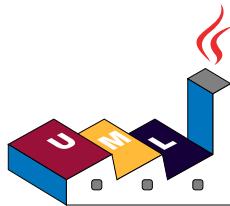


Построение диаграмм UML с использованием PlantUML



Справочное руководство по языку PlantUML

(Version 1.2025.0)

PlantUML — проект с открытым кодом, позволяющий быстро создавать:

- Диаграммы последовательности
- Диаграммы прецедентов
- Диаграммы классов
- Диаграммы объектов
- Диаграммы активности
- Диаграммы компонентов
- Диаграмма развертывания
- Диаграммы состояний
- Диаграмма синхронизации

Также поддерживаются следующие не-UML-диаграммы:

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

Для создания диаграмм применяется простой и интуитивно понятный язык.

1 Диаграммы последовательностей

Создание диаграмм последовательностей с помощью PlantUML удивительно просто. Эта простота во многом объясняется удобством синтаксиса, который разработан таким образом, чтобы быть интуитивно понятным и легко запоминающимся.

- **Интуитивно понятный синтаксис:**

Прежде всего, пользователи ценят простой и интуитивно понятный синтаксис PlantUML. Благодаря продуманному дизайну даже новички в области создания диаграмм быстро и без труда постигают его основы.

- **Соотношение текста и графики:**

Еще одной отличительной особенностью является тесное сходство между текстовым и графическим представлением. Благодаря этой гармоничной взаимосвязи текстовые наброски достаточно точно преобразуются в графические диаграммы, обеспечивая целостность и предсказуемость дизайна без неприятных сюрпризов в конечном результате.

- **Эффективный процесс разработки:**

Сильная корреляция между текстом и графическим результатом не только упрощает, но и значительно ускоряет процесс разработки. Пользователи выигрывают от более оптимизированного процесса с меньшими затратами времени на доработку и корректировку.

- **Визуализация в процессе разработки:**

Возможность представить себе конечный графический результат во время работы над текстом - функция, которую многие считают бесценной. Она способствует плавному переходу от первоначального проекта к окончательному варианту, повышая производительность и снижая вероятность ошибок.

- **Простота редактирования и внесения изменений:**

Важно отметить, что редактирование существующих диаграмм не требует особых усилий. Поскольку диаграммы создаются на основе текста, пользователи обнаруживают, что вносить в них изменения гораздо проще и точнее, чем в графические изображения. Все сводится к простому изменению текста, что гораздо проще и менее подвержено ошибкам, чем внесение изменений через графический интерфейс с помощью мыши.

PlantUML обеспечивает простой и удобный подход к созданию и редактированию диаграмм последовательности, удовлетворяя потребности как новичков, так и опытных разработчиков. Он умело использует простоту текстового ввода для создания визуально описательных и точных диаграмм, тем самым зарекомендовав себя как обязательный инструмент в наборе средств создания диаграмм.

Вы можете узнать больше о некоторых распространенных командах PlantUML, чтобы расширить свои возможности по созданию диаграмм.

1.1 Основные примеры

В диаграммах последовательности на языке PlantUML команда `->` используется для отображения сообщения между двумя участниками, которые распознаются автоматически, и не обязательно должны быть объявлены заранее.

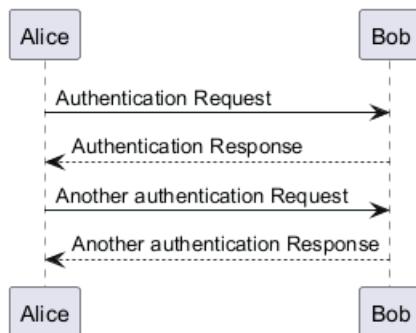
Используйте команду `-->`, чтобы изобразить пунктирную линию и повысить читаемость.

Для того, чтобы улучшить читаемость диаграммы, не меняя визуального представления, используйте команды в виде стрелок, которые направлены обратно `<-` или `<--`. Однако, обратите внимание на то, что такое утверждение об использовании этих команд справедливо только для диаграмм последовательности. Для других инструментов правила отличаются.

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

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





1.2 Объявление участников

При использовании ключевого слова `participant` возможно получить больший контроль над отображением участников.

Порядок перечисления участников задаёт также **порядок отображения** участников по умолчанию.

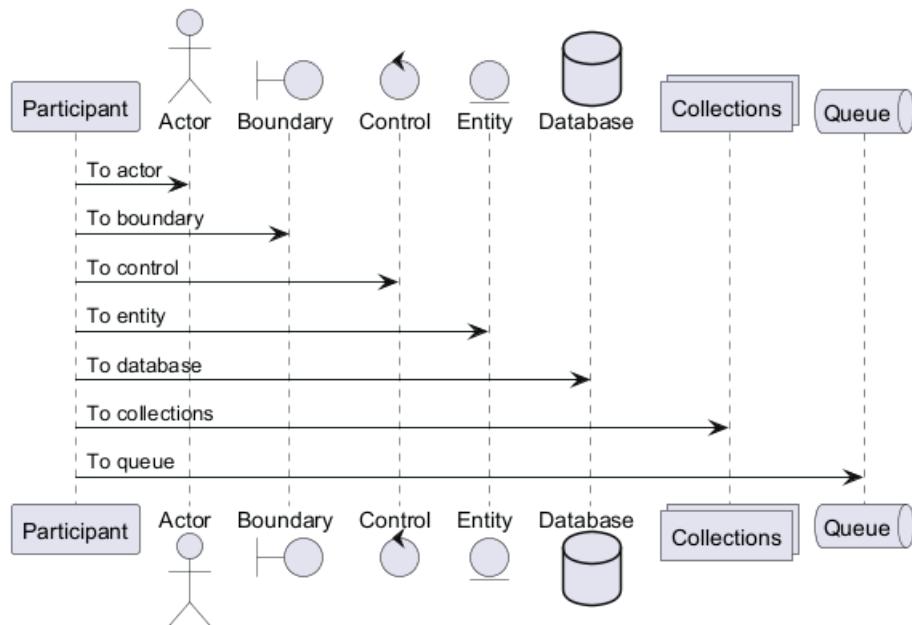
Использование других ключевых слов (отличных от `participant`) позволяет изменить **форму представления (отображения)** участника:

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

```

@startuml
participant Participant as Foo
actor      Actor      as Foo1
boundary   Boundary   as Foo2
control    Control    as Foo3
entity     Entity     as Foo4
database   Database   as Foo5
collections Collections as Foo6
queue      Queue      as Foo7
Foo -> Foo1 : To actor
Foo -> Foo2 : To boundary
Foo -> Foo3 : To control
Foo -> Foo4 : To entity
Foo -> Foo5 : To database
Foo -> Foo6 : To collections
Foo -> Foo7 : To queue
@enduml
  
```





Возможно переименовать участника используя ключевое слово `as`.

Также возможно изменить цвет фона участника.

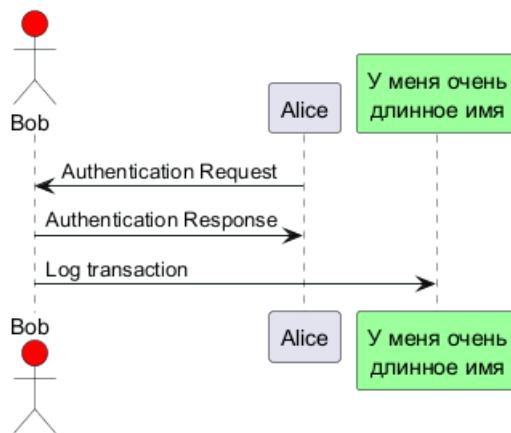
```
@startuml
actor Bob #red
'           actor
' participant -
participant Alice
participant "      \n      " as L #99FF99
'/           :
'   participant L as "      \n      " #99FF99
'/
```

Alice->Bob: Authentication Request

Bob->Alice: Authentication Response

Bob->L: Log transaction

@enduml

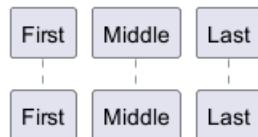


Возможно изменить порядок следования участников с помощью ключевого слова `order`.

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



@enduml



1.3 Объявление участника в многострочной форме

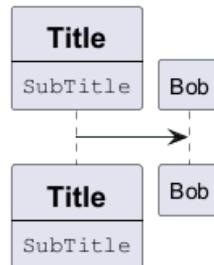
Вы можете объявить участника в многострочной форме.

```

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

participant Bob

Participant -> Bob
@enduml
  
```



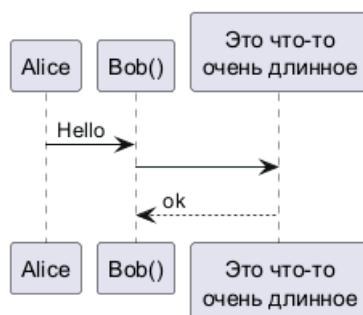
[Ссылка QA-15232]

1.4 Использование небуквенных символов в названиях участников

Вы можете использовать кавычки для задания участников. Также Вы можете использовать ключевое слово as для присвоения псевдонимов к этим участникам.

```

@startuml
Alice -> "Bob()" : Hello
"Bob()" -> "      - \n      " as Long
'
':
"Bob()" -> Long as "      - \n      "
Long --> "Bob()" : ok
@enduml
  
```



1.5 Сообщения к самому себе

Участник может посыпать сообщения сам себе.

Также возможно создание многострочных сообщений, используя \n.

```
@startuml
Alice -> Alice: .\n\n\n.
@enduml
```

```
sequenceDiagram
    participant Alice
    Alice->>Alice: Это сигнал к самому себе.  
Также это является демонстрацией  
многострочного  
текста.
```

```
@startuml
Alice <- Alice: .\n\n\n.
@enduml
```

```
sequenceDiagram
    participant Alice
    Alice-->Alice: Это сигнал к самому себе.  
Также это является демонстрацией  
многострочного  
текста.
```

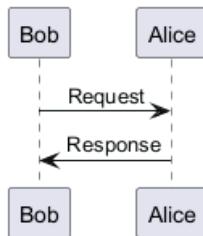
[Ref. QA-1361]

1.6 Выравнивание текста

Расположение текста относительно стрелки можно указывать как `left` (лево), `right` (право) или `center` (центр) используя слово `skinparam sequenceMessageAlign`.

Вы также можете использовать `direction` или `reverseDirection` для выравнивания текста относительно направления стрелки. Больше подробностей и примеров можно найти на странице `skinparam`.

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

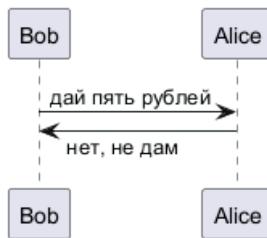


1.6.1 Текст ответного сообщения под стрелкой

Вы можете разместить текст ответного сообщения под стрелкой используя команду `skinparam responseMessageBelowArrow true`.



```
@startuml
skinparam responseMessageBelowArrow true
Bob -> Alice :
Bob <- Alice : ,
@enduml
```



1.7 Изменение стиля стрелок

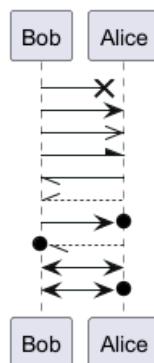
Можно изменить стиль стрелок следующими способами:

- закончить стрелку с помощью x для обозначения потерянного сообщения
- использовать \ или / вместо < или > для создания только верхней или нижней части стрелки
- повторить окончание стрелки (например, >> or //) для тонкой отрисовки
- использовать -- вместо – для создания пунктирной стрелки
- заканчивать символом ”о” во главе стрелки
- использовать двунаправленные стрелки <->

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

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

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

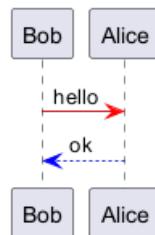


1.8 Изменить цвет стрелок

Вы можете изменить цвет отдельных стрелок, используя следующие правила:



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



1.9 Нумерация сообщений в последовательностях

Ключевое слово `autonumber` используется для автоматической нумерации сообщений.

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



Вы можете обозначить число с которого начнется отсчет `autonumber <start>`, а также число которое будет использоваться в качестве инкремента `autonumber <start> <increment>`.

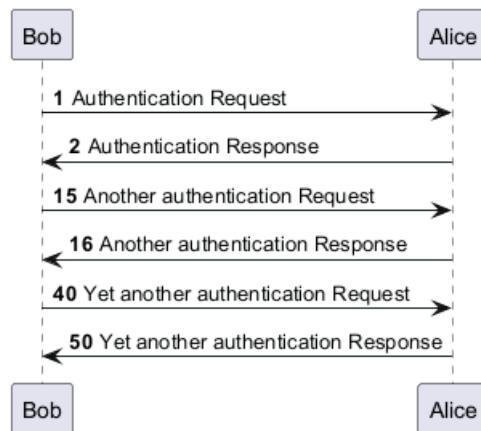
```
@startuml
autonumber
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

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
```





Можно задавать формат чисел, указав его в двойных кавычках.

Форматирование выполнено с использованием класса Java DecimalFormat (0 означает цифру, # означает цифру или ноль если пусто).

При форматировании также можно использовать теги html.

