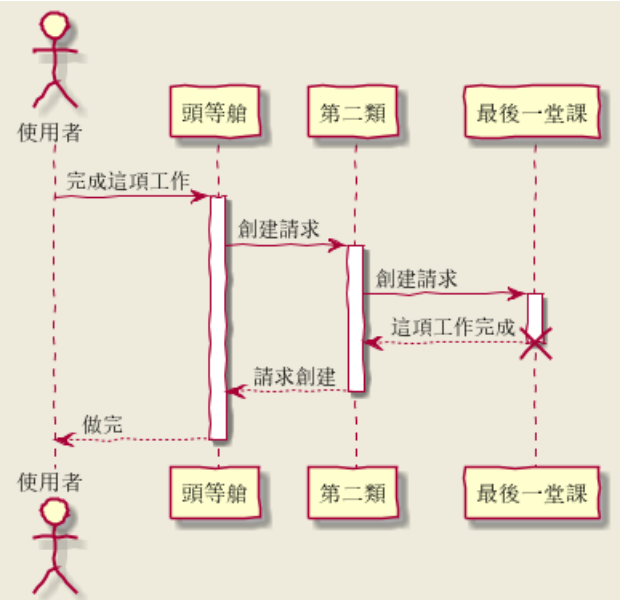
26 Unicode

The PlantUML language use *letters* to define actor, usecase and soon.

But *letters* are not only A-Z latin characters, it could be *any kind of letter from any language*.

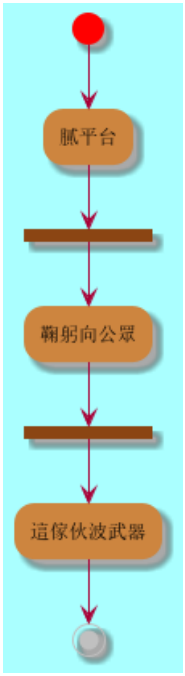
26.1 Examples

```
@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
```

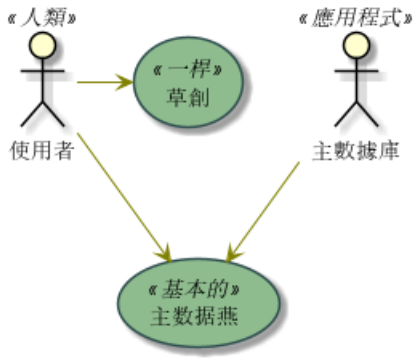



```
@startuml

(*) --> "膩平台"
--> === S1 ===
--> 鞠躬向公眾
```

```
--> === S2 ===
--> 這傢伙波武器
--> (*)

skinparam backgroundColor #AAFFFF
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
```



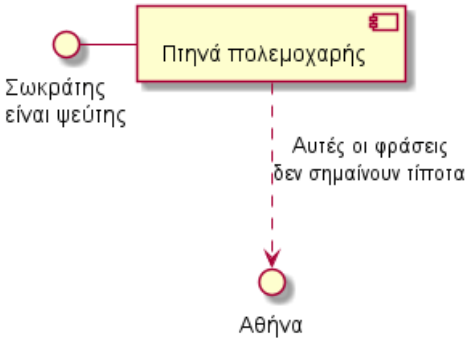
@startuml

() "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 the 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`.

27 Bibliothèque standard

Cette page détaille la *bibliothèque standard* officielle de PlantUML (`stdlib`). Cette bibliothèque est maintenant incluse dans les versions officielles de PlantUML. L'inclusion de fichier suit la convention utilisée en C pour la "Bibliothèque Standard du C".

Le contenu provenant de contributeurs externes, nous les remercions vivement pour leur travail !

27.1 List of Standard Library

You can list standard library folders using the special diagram:

```
@startuml
stdlib
@enduml
```

archimate
Version 0.0.1
Delivered by https://github.com/ebbypeter/Archimate-PlantUML

aws
Version 18.02.22
Delivered by https://github.com/milo-minderbinder/AWS-PlantUML

awslib
Version 7.0.0
Delivered by https://github.com/awslabs/aws-icons-for-plantuml

azure
Version 2.1.0
Delivered by https://github.com/RicardoNiepel/Azure-PlantUML

c4
Version 2.0.0
Delivered by https://github.com/RicardoNiepel/C4-PlantUML

cloudinsight
Version 1.0.0
Delivered by https://github.com/rabelenda/cicon-plantuml-sprites/

cloudogu
Version 1.0.2
Delivered by https://github.com/cloudogu/plantuml-cloudogu-sprites

elastic
Version 0.0.1
Delivered by https://github.com/Crashedmind/PlantUML-Elastic-icons

kubernetes
Version 5.3.45
Delivered by https://github.com/michiel/plantuml-kubernetes-sprites

logos
Version 1.0.0
Delivered by https://github.com/rabelenda/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 2.2.0
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`.

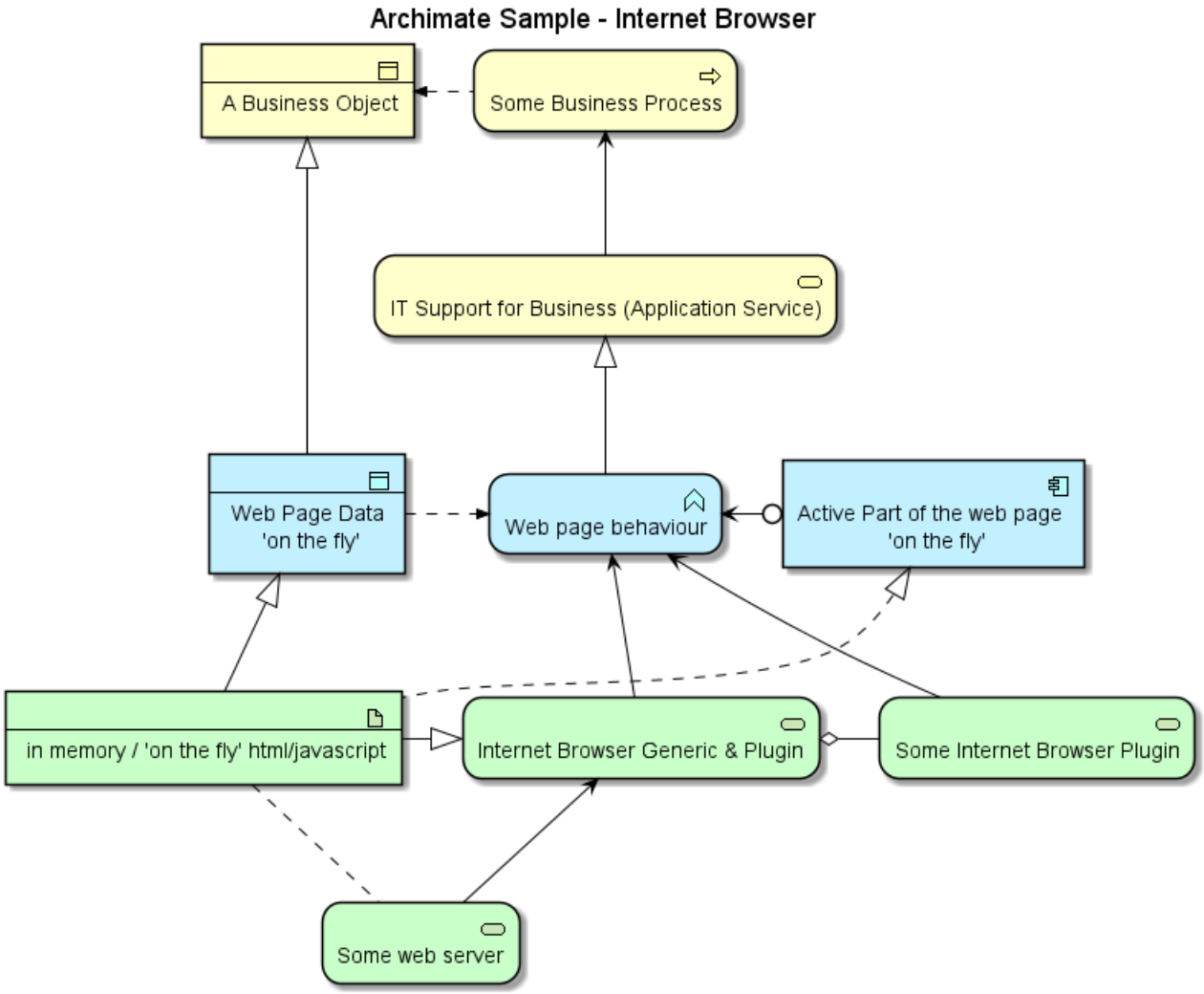
Sources used to build official PlantUML releases are hosted here https://github.com/plantuml/plantuml-stdlib. You can create Pull Request to update or add some library if you find it relevant.

27.2 ArchiMate [archimate]

- https://github.com/plantuml/plantuml-stdlib/tree/master/archimate
- https://github.com/ebbypeter/Archimate-PlantUML

This repository contains ArchiMate PlantUML macros and other includes for creating Archimate Diagrams easily and consistantly.

```
@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
```



27.3 Bibliothèque AWS

https://github.com/milo-minderbinder/AWS-PlantUML

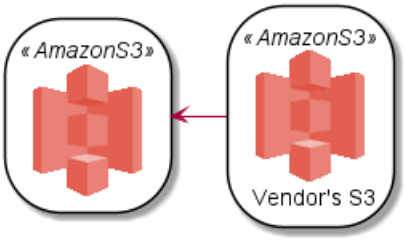
La bibliothèque AWS est composée des icônes AWS en deux tailles différentes.

Pour l'utiliser, il vous vaut inclure le ficheier qui contient le sprite (ex: `!include <aws/Storage/AmazonS3/AmazonS3>`). Une fois importé, vous pouvez utiliser le sprite normallement en l'appelant de la manière suivante `<$nom_du_sprite>`.

Vous pouvez aussi inclure le fichier `common.puml` qui contient plusieurs macros utiles avec la commande `!include <aws/common>`. Avec ce fichier importé, vous pouvez par exmeple appeler la macro "NOM_DU_SPRITE(parametres...).

Exemple d'utilisation :

```
@startuml
!include <aws/common>
!include <aws/Storage/AmazonS3/AmazonS3>
!include <aws/Storage/AmazonS3/bucket/bucket>

AMAZONS3(s3_internal)
AMAZONS3(s3_partner,"Vendor's S3")
s3_internal <- s3_partner
@enduml
```



27.4 Amazon Labs AWS Library [awslib]

- https://github.com/plantuml/plantuml-stdlib/tree/master/awslib
- https://github.com/awslabs/aws-icons-for-plantuml

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
'Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
'SPDX-License-Identifier: MIT (For details, see https://github.com/awslabs/aws-icons-for-plantuml/blo

!include <awslib/AWSCommon>

' Uncomment the following line to create simplified view
' !include <awslib/AWSSimplified>