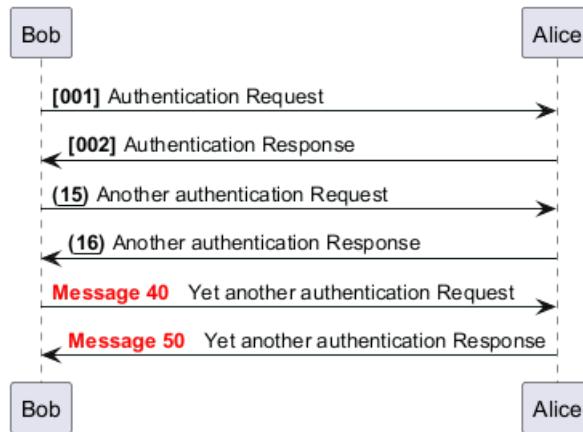
```

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

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

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

@enduml



Вы так же можете использовать autonumber stop и autonumber resume <increment> <format> чтобы соответственно остановить и продолжить автоматическое нумерование.

```

@startuml
autonumber 10 10 "<b>[000]</b>"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

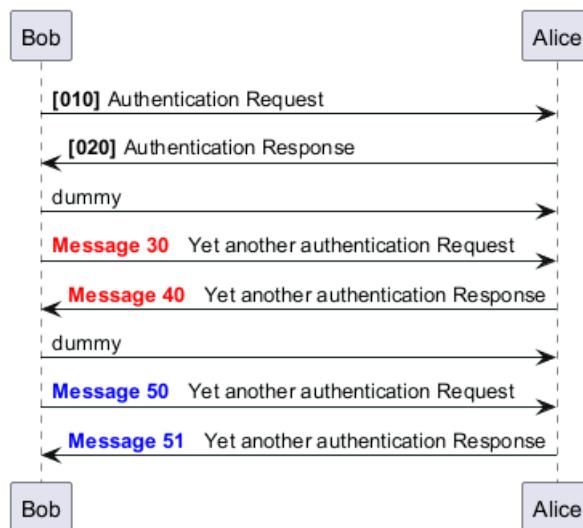
autonumber stop
Bob -> Alice : dummy
  
```



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



Ваше число, с которого начнется отсчет может быть последовательностью из двух или трех цифр, разделенных знаками ., ;, , или :. Также это могут одновременно два разных знака из этого списка. Например: 1.1.1 или 1.1:1.

Автоматически инкрементируется последняя цифра из указанной последовательности.

Для того, чтобы инкрементировать первую цифру последовательности, укажите autonumber inc A. Для инкрементирования второй цифры последовательности укажите соответственно autonumber inc B.

```
@startuml
autonumber 1.1.1
Alice -> Bob: Authentication request
Bob --> Alice: Response

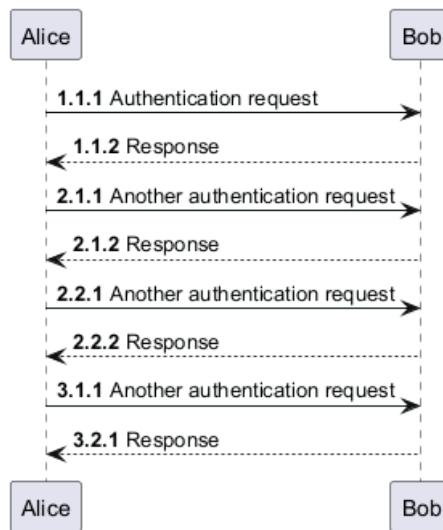
autonumber inc A
'Now we have 2.1.1
Alice -> Bob: Another authentication request
Bob --> Alice: Response

autonumber inc B
'Now we have 2.2.1
Alice -> Bob: Another authentication request
Bob --> Alice: Response

autonumber inc A
'Now we have 3.1.1
Alice -> Bob: Another authentication request
autonumber inc B
'Now we have 3.2.1
Bob --> Alice: Response
```



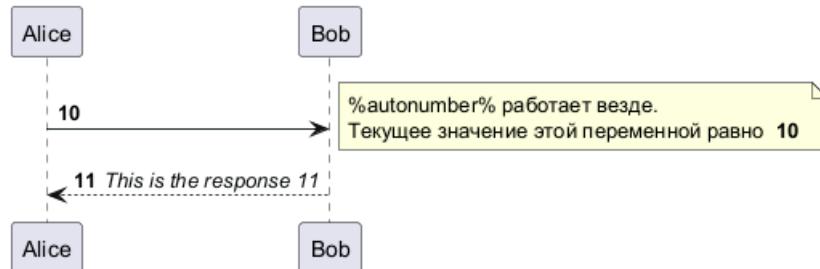
@enduml



Вы также можете использовать значение autonumber вместе с %autonumber% переменной:

```

@startuml
autonumber 10
Alice -> Bob
note right
    <U+0025>autonumber<U+0025>
    ** %autonumber% **
end note
Bob --> Alice: //This is the response %autonumber%/
@enduml
  
```



[Ref. QA-7119]

1.10 Название, Заголовок и футтер страницы

Ключевое слово title используется, чтобы добавить название страницы.

Также страницы могут иметь заголовок и футтер с помощью ключевых слов header и footer.

@startuml

```

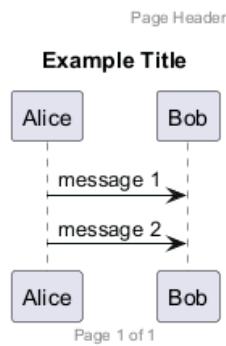
header Page Header
footer Page %page% of %lastpage%

title Example Title

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

@enduml





1.11 Разбиение диаграмм

Ключевое слово `newpage` используется для разбиения диаграмм на несколько изображений.

Вы можете указать название страницы сразу после ключевого слова `newpage`. Это название заменяет указанное ранее название, если такое имеется.

Это очень полезно для печати длинных диаграмм на нескольких страницах.

(Примечание: это действительно работает. В превью показана только первая страница, но это из-за особенностей онлайн отображения.)

`@startuml`

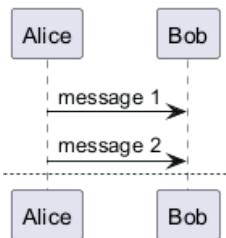
```
Alice -> Bob : message 1
Alice -> Bob : message 2
```

`newpage`

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

`newpage A title for the\last page`

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



1.12 Группировка сообщений

Группировать сообщения возможно используя следующие ключевые слова:

- `alt/else`
- `opt`
- `loop`
- `par`
- `break`
- `critical`



- group, соответствует тексту который должен быть отображен

Имеется возможность добавить текст который должен быть отображен в заголовке. (для group, смотрите следующий параграф 'Дополнительная метка группы').

Ключевое слово end используется для завершения группы.

Допустимо вложение группы в группу.

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

alt successful case

    Bob -> Alice: Authentication Accepted

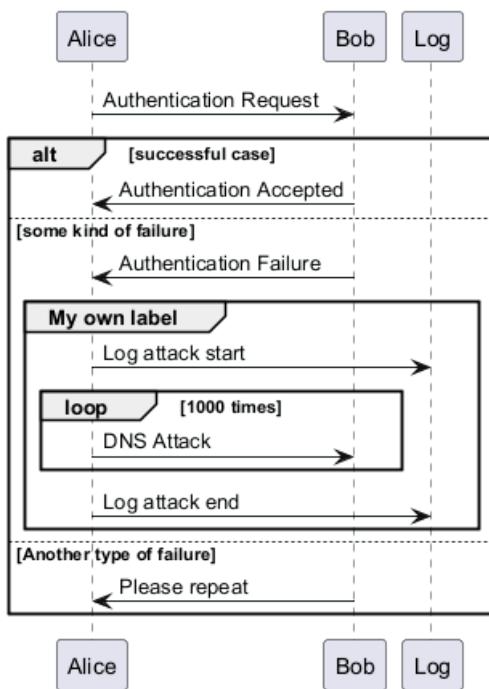
else some kind of failure

    Bob -> Alice: Authentication Failure
    group My own label
        Alice -> Log : Log attack start
        loop 1000 times
            Alice -> Bob: DNS Attack
        end
        Alice -> Log : Log attack end
    end

else Another type of failure

    Bob -> Alice: Please repeat

end
@enduml
```



1.13 Дополнительная метка группы

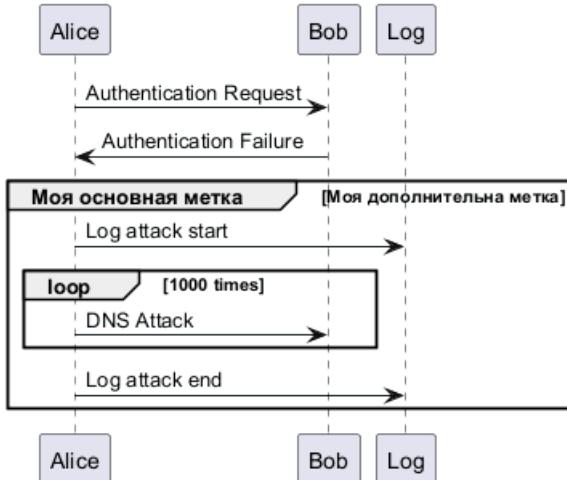
Для group можно добавить дополнительный текст или метку, которые будут отображаться в заголовке. Для этого их надо указать между квадратными скобками [и].



```

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

```



[Ссылка QA-2503]

1.14 Примечания в сообщениях

Можно помещать заметки к сообщениям, используя ключевые слова `note left` (заметка слева) или `note right` (заметка справа) *сразу после сообщения*.

Можно делать многострочные заметки используя ключевое слово `end note` для завершения заметки.

```

@startuml
Alice->Bob :
note left:
Bob->Alice :
note right:
Bob->Bob :
note left

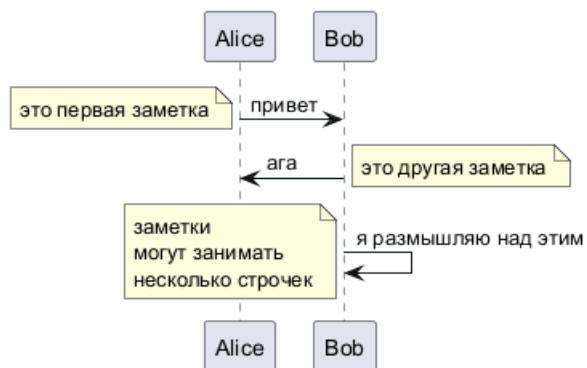
```

```

end note
@enduml

```





1.15 Другие примечания

Возможно размещение примечаний относительно участников с использованием ключевых слов `note left of` (разместить слева от), `note right of` (разместить справа от) или `note over` (разместить над).

Возможно выделить примечание, изменив цвет фона.

Можно делать многострочные заметки используя ключевое слово `end note` для завершения заметки.

```
@startuml
participant Alice
participant Bob
note left of Alice #aqua
```

```
end note

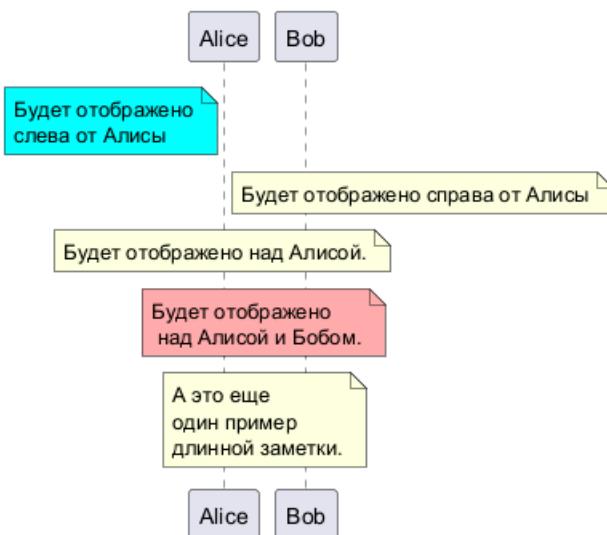
note right of Alice:

note over Alice:

note over Alice, Bob #FFAAAA: \n

note over Bob, Alice
```

```
end note
@enduml
```



1.16 Изменение формы примечаний

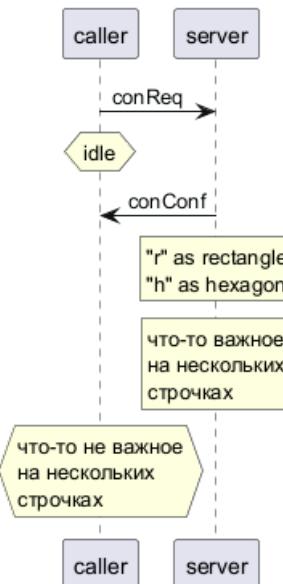
Вы можете использовать `hnote` и `rnote` для изменения формы примечаний:

- `hnote` для примечания в виде шестиугольника;
- `rnote` для примечания в виде прямоугольника.

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