!include <awslib/General/Users>
!include <awslib/Mobile/APIGateway>
!include <awslib/SecurityIdentityAndCompliance/Cognito>
!include <awslib/Compute/Lambda>
!include <awslib/Database/DynamoDB>

left to right direction

Users(sources, "Events", "millions of users")
APIGateway(votingAPI, "Voting API", "user votes")
Cognito(userAuth, "User Authentication", "jwt to submit votes")
Lambda(generateToken, "User Credentials", "return jwt")
Lambda(recordVote, "Record Vote", "enter or update vote per user")
DynamoDB(voteDb, "Vote Database", "one entry per user")

sources --> userAuth
sources --> votingAPI
userAuth <--> generateToken
votingAPI --> recordVote
recordVote --> voteDb
@enduml
```

27.5 Azure library [azure]

- https://github.com/plantuml/plantuml-stdlib/tree/master/azure
- https://github.com/RicardoNiepel/Azure-PlantUML/

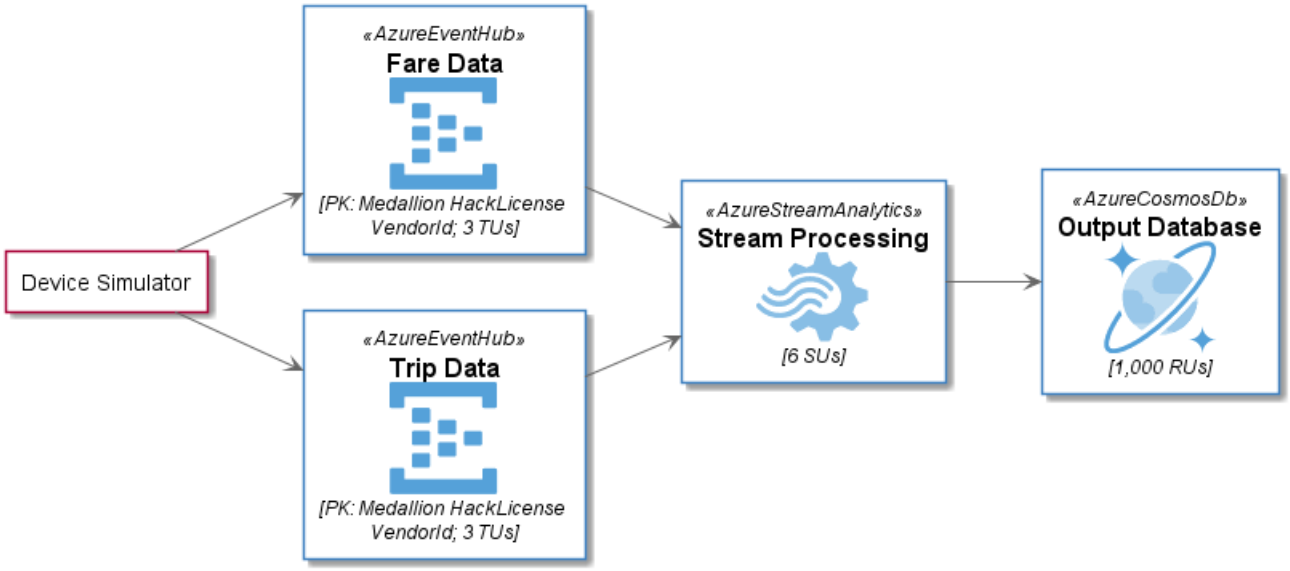
The Azure library consists of Microsoft Azure icons.

Use it by including the file that contains the sprite, eg: `!include <azure/Analytics/AzureEventHub.puml>`. 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.puml>`, 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.puml>
!include <azure/Analytics/AzureEventHub.puml>
!include <azure/Analytics/AzureStreamAnalytics.puml>
!include <azure/Databases/AzureCosmosDb.puml>

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")
AzureStreamAnalytics(streamAnalytics, "Stream Processing", "6 SUs")
AzureCosmosDb(outputCosmosDb, "Output Database", "1,000 RUs")

devices --> fareDataEventHub
devices --> tripDataEventHub
fareDataEventHub --> streamAnalytics
tripDataEventHub --> streamAnalytics
streamAnalytics --> outputCosmosDb
@enduml
```

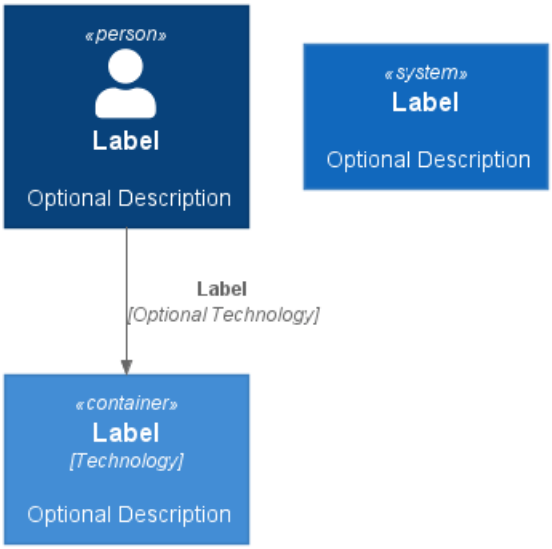


27.6 C4 Library [C4]

- https://github.com/plantuml/plantuml-stdlib/tree/master/C4
- https://github.com/plantuml-stdlib/C4-PlantUML

```
@startuml
!include <C4/C4_Container>

Person(personAlias, "Label", "Optional Description")
```

```
Container(containerAlias, "Label", "Technology", "Optional Description")
System(systemAlias, "Label", "Optional Description")

Rel(personAlias, containerAlias, "Label", "Optional Technology")
@enduml
```

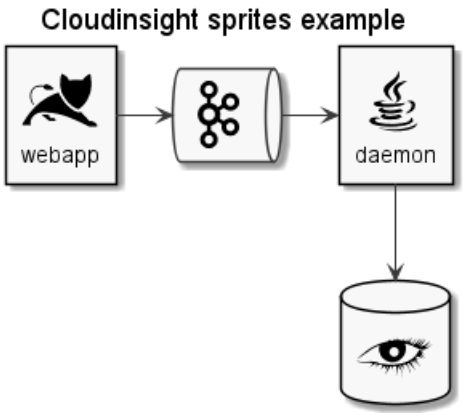


27.7 Cloud Insight [cloudinsight]

- https://github.com/plantuml/plantuml-stdlib/tree/master/cloudinsight
- https://github.com/rabelenda/cicon-plantuml-sprites

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
```

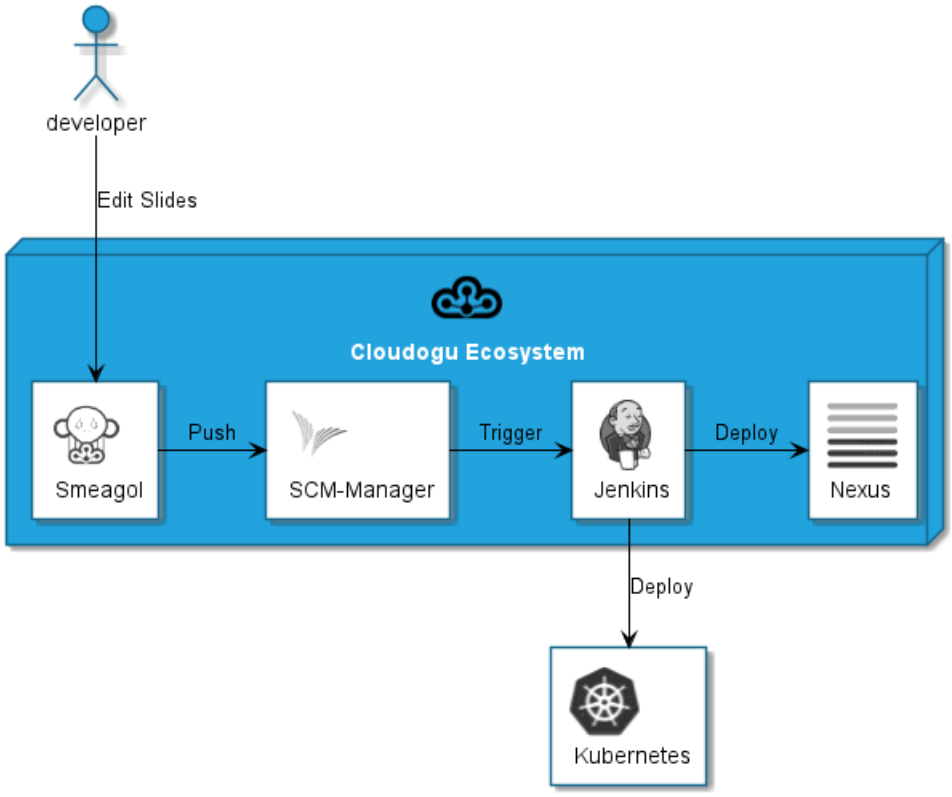


27.8 Cloudogu [cloudogu]

- https://github.com/plantuml/plantuml-stdlib/tree/master/cloudogu
- https://github.com/cloudogu/plantuml-cloudogu-sprites
- https://cloudogu.com

The Cloudogu library provides PlantUML sprites, macros, and other includes for Cloudogu services and resources.

```
@startuml
!include <cloudogu/common.puml>
!include <cloudogu/dogus/jenkins.puml>
!include <cloudogu/dogus/cloudogu.puml>
!include <cloudogu/dogus/scm.puml>
!include <cloudogu/dogus/smeagol.puml>
!include <cloudogu/dogus/nexus.puml>
!include <cloudogu/tools/k8s.puml>

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.

27.9 Elastic library [elastic]

- https://github.com/plantuml/plantuml-stdlib/tree/master/elastic
- https://github.com/Crashedmind/PlantUML-Elastic-icons

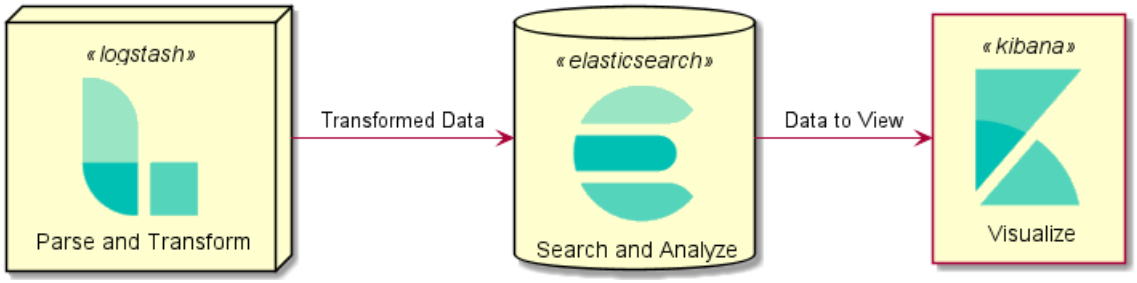
The Elastic library consists of Elastic icons. It is similar in use to the AWS and Azure libraries (it used the same tool to create them).