```
endrnote
hnote over caller
-
```

```
endhnote
@enduml
```



[Ref. QA-1765]

1.17 Заметка отображаемая поверх всех участников

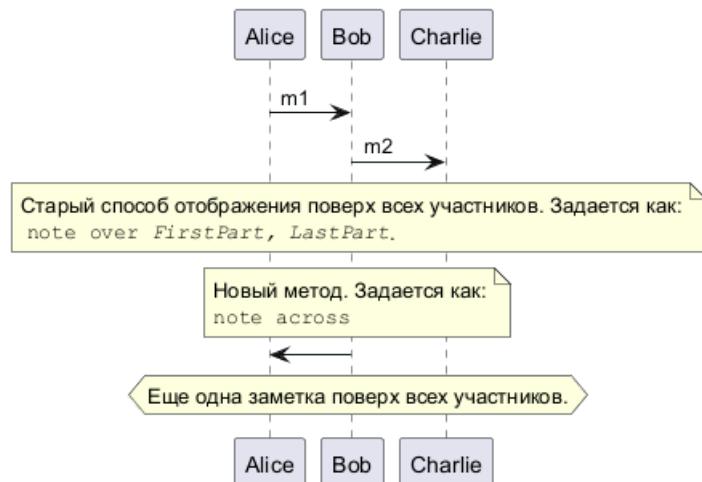
Вы можете сделать заметку, отображаемую поверх всем участников, используя синтаксис:

- `note across: note_description`

```
@startuml
Alice->Bob:m1
Bob->Charlie:m2
note over Alice, Charlie: . . . . . : \n ""note over //FirstPart, LastPart"
note across: . . . . . : \n ""note across""
Bob->Alice
```



```
hnote across:  
@enduml
```



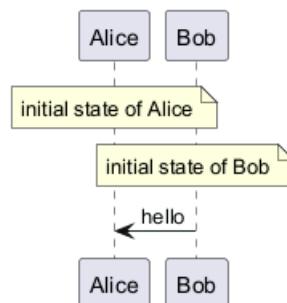
[Ref. QA-9738]

1.18 Выравнивание нескольких заметок по одному уровню [/]

Вы можете выровнить несколько заметок по одному уровню используя синтаксис `/`:

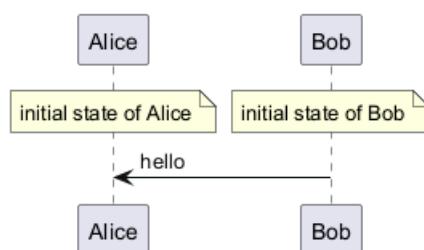
- без `/` (по умолчанию заметки не выравниваются)

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



- с `/` (заметки выровнены)

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



[Ref. QA-354]



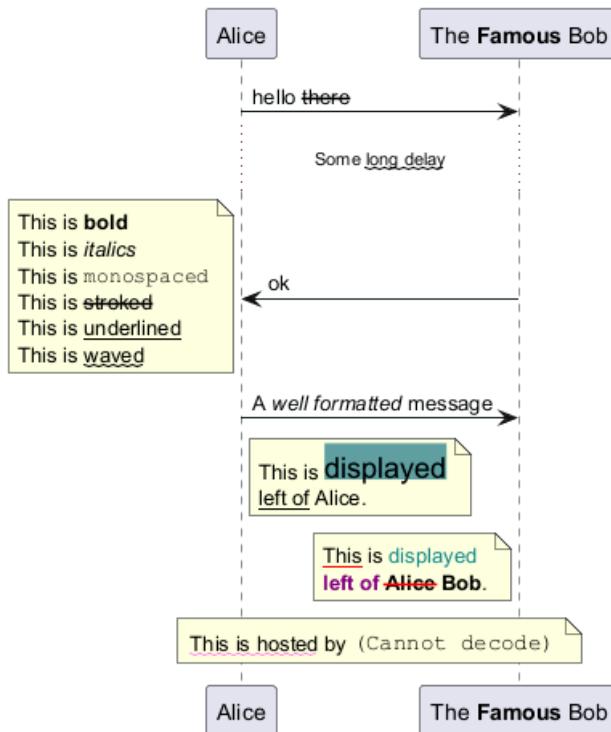
1.19 Creole и HTML

Так же можно использовать форматирование на Creole:

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

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

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



1.20 Разделитель

Вы можете использовать разделитель ==, чтобы разбить диаграмму на несколько логических этапов.

```
@startuml
```

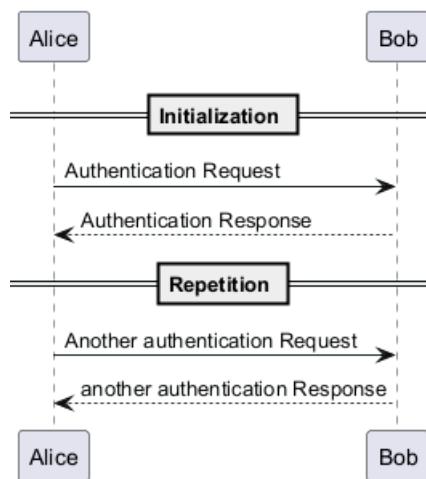
```
== Initialization ==
```

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

```
== Repetition ==
```

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

```
@enduml
```



1.21 Ссылки

Вы можете использовать ссылки в диаграммах с помощью ключевого слова ref over.

```
@startuml
participant Alice
actor Bob
```

```
ref over Alice, Bob :
```

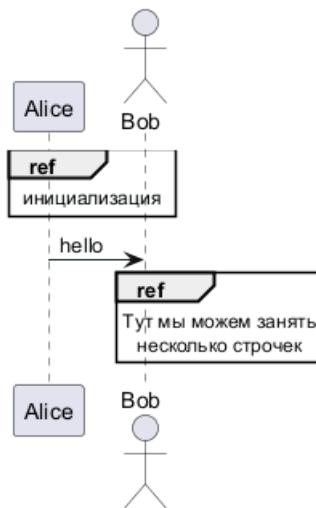
```
Alice -> Bob : hello
```

```
ref over Bob
```

```
end ref
```

```
@enduml
```





1.22 Задержка на диаграммах

Вы можете использовать конструкцию `...
...` для представления временной задержки в процессе на диаграмме. При необходимости можно снабдить задержку комментарием.

```
@startuml
```

```
Alice -> Bob:  
...  
Bob --> Alice:  
...5 ...  
Bob --> Alice: !
```

```
@enduml
```



1.23 Перенос текста

По умолчанию текст сообщения отображается в одну строку.

Перенос текста для его отображения на нескольких строках можно сделать:

- вручную, добавив `\n` в месте разрыва строки;
- автоматически, установив максимальное количество символов в строке с использованием параметра `maxMessageSize`. При этом перенос осуществляется по словам.

```
@startuml
```

```
skinparam maxMessageSize 100
```



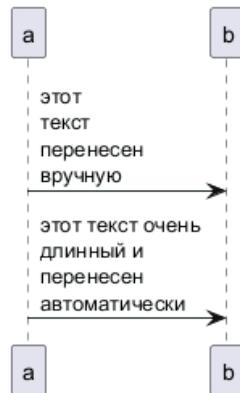
```

participant a
participant b

a -> b : \n\n\n
a -> b :

@enduml

```



1.24 Промежутки

Вы можете использовать ||| чтобы показать промежутки в диаграммах..

Так же возможно указать промежуток в пикселях.

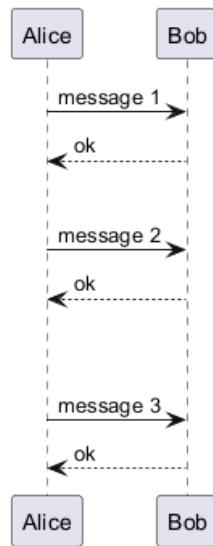
```
@startuml
```

```

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

```

```
@enduml
```



1.25 Активация и деактивация линии существования

`activate` и `deactivate` используются чтобы обозначить активацию участника.

Линия существования появляется в момент активации участника.

`activate` и `deactivate` применяются к предыдущему сообщению.

`destroy` обозначает конец линии существования участника.

```
@startuml
participant User
```

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

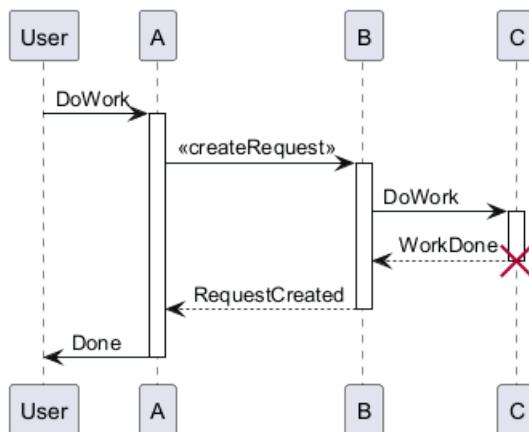
```
A -> B: << createRequest >>
activate B
```

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

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

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

```
@enduml
```



Можно использовать вложенные линии существования, и можно указывать цвет для этих линий.

```
@startuml
participant User
```

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

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

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

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

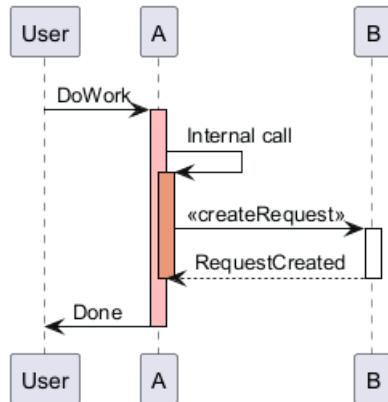


```

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

```

@enduml



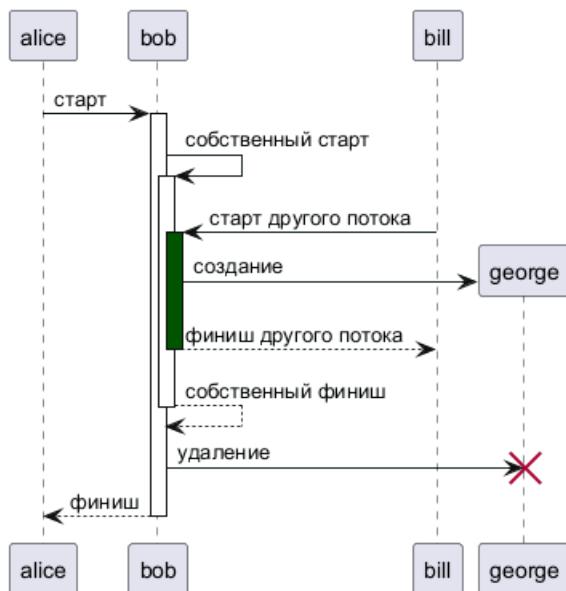
Autoactivation is possible and works with the return keywords: Автоактивация также возможна, для этого используйте команду autoactivate on. Для определения конца активации используются команды return:

```

@startuml
autoactivate on
alice -> bob :
bob -> bob :
bill -> bob #005500 :
bob -> george ** :
return
return
bob -> george !! :
return

```

@enduml



1.26 Return

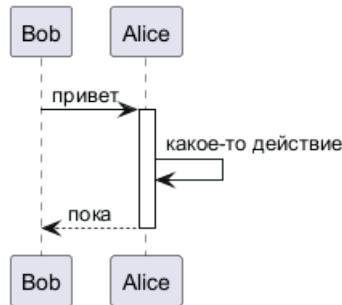
Команда `return` генерирует сообщение возврата с необязательной текстовой меткой.



Точкой возврата является та линия, которая вызвала последнюю активацию текущей линии жизни.

Синтаксис - return label, где label, если указано, является любой строкой, приемлемой для обычных сообщений.