Use it by including the file that contains the sprite, eg: `!include elastic/elastic_search/elastic_search.puml>`. 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:

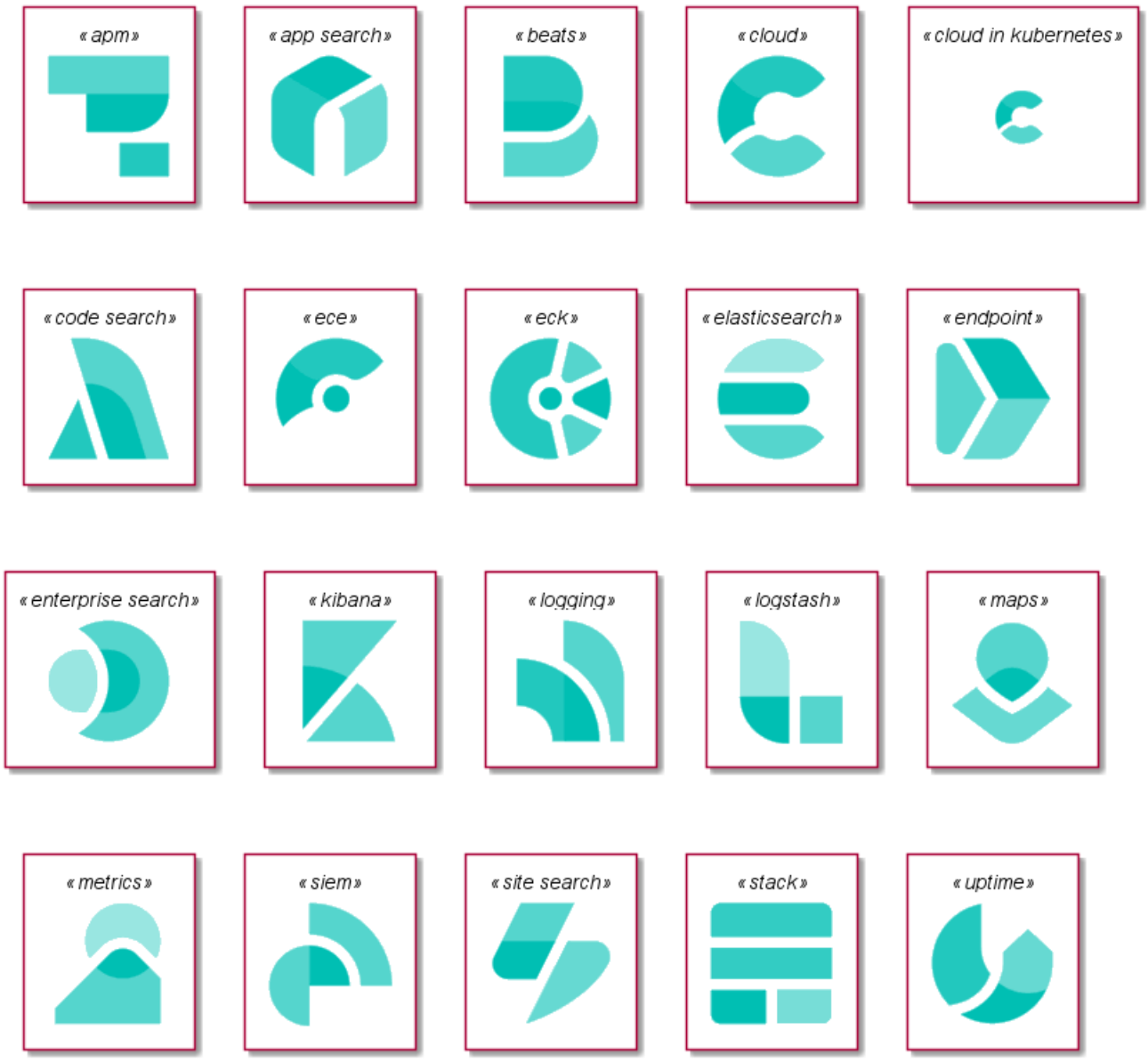
```
@startuml
!include <elastic/common>
!include <elastic/elasticsearch/elasticsearch>
!include <elastic/logstash/logstash>
!include <elastic/kibana/kibana>

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.puml>
!include <elastic/apm/apm.puml>
!include <elastic/app_search/app_search.puml>
!include <elastic/beats/beats.puml>
!include <elastic/cloud/cloud.puml>
!include <elastic/cloud_in_kubernetes/cloud_in_kubernetes.puml>
!include <elastic/code_search/code_search.puml>
!include <elastic/ece/ece.puml>
!include <elastic/eck/eck.puml>
' Beware of the difference between Crashedmind and plantuml-stdlib version: with '_' usage!
!include <elastic/elasticsearch/elasticsearch.puml>
!include <elastic/endpoint/endpoint.puml>
!include <elastic/enterprise_search/enterprise_search.puml>
!include <elastic/kibana/kibana.puml>
!include <elastic/logging/logging.puml>
!include <elastic/logstash/logstash.puml>
!include <elastic/maps/maps.puml>
!include <elastic/metrics/metrics.puml>
!include <elastic/siem/siem.puml>
!include <elastic/site_search/site_search.puml>
!include <elastic/stack/stack.puml>
!include <elastic/uptime/uptime.puml>

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
```



27.10 Google Material Icons [material]

- https://github.com/plantuml/plantuml-stdlib/tree/master/material
- https://github.com/Templarian/MaterialDesign

This library consists of a free Material style icons from Google and other artists.

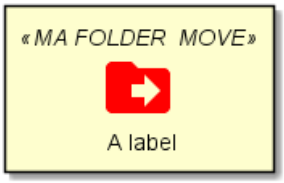
Use it by including the file that contains the sprite, eg: `!include <material/ma_folder_move>`. When imported, you can use the sprite as normally you would, using `<$ma_sprite_name>`. Notice that this library requires an `ma_` prefix on sprites names, this is to avoid clash of names if multiple sprites 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
```

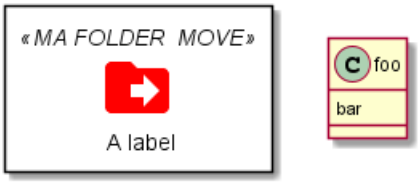


Notes:

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:

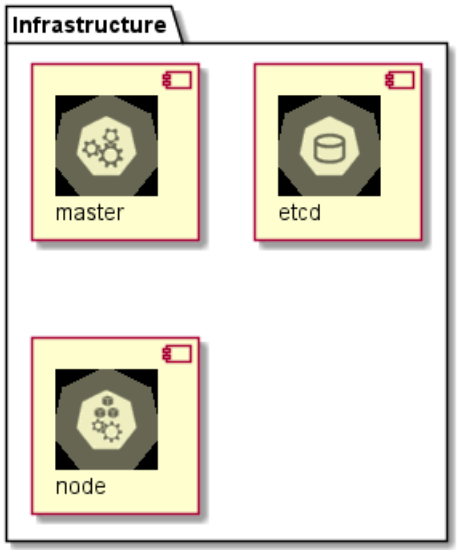
```
@startuml
!include <material/common>
' To import the sprite file you DON'T need to place a prefix!
!include <material/folder_move>

MA_FOLDER_MOVE(Red, 1, dir, rectangle, "A label") {
}

class foo {
    bar
}
@enduml
```



27.11 Kubernetes [kubernetes]

- https://github.com/plantuml/plantuml-stdlib/tree/master/kubernetes
- https://github.com/michiel/plantuml-kubernetes-sprites

```
@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
```

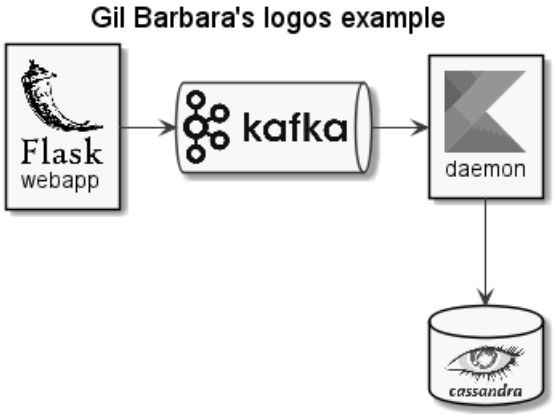


27.12 Logos [logos]

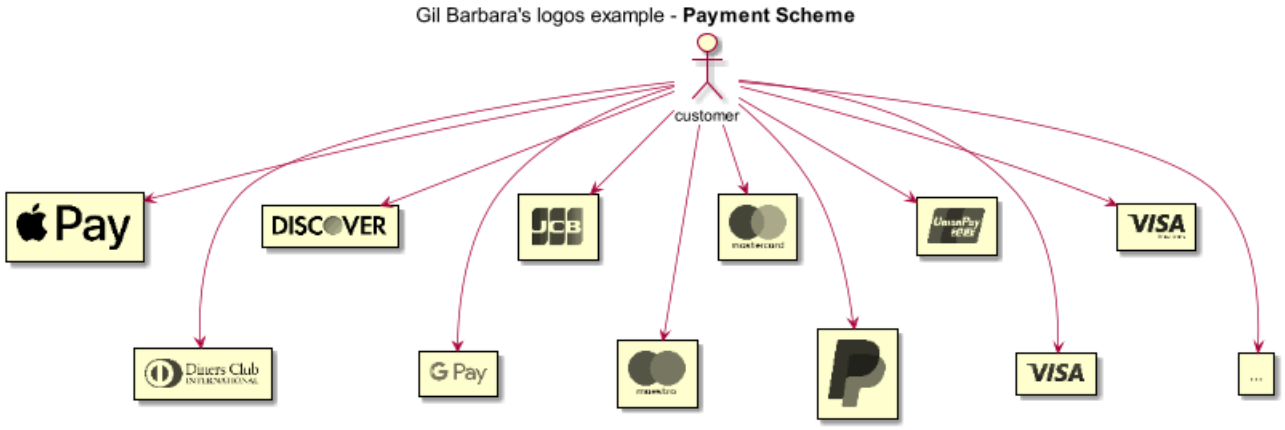
- https://github.com/plantuml/plantuml-stdlib/tree/master/logos
- https://github.com/plantuml-stdlib/gilbarbara-plantuml-sprites

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.puml>
!include <logos/kafka.puml>
!include <logos/kotlin.puml>
!include <logos/cassandra.puml>

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
```



```
@startuml
scale 0.7
!include <logos/apple-pay.puml>
!include <logos/dinersclub.puml>
!include <logos/discover.puml>
!include <logos/google-pay.puml>
!include <logos/jcb.puml>
!include <logos/maestro.puml>
!include <logos/mastercard.puml>
!include <logos/paypal.puml>
!include <logos/unionpay.puml>
!include <logos/visaelectron.puml>
!include <logos/visa.puml>
' ...

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
```



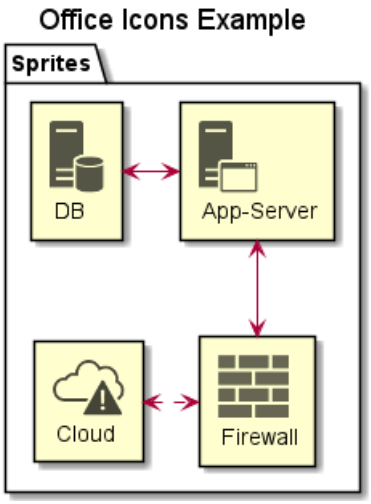
27.13 Office [office]

- https://github.com/plantuml/plantuml-stdlib/tree/master/office
- https://github.com/Roemer/plantuml-office