```
@startuml
Bob -> Alice :
activate Alice
Alice -> Alice : -
return
@enduml
```



1.27 Отображение создания участника процессом

Вы можете использовать ключевое слово `create` перед декларацией сообщения для акцентирования факта, что принимающий участник *создается* данным сообщением.

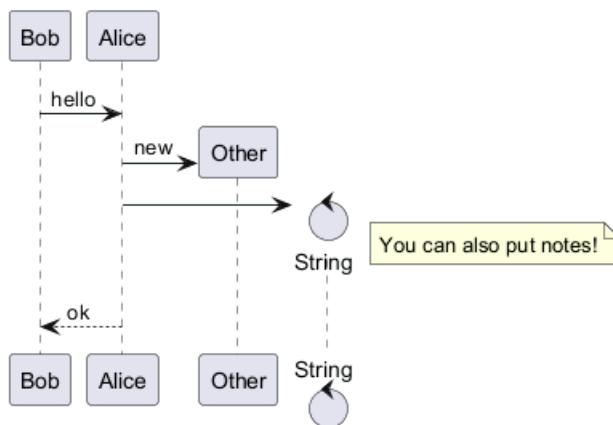
```
@startuml
Bob -> Alice : hello

create Other
Alice -> Other : new

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

Alice --> Bob : ok

@enduml
```



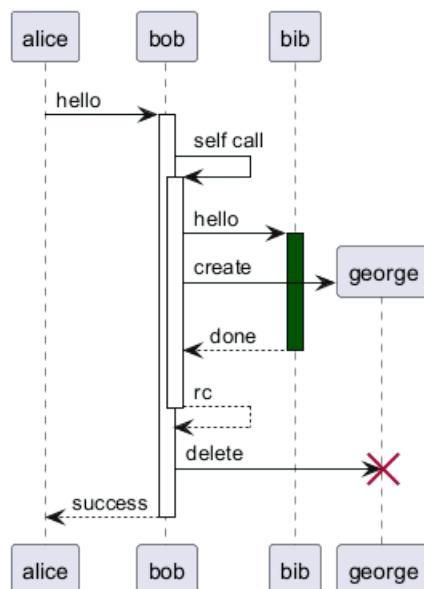
1.28 Быстрый синтаксис для активации, деактивации и создания

Сразу после указания целевого участника можно использовать следующий синтаксис:



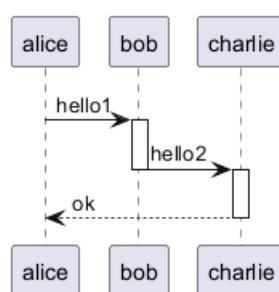
- ++ Активировать цель (опционально за этим может следовать цвет)
- -- Деактивировать источник
- ** Создать экземпляр цели
- !! Уничтожить экземпляр цели

```
@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
```



Затем вы можете смешивать активацию и деактивацию в одной строке

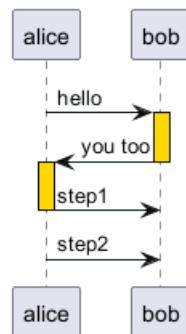
```
@startuml
alice -> bob ++ : hello1
bob -> charlie ---++ : hello2
charlie --> alice -- : ok
@enduml
```



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



```
@enduml
@enduml
```



[Ссылка на QA-4834, QA-9573 и QA-13234]

1.29 Входящие и исходящие сообщения

Вы можете использовать входящие или исходящие стрелки если вы хотите сфокусироваться на части диаграммы.

Используйте квадратные скобки для указания левой "[" или правой "]" стороны диаграммы

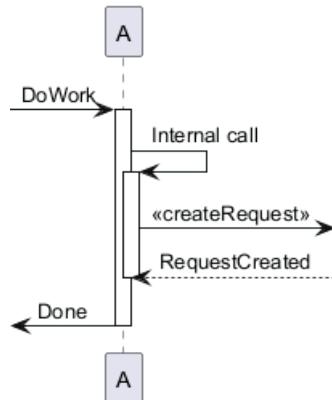
```
@startuml
[-> A: DoWork

activate A
```

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

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

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



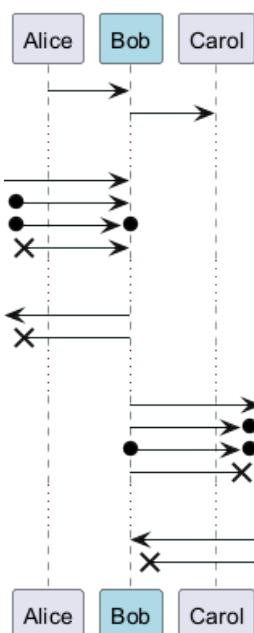
Вы также можете использовать следующий синтаксис:

```
@startuml
participant Alice
participant Bob #lightblue
Alice -> Bob
Bob -> Carol
...
[-> Bob
```



```
[o-> Bob
[o->o Bob
[x-> Bob
...
[<- Bob
[x<- Bob
...
Bob ->]
Bob ->o]
Bob o->o]
Bob ->x]
...
Bob <-]
Bob x<-]
```

@enduml

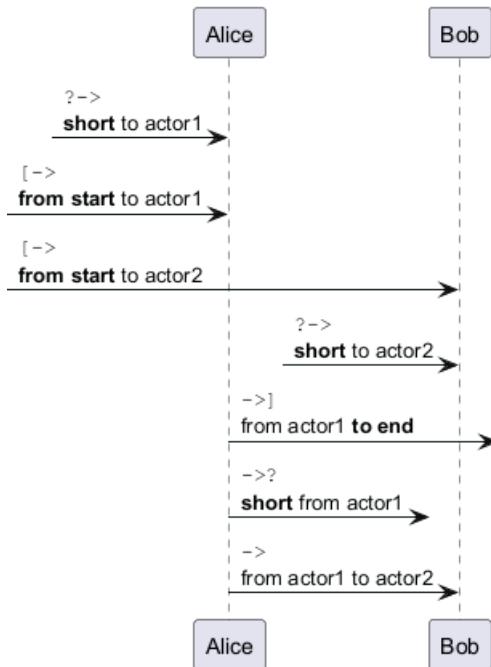


1.30 Короткие стрелки для входящих и исходящих сообщений

Вы можете использовать **короткие** стрелки с помощью ?

```
@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
```





[Ссылка на QA-310]

1.31 Якоря и длительность

С помощью teoz можно добавить якоря на диаграмму и использовать якоря для указания длительности

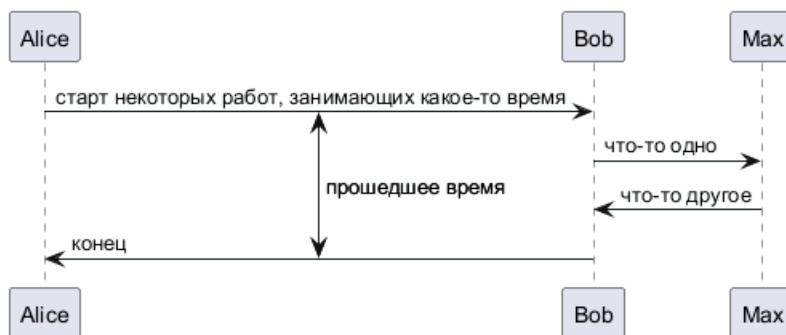
```

@startuml
!pragma teoz true

{start} Alice -> Bob : , , -
Bob -> Max : -
Max -> Bob : -
{end} Bob -> Alice :

{start} <-> {end} :

@enduml
  
```



Для указания прагмы можно использовать опцию командной строки -P

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

[Ref. issue-582]

1.32 Стереотипы и отметки

Можно добавить стереотипы к участникам используя << и >>.



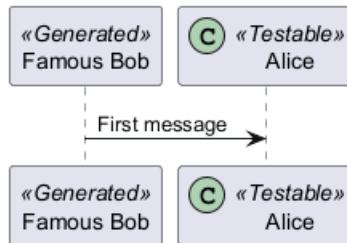
В стереотипе вы можете добавить отмеченного участника в цветном круге используя синтаксис `(X,color)`.

```
@startuml
```

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

Bob->Alice: First message

```
@enduml
```



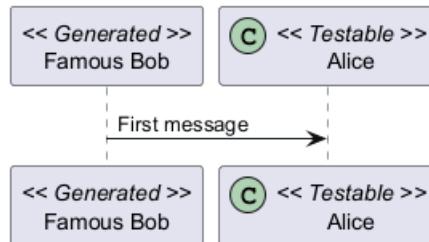
По умолчанию, для отображения стереотипа используются кавычки "ёлочки". Вы можете изменить это поведение, используя skinparam guillemet:

```
@startuml
```

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

Bob->Alice: First message

```
@enduml
```

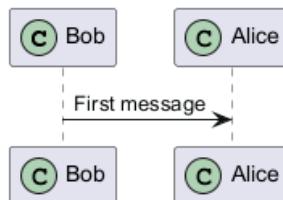


```
@startuml
```

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

Bob->Alice: First message

```
@enduml
```



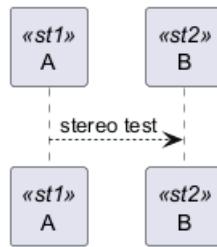
1.33 Расположение стереотипов

Чтобы указать желаемое расположение стереотипа используйте команды (`top` или `bottom`) вместе с командой `skinparam stereotypePosition`.

1.33.1 Расположение сверху (настройка по умолчанию)

```
@startuml
skinparam stereotypePosition top
```

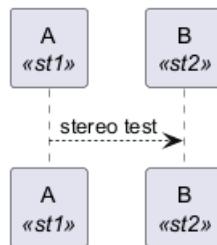
```
participant A<<st1>>
participant B<<st2>>
A --> B : stereo test
@enduml
```



1.33.2 Расположение снизу

```
@startuml
skinparam stereotypePosition bottom
```

```
participant A<<st1>>
participant B<<st2>>
A --> B : stereo test
@enduml
```



[Cм. QA-18650]

1.34 Больше информации в заголовках

Вы можете использовать форматирование на Creole для заголовков.

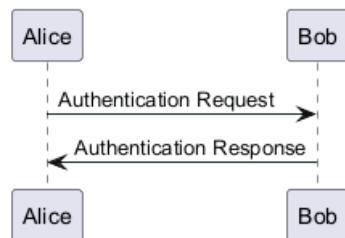
```
@startuml
```

```
title __Simple__ **communication** example
```

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

```
@enduml
```



Simple communication example

С помощью последовательности символов \n вы можете добавить перевод строки в заголовок.

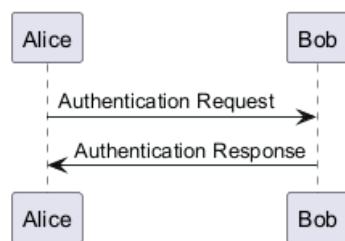
@startuml

```

title __Simple__ communication example\nnon several lines

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

@enduml

**Simple communication example
on several lines**

Вы также можете задать заголовок на нескольких строках, используя ключевые слова title и end title .

@startuml

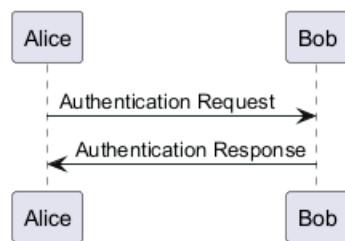
```

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

```

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

@enduml

**Simple communication example
on several lines and using html**
This is hosted by (Cannot decode)

1.35 Группировка участников

Можно создать прямоугольник вокруг участников, используя команды `box` и `end box`.

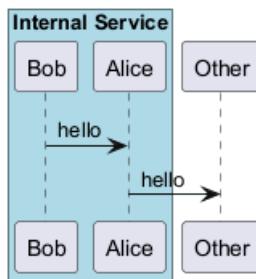
Вы можете задать настраиваемый заголовок и цвет фона в прямоугольнике группировки после команды `box`.

```
@startuml
```

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

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

```
@enduml
```



Архитектура teoz позволяет также создавать вложенные группы —группы внутри групп —например:

```
@startuml
```

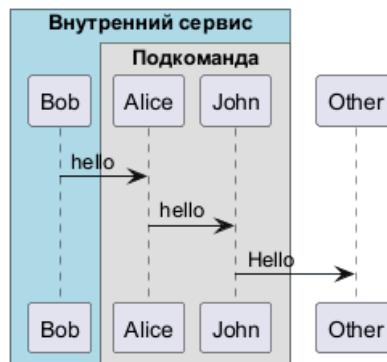
```
!pragma teoz true
box " " #LightBlue
participant Bob
box " "
participant Alice
participant John
end box

end box
participant Other
```

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

```
@enduml
```





1.36 Удаление футера

Вы можете использовать ключевое слово `hide footbox` для удаления футера из диаграммы.

```
@startuml
```

```
hide footbox
title Footer removed
```

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

```
@enduml
```



1.37 Skinparam

Вы можете использовать команду `skinparam` для изменения шрифтов и цветов диаграммы

Вы можете использовать данную команду :

- В определении диаграммы, как любую другую команду,
- В подключенном файле,
- В конфигурационном файле, указанном в командной строке в задании ANT.

Вы можете изменить другие параметры отображения, как видно из следующих примеров:

```
@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

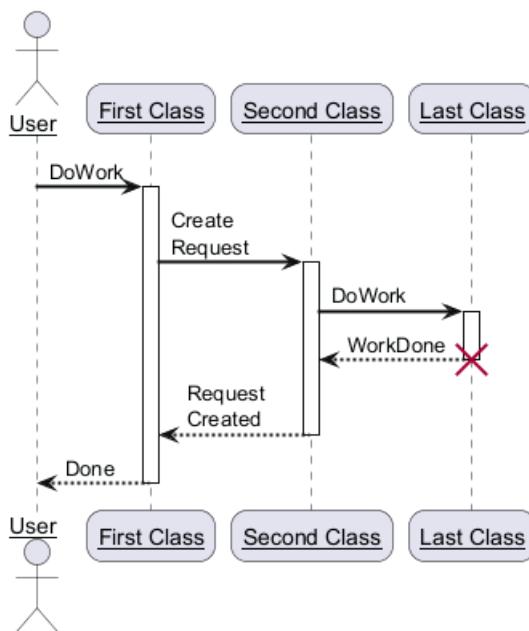
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

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

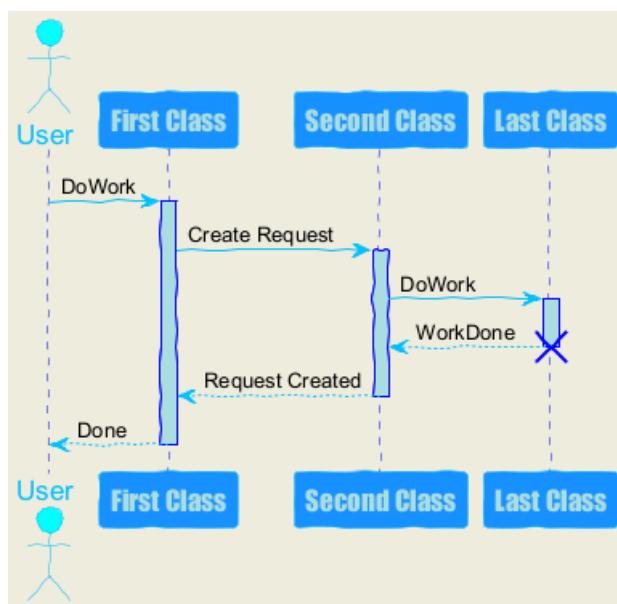
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml

```



1.38 Изменение отступов

Вы можете изменить некоторые настройки отступов

```

@startuml
skinparam ParticipantPadding 20
skinparam BoxPadding 10

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

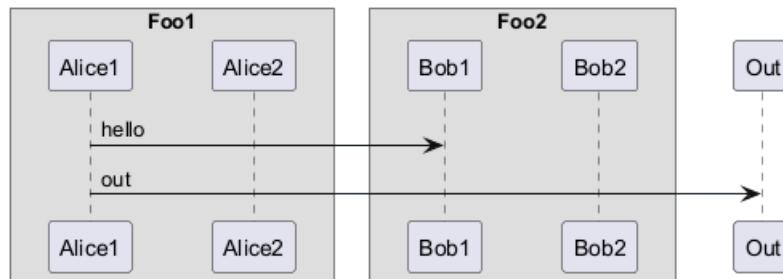
```



```

participant Bob2
end box
Alice1 -> Bob1 : hello
Alice1 -> Out : out
@enduml

```



1.39 Приложение: Примеры всех типов стрелок

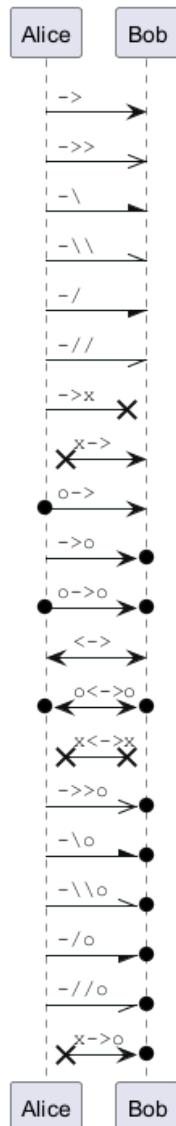
1.39.1 Обычная стрелка

```

@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

```





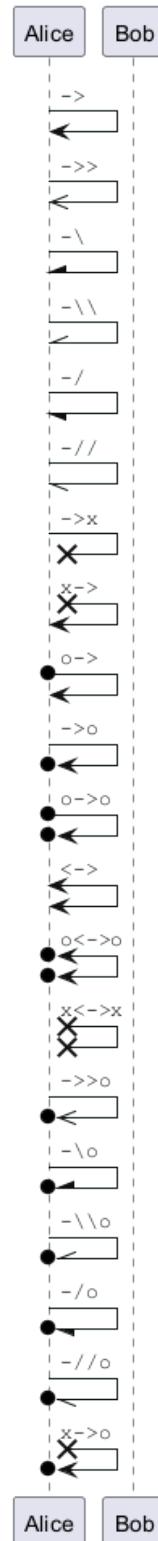
1.39.2 Возвращающаяся трелка

```

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



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



1.39.3 Входящие и исходящие сообщения (с '[' , ']')

1.39.4 Входящие сообщения (с '[')

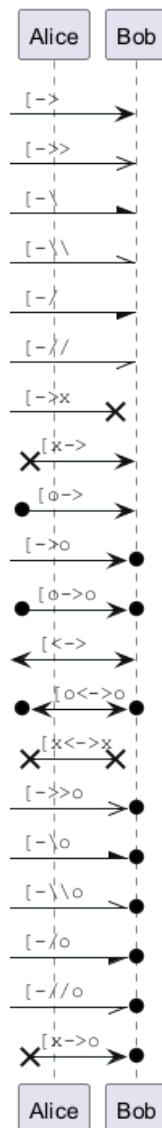
```
@startuml
```



```
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
```





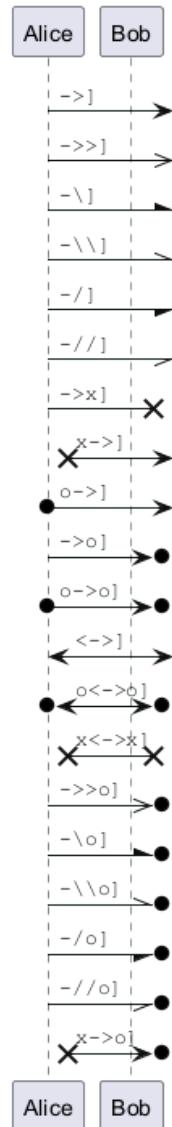
1.39.5 Исходящие сообщения (с '[')

```

@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
```



1.39.6 Короткие входящие и исходящие сообщения (с '?')

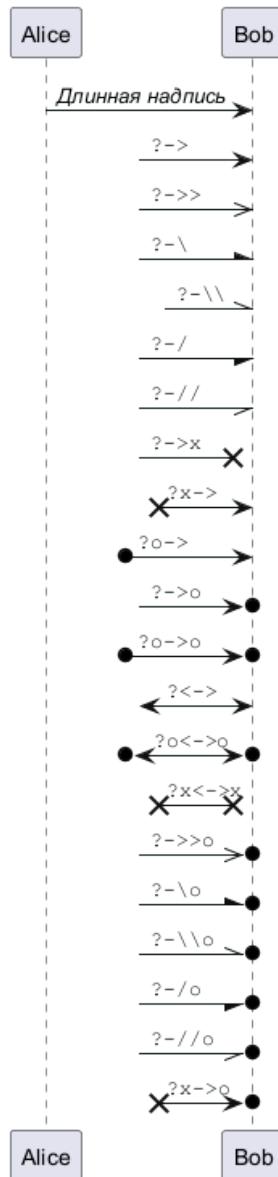
1.39.7 Короткие входящие (с '?')

```
@startuml
participant Alice as a
participant Bob   as b
a ->    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 """
?x->o   b : """?x->o """

@enduml
```



1.39.8 Короткие исходящие (с '??')

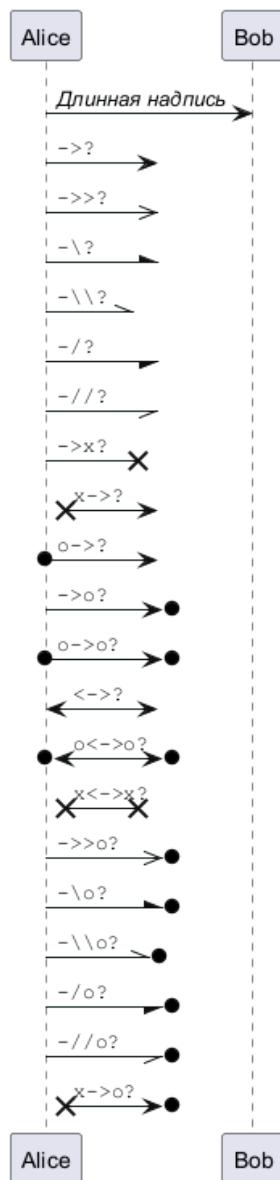
```
@startuml
participant Alice as a
participant Bob as b
a ->    b : //      //
a ->?   : """->?   """
a ->>?  : """->>? """
a -\?   : """-\?   """
```



```

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

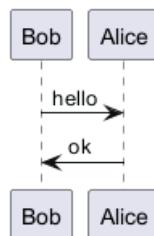
```



1.40 Разные стили отображения

1.40.1 По умолчанию

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

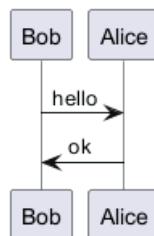


1.40.2 LifelineStrategy

- nosolid

Этот стиль и используется по умолчанию

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

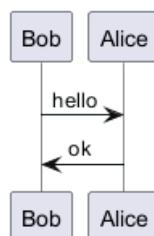


[Ссылка на QA-9016]

- solid

Для того, чтобы сделать вертикальные линии сплошными используйте команду `skinparam lifelineStrategy solid`

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



[Ссылка на QA-2794]

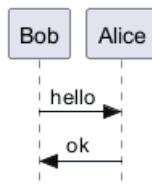


1.40.3 style strictuml

Для отображения в строгом соответствии с UML (*также меняется стиль стрелок: кончики отображаются как треугольники*), вы можете использовать

- `skinparam style strictuml`

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



[ссылка QA-1047]

1.41 Скрытие ни с кем не связанных участников

По умолчанию отображаются все участники.

```
@startuml
participant Alice
participant Bob
participant Carol

Alice -> Bob :
@enduml
```



Но Вы можете скрыть таких участников используя команду `hide unlinked`.

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

Alice -> Bob :
@enduml
```



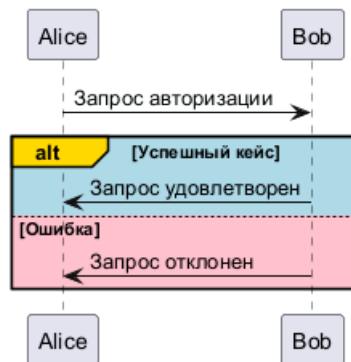
[Ref. QA-4247]



1.42 Цвет сгруппированных сообщений

Можно задать цвет для сгруппированных сообщений:

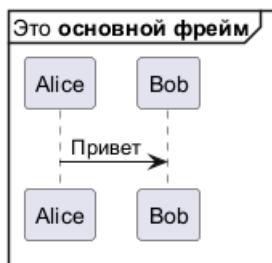
```
@startuml
Alice -> Bob:
alt#Gold #LightBlue
    Bob -> Alice:
else #Pink
    Bob -> Alice:
end
@enduml
```



[Ref. QA-4750 and QA-6410]

1.43 Основной фрейм

```
@startuml
mainframe ** **
Alice->Bob :
@enduml
```



[Ref. QA-4019 and Issue#148]

1.44 Slanted or odd arrows

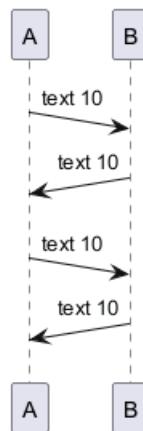
You can use the (nn) option (before or after arrow) to make the arrows slanted, where nn is the number of shift pixels.

[Available only after v1.2022.6beta+]

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

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

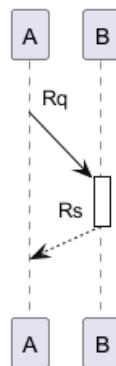




```

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

```

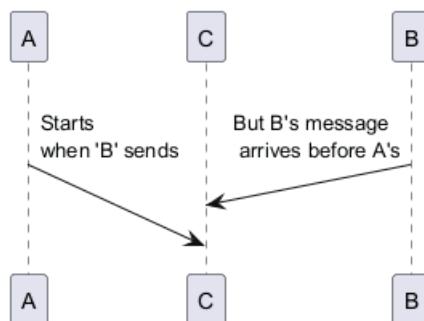


[Ref. QA-14145]

```

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

```



[Ref. QA-6684]

```

@startuml
!pragma teoz true

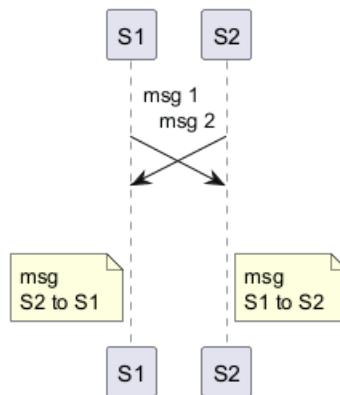
S1 ->(30) S2: msg 1\n
& S2 ->(30) S1: msg 2

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

```



```
@enduml
```

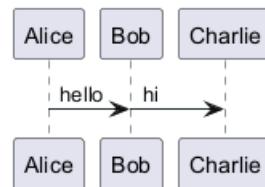


[Ref. QA-1072]

1.45 Параллельные сообщения (в архитектуре teoz)

Чтобы отображать команды параллельно используйте команду & (амперсанд) в окружении teoz:

```
@startuml
!pragma teoz true
Alice -> Bob : hello
& Bob -> Charlie : hi
@enduml
```



(См. подробнее про архитектуру Teoz)



2 Диаграмма вариантов использования

Диаграмма вариантов использования - это визуальное представление, используемое в программной инженерии для отображения взаимодействий между **участниками системы и самой системой**. Она отражает динамическое поведение системы, иллюстрируя ее **сценарии использования** и роли, которые с ними взаимодействуют. Эти диаграммы необходимы для уточнения **функциональных требований** к системе и понимания того, как пользователи будут взаимодействовать с системой. Предоставляя высокоуровневое представление, диаграммы вариантов использования помогают заинтересованным сторонам понять функциональность системы и ее потенциальную ценность.

PlantUML предлагает уникальный подход к созданию диаграмм вариантов использования за счет использования текстового языка. Одним из основных преимуществ использования PlantUML является его **простота и эффективность**. Вместо того чтобы вручную рисовать фигуры и связи, пользователи могут определять свои диаграммы с помощью интуитивно понятных и лаконичных текстовых описаний. Это не только ускоряет процесс создания диаграмм, но и обеспечивает **последовательность и точность**. Возможность интеграции с различными платформами документирования и широкий спектр поддерживаемых выходных форматов делают PlantUML универсальным инструментом как для разработчиков, так и для тех, кто не занимается разработкой. Наконец, благодаря **открытым исходным кодам** PlantUML имеет сильное сообщество, которое постоянно вносит свой вклад в его совершенствование и предлагает множество ресурсов для пользователей всех уровней.