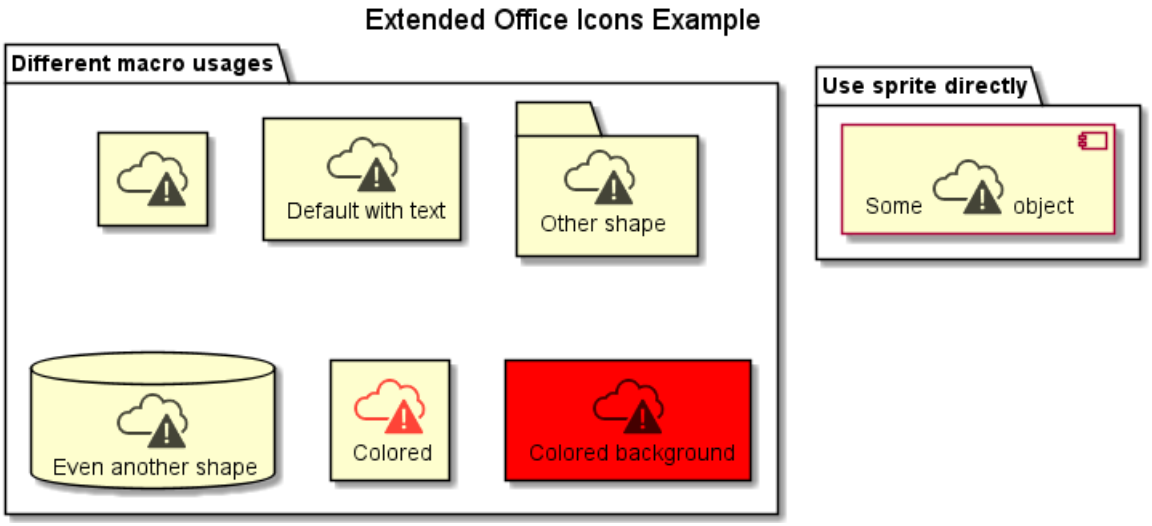
There are sprites (*.puml) and colored png icons available. Be aware that the sprites are all only monochrome even if they have a color in their name (due to automatically generating the files). You can either color the sprites with the macro (see examples below) or directly use the fully colored pngs. 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
```




```
@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
```

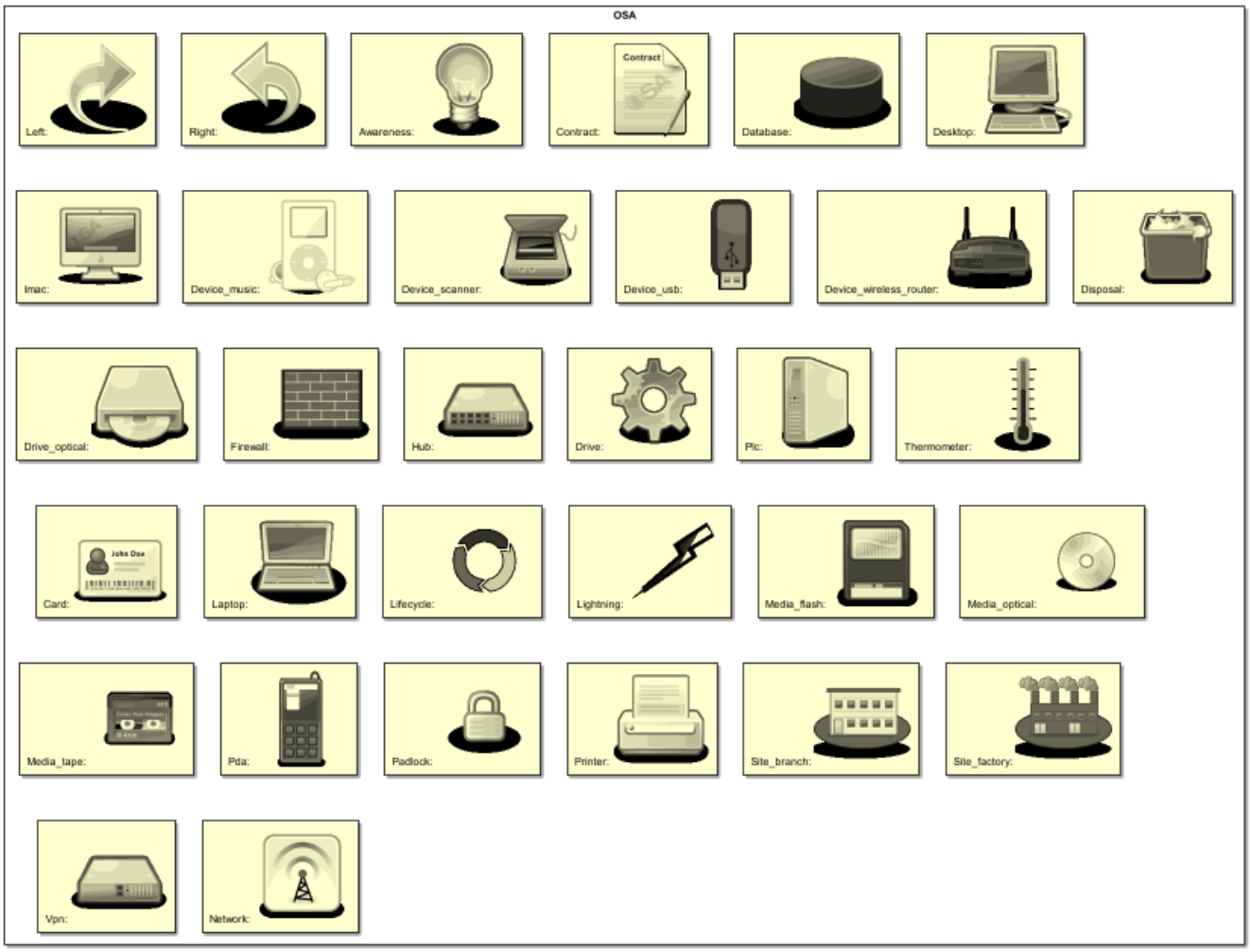


27.14 Open Security Architecture (OSA) [osa]

- https://github.com/plantuml/plantuml-stdlib/tree/master/osa
- https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons
- 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.puml>
!include <osa/arrow/yellow/right/right.puml>
!include <osa/awareness/awareness.puml>
!include <osa/contract/contract.puml>
!include <osa/database/database.puml>
!include <osa/desktop/desktop.puml>
!include <osa/desktop/imac/imac.puml>
!include <osa/device_music/device_music.puml>
!include <osa/device_scanner/device_scanner.puml>
!include <osa/device_usb/device_usb.puml>
!include <osa/device_wireless_router/device_wireless_router.puml>
!include <osa/disposal/disposal.puml>
!include <osa/drive_optical/drive_optical.puml>
!include <osa/firewall/firewall.puml>
!include <osa/hub/hub.puml>
!include <osa/ics/drive/drive.puml>
!include <osa/ics/plc/plc.puml>
!include <osa/ics/thermometer/thermometer.puml>
!include <osa/id/card/card.puml>
!include <osa/laptop/laptop.puml>
!include <osa/lifecycle/lifecycle.puml>
!include <osa/lightning/lightning.puml>
!include <osa/media_flash/media_flash.puml>
!include <osa/media_optical/media_optical.puml>
!include <osa/media_tape/media_tape.puml>
!include <osa/mobile/pda/pda.puml>
!include <osa/padlock/padlock.puml>
!include <osa/printer/printer.puml>
!include <osa/site_branch/site_branch.puml>
!include <osa/site_factory/site_factory.puml>
!include <osa/vpn/vpn.puml>
```

```
!include <osa/wireless/network/network.puml>

rectangle "OSA" {
rectangle "Left: <$left>"
rectangle "Right: <$right>"
rectangle "Awareness: <$awareness>"
rectangle "Contract: <$contract>"
rectangle "Database: <$database>"
rectangle "Desktop: <$desktop>"
rectangle "Imac: <$imac>"
rectangle "Device_music: <$device_music>"
rectangle "Device_scanner: <$device_scanner>"
rectangle "Device_usb: <$device_usb>"
rectangle "Device_wireless_router: <$device_wireless_router>"
rectangle "Disposal: <$disposal>"
rectangle "Drive_optical: <$drive_optical>"
rectangle "Firewall: <$firewall>"
rectangle "Hub: <$hub>"
rectangle "Drive: <$drive>"
rectangle "Plc: <$plc>"
rectangle "Thermometer: <$thermometer>"
rectangle "Card: <$card>"
rectangle "Laptop: <$laptop>"
rectangle "Lifecycle: <$lifecycle>"
rectangle "Lightning: <$lightning>"
rectangle "Media_flash: <$media_flash>"
rectangle "Media_optical: <$media_optical>"
rectangle "Media_tape: <$media_tape>"
rectangle "Pda: <$pda>"
rectangle "Padlock: <$padlock>"
rectangle "Printer: <$printer>"
rectangle "Site_branch: <$site_branch>"
rectangle "Site_factory: <$site_factory>"
rectangle "Vpn: <$vpn>"
rectangle "Network: <$network>"
}
@enduml
```