2.1 Прецеденты

Прецеденты заключаются в две скобки (потому что две скобки выглядят как овал).

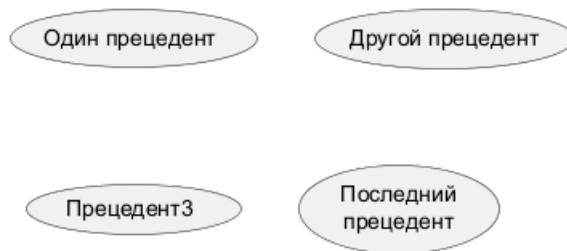
Вы можете использовать `usecase` для создания precedента.

Также, Вы можете создать псевдоним, используя ключевое слово `as`. Этот псевдоним позже можно будет использовать для определения связей между элементами.

```
@startuml
```

```
(          )
(          ) as (UC2)
usecase      3
usecase (      \n      ) as UC4
```

```
@enduml
```



2.2 Акторы (Роли, Участники)

Примечание переводчика: я не нашел одного устоявшегося варианта перевода для этого термина. Встречаются варианты: `,`, `или` . Далее по тексту буду использовать слово `.`

Определить актора можно просто указав его название между двумя вертикальными двоеточиями.

Другой вариант определения актора - это использование ключевого слова `actor`.

Также, при определении актора можно указать для него короткий псевдоним, который затем можно будет использовать при определении связей между элементами. Это делается при помощи ключевого слова `as`.

Пример ниже показывает разные варианты определения акторов.

```
@startuml
```

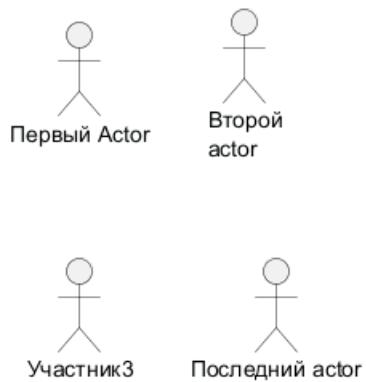


```

: Actor:
: \nactor: as      2
actor      3
actor :      actor: as      4

@enduml

```



2.3 Изменение стиля отображения актора

Вы можете изменить стиль отображения актора. Доступны следующие варианты:

- Человечек из палочек (*по умолчанию*);
- В виде бюста, команда `skinparam actorStyle awesome;`
- Контур человечка, команда `skinparam actorStyle hollow .`

2.3.1 Человечек из палочек (*по умолчанию*)

```

@startuml
:      : --> (Use)
"      " as Admin
"      " as (Use)
Admin --> (      )
@enduml

```



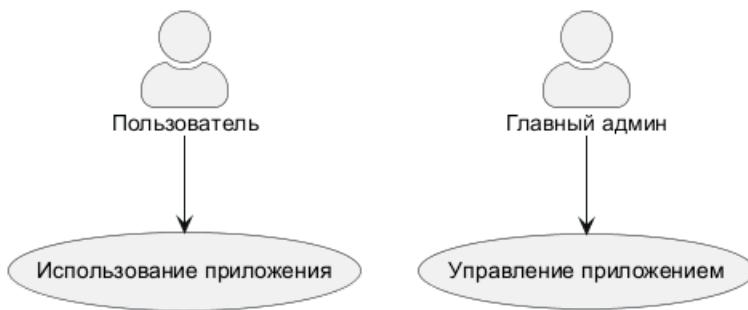
2.3.2 В виде бюста

```

@startuml
skinparam actorStyle awesome
:      : --> (Use)
"      " as Admin
"      " as (Use)
Admin --> (      )
@enduml

```





[Ссылка на QA-10493]

2.3.3 Контур человечка

```

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



[ссылка на PR#396]

2.4 Описание прецедентов

Если Вам нужно расположить описание в несколько строк, Вы можете использовать кавычки.

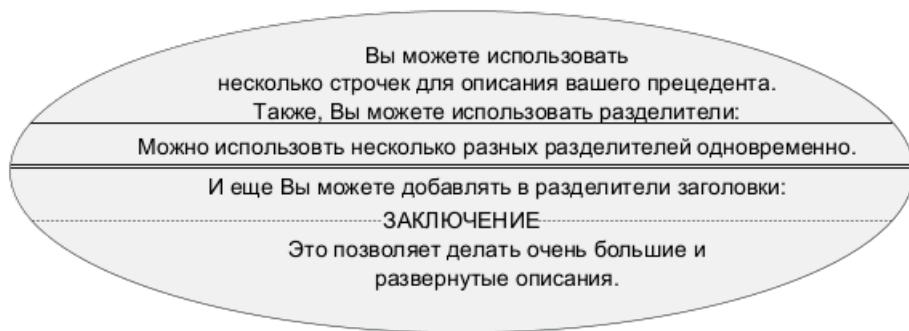
Также Вы можете использовать следующие разделители: --, . . . , == и ___. Внутрь этих разделителей можно вставлять заголовки.

```

@startuml

usecase UC1 as "
    .
    :
    --
    ==
    :
    ...
    ..
    .
"
@enduml
  
```

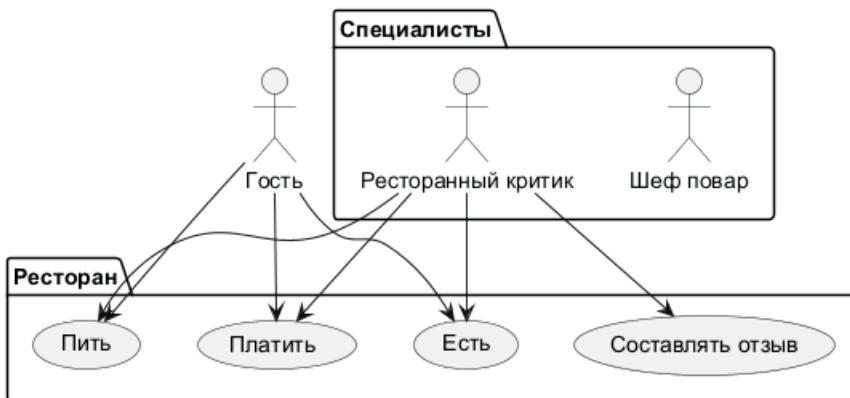




2.5 Использование пакетов

Чтобы сгруппировать акторов или прецеденты, можно использовать пакеты.

```
@startuml
actor      as g
package    {
    actor "      " as c
    actor "      " as fc
}
package    {
    usecase "  " as UC1
    usecase "  " as UC2
    usecase "  " as UC3
    usecase "      " as UC4
}
fc --> UC1
fc --> UC2
fc --> UC3
fc --> UC4
g --> UC1
g --> UC2
g --> UC3
@enduml
```



Для изменения внешнего вида пакета можно воспользоваться ключевым словом `rectangle` используя его вместо ключевого слова `package`.

```
@startuml
left to right direction
actor "      " as fc
rectangle    {
    usecase "  " as UC1
    usecase "  " as UC2
```



```

usecase " " as UC3
usecase " " as UC4
}
fc --> UC1
fc --> UC2
fc --> UC3
fc --> UC4
@enduml

```



2.6 Простой пример

Для создания связей между акторами и прецедентами используется стрелка `-->`.

Чем больше знаков тире – в стрелке, тем она длиннее. Вы можете добавить заметку на стрелку используя символ `:` после определения стрелки.

В этом примере можно заметить, что *Пользователь* не определён заранее и по умолчанию определяется как актор.

```
@startuml
```

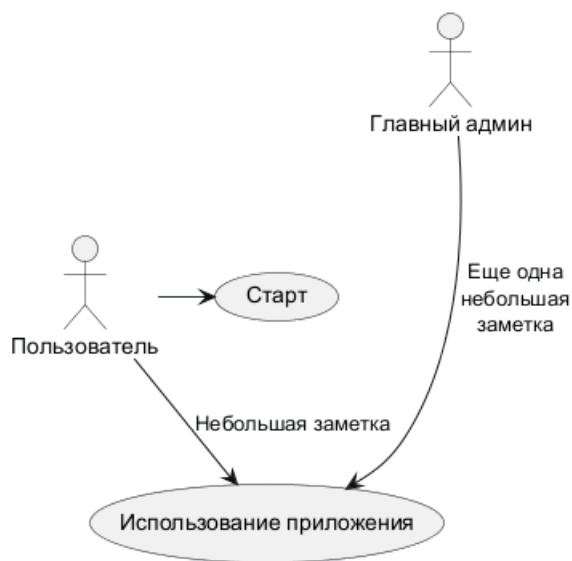
```

-> ( )
--> ( ) :
:
: : ---> ( ) :
\ n \ n

```

```
@enduml
```





2.7 Обобщение (наследование)

Если актор или прецедент являются обобщением других актора или прецедента, Вы можете использовать символ <|--.

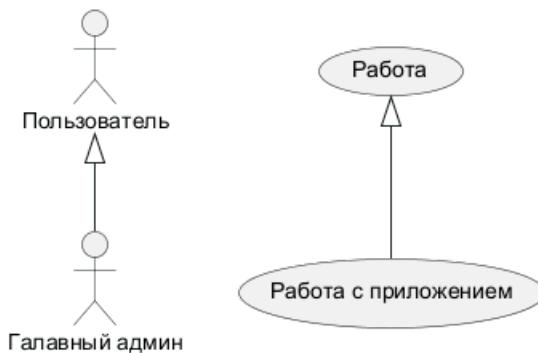
```

@startuml
:      : as Admin
(      ) as (Use)

<|-- Admin
(  ) <|-- (Use)

@enduml

```



2.8 Использование заметок

Вы можете использовать ключевые слова `note left of`, `note right of`, `note top of`, `note bottom of` чтобы создать и расположить заметку относительно какого-либо элемента.

Заметка так же может быть создана с помощью ключевого слова `note`, а затем прикреплена к какому-либо элементу при помощи символа ...

```

@startuml
:      : as Admin
(      ) as (Use)

-> (  )
--> (Use)

```



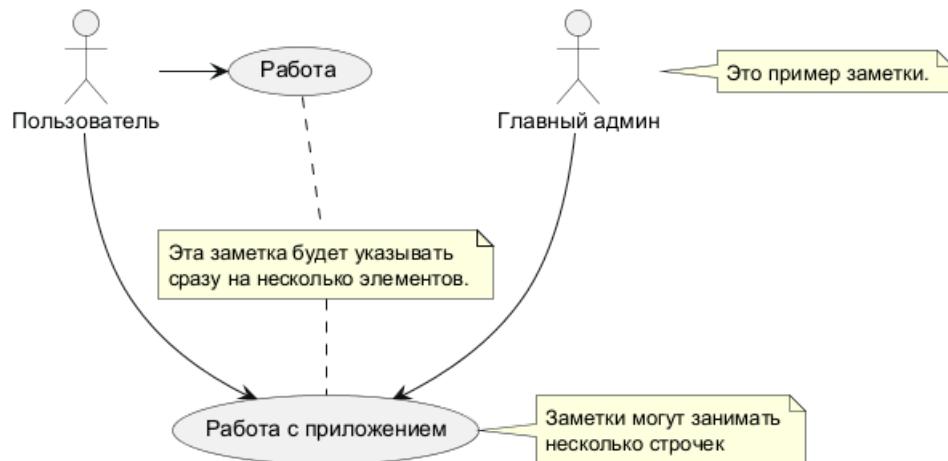
```
Admin ---> (Use)
```

```
note right of Admin : .
```

```
note right of (Use)
```

```
end note
```

```
note "
  \n
  ." as N2
( ) .. N2
N2 .. (Use)
@enduml
```



2.9 Шаблоны

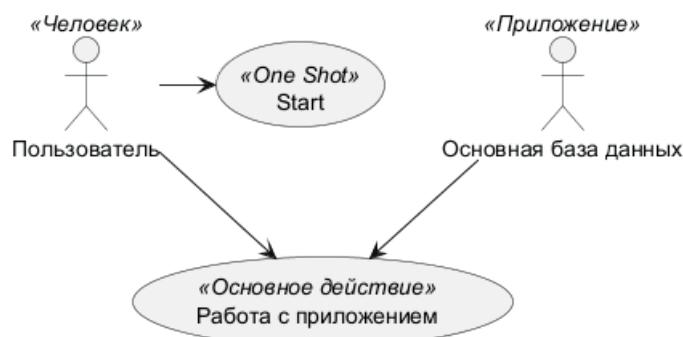
Вы можете добавить названия шаблонов когда определяете акторов или прецеденты используя символы << и >>.