27.15 Tupadr3 library [tupadr3]

- https://github.com/plantuml/plantuml-stdlib/tree/master/tupadr3
- https://github.com/tupadr3/plantuml-icon-font-sprites

This library contains several libraries of icons (including Devicons and Font Awesome).

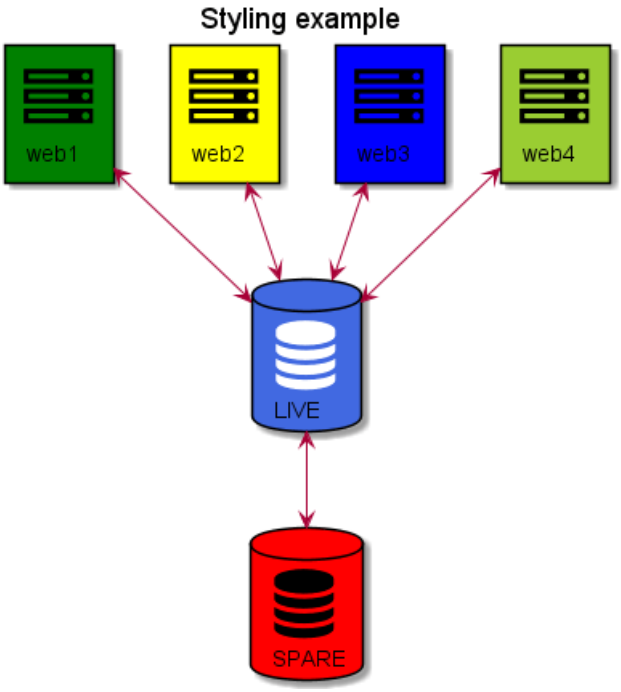
Use it by including the file that contains the sprite, eg: `!include <font-awesome/align_center>`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

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.

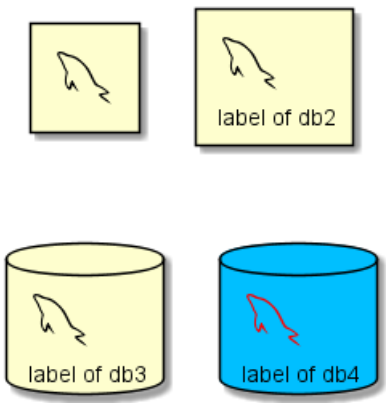
Example of usage:

```
@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
```



Contents