```
@startuml
    <<      >>
    :       : as MySql <<      >>
(Start) << One Shot >>
(      ) as (Use) <<      >>

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

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

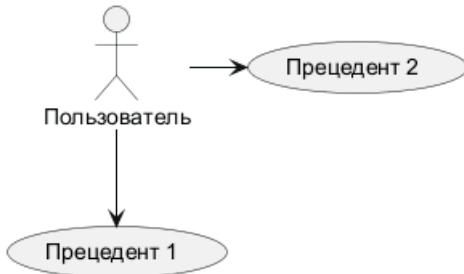
```
@enduml
```



2.10 Смена направления стрелок

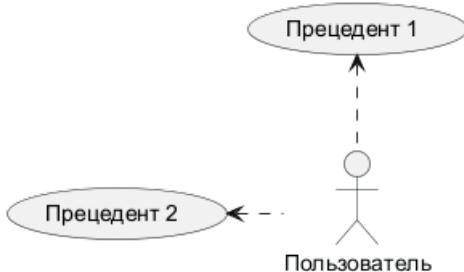
По умолчанию, связи между элементами задаются при помощи двух тире -- для сплошных линий или при помощи двух точек .. для пунктирных линий и ориентированы вертикально. Можно указать, что связь необходимо изобразить горизонтально, для этого необходимо использовать одиночное тире - для сплошной линии или одну точку . для штриховой линии.

```
@startuml
:      : --> (      1)
:      : -> (      2)
@enduml
```



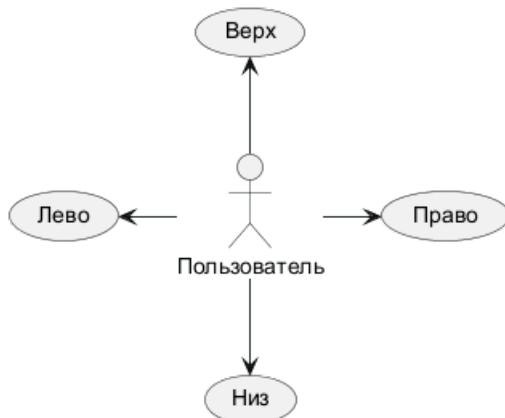
Также можно изменить направление связи, переместив символ окончания стрелки на другую сторону:

```
@startuml
(      1) <.. :      :
(      2) <.. :      :
@enduml
```



Можно управлять направлением связи добавляя ключевые слова left, right, up или down внутрь стрелки:

```
@startuml
:      : -left-> ( )
:      : -right-> ( )
:      : -up-> ( )
:      : -down-> ( )
@enduml
```

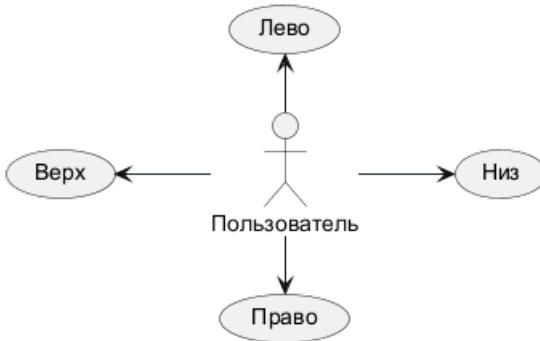


Можно использовать более короткий синтаксис используя только первый символ указанных выше ключевых слов, например `-d-` или `-do-` вместо полного слова `-down-`.

Правда злоупотреблять такой короткой записью не стоит, так как *GraphViz* обычно дает более хороший результат, когда такая короткая запись НЕ используется.

И наконец, можно поменять все направления местами используя ключевое слово `left to right direction`. Для этого можно сравнить предыдущую диаграмму с диаграммой, представленной ниже:

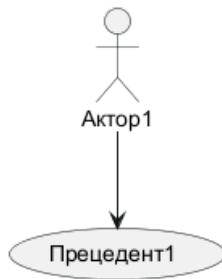
```
@startuml
left to right direction
:      : -left-> ( )
:      : -right-> ( )
:      : -up-> ( )
:      : -down-> ( )
@enduml
```



2.11 Разделение диаграмм

Ключевое слово `newpage` используется для разделения диаграмм на несколько страниц или изображений.

```
@startuml
: 1: --> (     1)
newpage
: 2: --> (     2)
@enduml
```

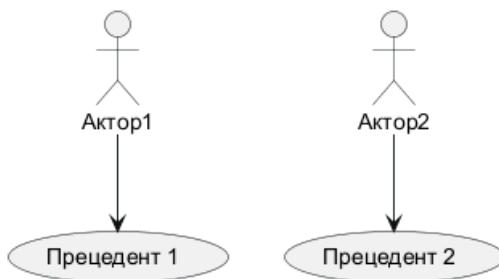


2.12 Направление слева направо

Общее поведение по умолчанию - построение диаграмм сверху вниз.

```
@startuml
'default
top to bottom direction
1--> (     1)
2--> (     2)
@enduml
```





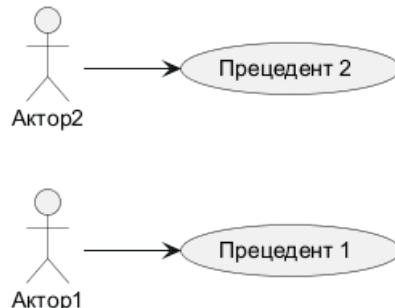
Вы можете изменить направление на **слева направо** используя команду `left to right direction`. Часто результат с таким направлением выглядит лучше.

`@startuml`

```

left to right direction
1--> (      1)
2--> (      2)
  
```

`@enduml`



See also 'Change diagram orientation' on [Deployment diagram](deployment-diagram) page.

2.13 Параметры отображения

Вы можете использовать команду `skinparam` для изменения шрифтов и цветов диаграммы.

Вы можете использовать данную команду :

- При определении диаграммы, как любую другую команду,
- При подключении файла,
- В конфигурационном файле, указанном в командной строке или в задании ANT.

Вы можете задать цвет или шрифт для шаблонов, в которые объединены акторы или прецеденты.

`@startuml`
`skinparam handwritten true`

```

skinparam usecase {
BackgroundColor DarkSeaGreen
BorderColor DarkSlateGray

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

ArrowColor Olive
ActorBorderColor black
ActorFontName Courier

ActorBackgroundColor<<      >> Gold
  
```



```

}

<<           >>
:           : as MySql <<           >>
(Start) << One Shot >>
(           ) as (Use) <<           >>

-> (Start)
--> (Use)

MySql --> (Use)

@enduml

```

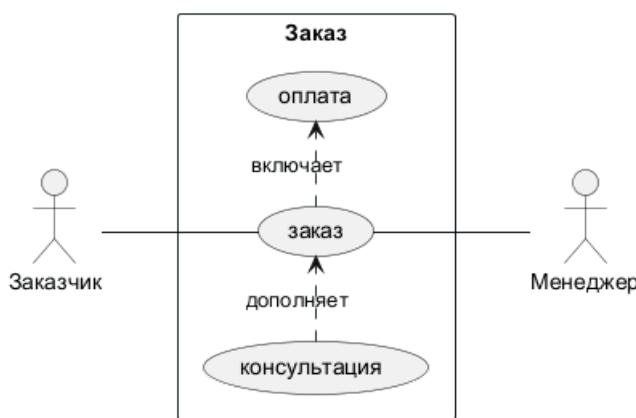


2.14 Полноценный пример

```

@startuml
left to right direction
skinparam packageStyle rectangle
actor
actor
rectangle {
    -- ( )
    ( ) .> ( ) :
    (     ) .> ( ) :
    ( ) --
}
@enduml

```



2.15 Бизнес прецеденты и акторы

Вы можете добавить символ / для того, чтобы создать Бизнес прецеденты или акторов.

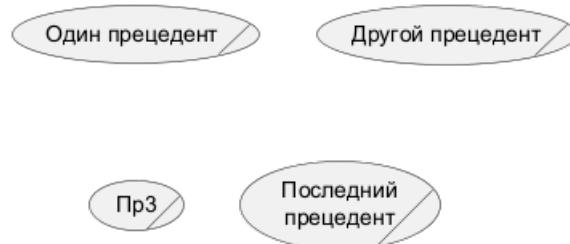


2.15.1 Бизнес прецедент

```
@startuml
```

```
(      )/  
(      )/ as (UC2)  
usecase/ 3  
usecase/ (      \n      ) as UC4
```

```
@enduml
```



2.15.2 Бизнес актор

```
@startuml
```

```
:      :/  
:      \n      :/ as Man2  
actor/  
actor/ :      as Person1
```

```
@enduml
```



[Ref. QA-12179]

2.16 Изменение цвета и стиля стрелок (встраиваемый синтаксис)

Вы можете изменить цвет или стиль индивидуально для каждой стрелочки используя следующий встраиваемый синтаксис:

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

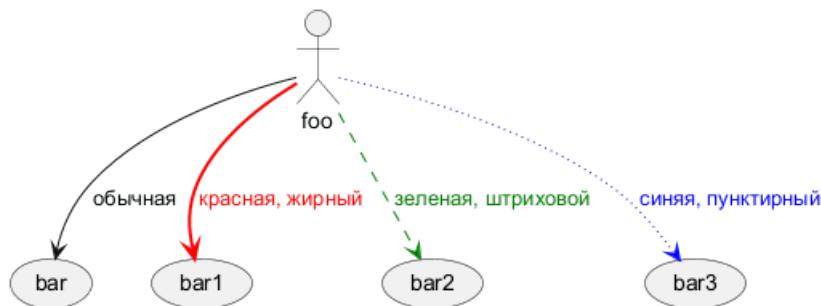
где:

- # - начало режима конфигурирования,
- color - цвет стрелки,
- ; - разделитель между конфигурационными параметрами,
- line.type - тип линии. type может принимать значения: bold - жирная, dashed - штриховая, dotted - пунктирная,



- `text:color` - цвет текста заметки на линии.

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



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

2.17 Изменение цвета и стиля элементов (встраиваемый синтаксис)

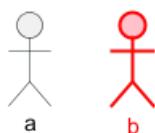
Вы можете изменить цвет или стиль индивидуально для каждого элемента используя следующий встраиваемый синтаксис:

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

где:

- `#` - начало режима конфигурирования,
- `color` или `back:color` - цвет фона элемента,
- `;` - разделитель между конфигурационными параметрами,
- `line.color` - цвет контура элемента,
- `line.type` - тип линии. `type` может принимать значения: `bold` - жирная, `dashed` - штриховая, `dotted` - пунктирная,
- `text:color` - цвет текста элемента.

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



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



2.18 Display JSON Data on Usecase diagram

2.18.1 Simple example

```
@startuml  
allowmixing  
  
actor      Actor  
usecase    Usecase  
  
json JSON {  
    "fruit": "Apple",  
    "size": "Large",  
    "color": ["Red", "Green"]  
}  
@enduml
```



JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

For another example, see on JSON page.



3 Диаграммы классов

Диаграммы классов строятся с использованием синтаксиса, повторяющего синтаксис, традиционно используемый в языках программирования. Такое сходство создает привычную среду для разработчиков, что способствует более легкому и интуитивно понятному процессу создания диаграмм.

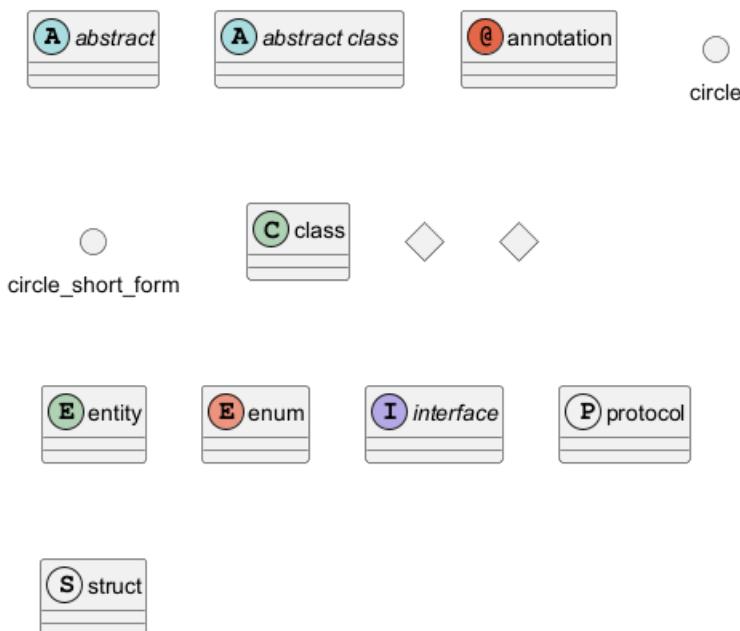
Такой подход к проектированию не только лаконичен, но и позволяет создавать представления, которые одновременно лаконичны и выразительны. Более того, он позволяет изображать отношения между классами с помощью синтаксиса, повторяющего синтаксис диаграмм последовательности, что открывает путь к плавному и глубокому изображению взаимодействия классов.

Помимо структурных и реляционных представлений, синтаксис диаграмм классов поддерживает такие дополнительные возможности, как включение примечаний и применение цветов, что позволяет пользователям создавать информативные и визуально привлекательные диаграммы.

Вы можете узнать больше о некоторых общих командах PlantUML для улучшения опыта создания диаграмм.

3.1 Декларирующий элемент

```
@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
protocol      protocol
struct        struct
@enduml
```



[Ссылка для protocol и struct: GH-1028]

[Ref. for protocol and struct: GH-1028, for exception: QA-16258]



3.2 Взаимосвязи между классами

Взаимосвязи между классами (согласно нотации UML) задаются с помощью следующих символов:

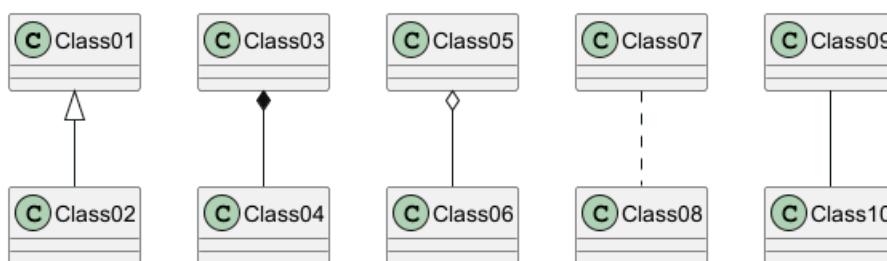
Тип	Символ	Назначение
Расширение	< --	Специализация класса в иерархии
Реализация	< ..	Реализация интерфейса классом
Композиция	*--	Часть не может существовать без целого
Агрегация	o--	Часть может существовать независимо от целого
Зависимость	-->	Объект использует другой объект
Зависимость	...>	Более слабая форма зависимости

Если рассматривать более универсально,

- есть два типа линий:
 - --- - рисует сплошную линию
 - ... - рисует штриховую линию
- к любой стороне (или сразу к обоим сторонам) линии, соединяющей два элемента, можно добавить разные наконечники:
 - < - рисует заостренный наконечник стрелочки
 - <| или ^ - рисует наконечник стрелочки в виде треугольника
 - * - рисует неконечник стрелочки в виде сплошного ромба
 - o - рисует неконечник стрелочки в виде полого ромба
 - # - рисует неконечник стрелочки в виде полого квадратика
 - x - рисует неконечник стрелочки в виде крестика
 - } - рисует неконечник стрелочки в виде обратного треугольника
 - + - рисует неконечник стрелочки в виде кружочка с крестиком внутри

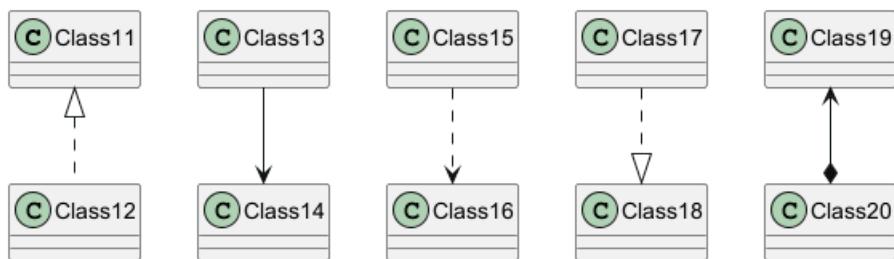
Зная эти правила можно нарисовать следующие изображения:

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



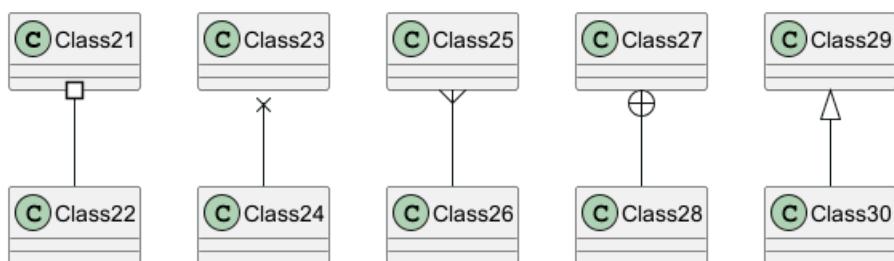
```
@startuml
Class11 <|.. Class12
Class13 --> Class14
Class15 ..> Class16
Class17 ..|> Class18
Class19 <--* Class20
@enduml
```





```

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



3.3 Метки на взаимосвязях

Для каждой связи можно добавить метку. Делается это с помощью добавления символа : после которого указывается текст метки.

Для добавления меток ближе к началу или концу стрелочки можно использовать двойные кавычки "" и помещать метки, обрамленные такими кавычками, перед или после стрелочки как на примере ниже:

```

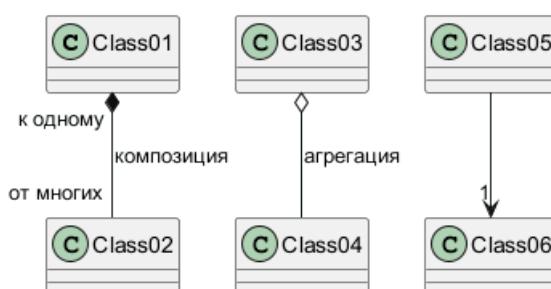
@startuml

Class01 "      " *-- "      " Class02 :

Class03 o-- Class04 :

Class05 --> "1" Class06

@enduml
  
```



Вы можете добавить дополнительные стрелки < или > в начале или в конце метки, чтобы указать на использование одного объекта другим.

```

@startuml
class
  
```



```

-      :      >
*-     :      4 >
--    :      <

```

@enduml



3.4 Использование не буквенных символов

Если Вы хотите использовать не буквенные (, /, \, *, # и прочее) или специальные unicode символы в названии класса (а также других элементов), Вам необходимо выполнить следующие условия:

- Поставить кавычки " " вокруг имени элемента которое содержит не буквенные, специальные unicode символы или символы пробела.
- Использовать ключевое слово `as` для того, чтобы дополнительно определить короткое ссылочное имя и по нему потом ссылаться на этот элемент при определении взаимосвязей

```

@startuml
class "          \n          " as class1
class class2 as "          "
class2 *-- "foo/dummy":           ,\n           ''/'''

class1 *--      :           \n           \n
@enduml

```



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

3.5 Добавление методов

Для объявления полей и методов вы можете использовать символ `:`. Для этого каждый раз указываете класс, затем символ `:` и затем имя поля или метода, который Вы хотите добавить в этот класс.

Система определяет что именно Вы добавили по типу скобок, которые Вы укажете после имени. Для поля данных это должны быть квадратные скобки [], а для метода круглые скобки () .

```

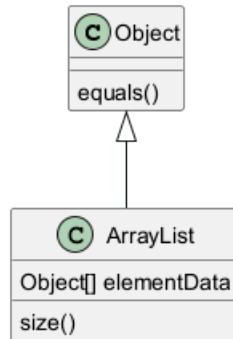
@startuml
Object <|-- ArrayList

```



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

@enduml



Чтобы не указывать каждый раз имя класса, для которого Вы хотите добавить поле данных или метод, можно все такие поля и методы указать сразу перечислив их внутри фигурных скобок {}.

Синтаксис порядка описания типа/имени довольно гибок.

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

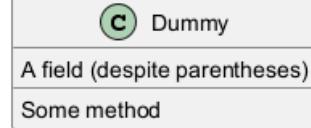
class Flight {
    flightNumber : Integer
    departureTime : Date
}
@enduml
```



Вы можете специально указать что именно Вы добавляете (поле данных или метод) используя ключевые слова {field} - поле данных или {method} - метод. Они переопределяют собой типы, определяемые парсером по умолчанию.

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

@enduml
```



3.6 Указание видимости

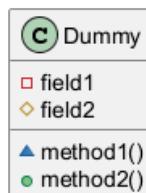
Определяя методы и поля данных, вы можете использовать символы указания видимости, приведённые в таблице ниже:

Символ	Иконка для поля данных	Иконка для метода	Видимость
-	□	■	private
#	◊	◊	protected
~	△	▲	package private
+	○	●	public

```
@startuml
```

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

```
@enduml
```



Заменить значки их текстовыми представлениями можно используя команду `skinparam classAttributeIconSize 0`:

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

```
@enduml
```



[Ref. [QA-4755](https://forum.plantuml.net/4755/provide-display-visibility-attributes-private-protected)]

3.7 Абстрактные и статические

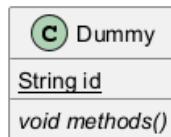
Вы можете определить статические или абстрактные методы и поля данных используя модификаторы `{static}` и `{abstract}` соответственно.

Эти модификаторы могут располагаться как в начале, так и в конце строки. Вы так же можете использовать `{classifier}` как замену для `{static}`.

```
@startuml
```



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



3.8 Расширенное тело класса

По умолчанию, методы и поля автоматически группируются PlantUML. Вы можете использовать разделители, чтобы определить собственный порядок полей и методов. Можно использовать следующие разделители:

-- .. == __.

Вы также можете использовать заголовки внутри разделителей:

```
@startuml
class Foo1 {

    ..
    ,
    ==

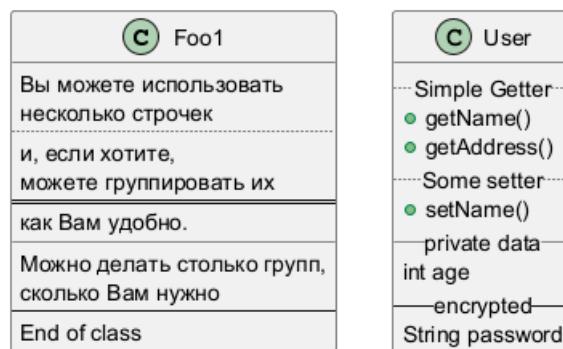
    --
    ,

    --
    End of class
}

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

@enduml
```





3.9 Заметки и шаблоны

Шаблоны задаются ключевым словом `class` с парой символов `<<` и `>>`.

Также вы можете создать и спозиционировать относительно любого класса заметку, используя ключевые слова `note left of`, `note right of`, `note top of` и `note bottom of`.

Вы также можете создать и спозиционировать заметку относительно последнего определённого класса, используя ключевые слова `note left`, `note right`, `note top` и `note bottom`.

Ключевым словом `note` легко создать заметку без привязки и затем привязать её используя символ `..`

```

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

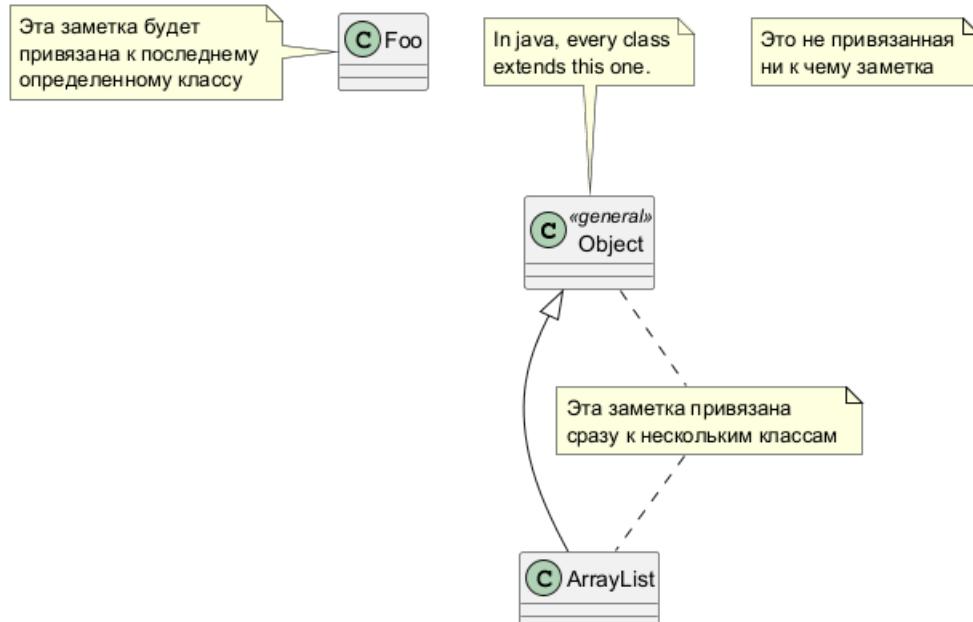
note top of Object : In java, every class\nextends this one.

note "          \n      " as N1
note "          \n      " as N2
Object .. N2
N2 .. ArrayList

class Foo
note left:           \n

```

`@enduml`



3.10 Больше о заметках

Допускается использование некоторых HTML-тегов (также смотри синтаксис записи Creole), таких как:

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

Также, заметку можно задавать в несколько строчек. В таком случае, конец заметки определяется по ключевым словам `end note`.

```
@startuml
```

```
class Foo
note left: On last defined class

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

note as N1
    <u>    </u>
    <b><color:royalBlue>      </color>
    <s>      </s> .
    And this is hosted by <img:sourceforge.jpg>
end note
```

```
@enduml
```



3.11 Создание заметки для поля данных или метода

Можно создать и привязать заметку к полю данных или методу.

3.11.1 □ Ограничения

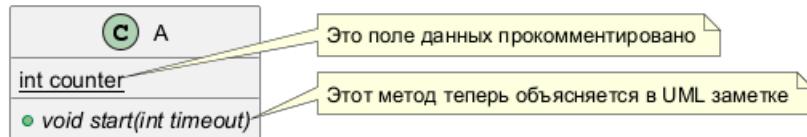
- Нельзя использовать `top` или `bottom` (*реализованы только left и right*)
- Нельзя использовать с разделителем пространства имён `::`



3.11.2 Заметка к полю данных или методу

```
@startuml
class A {
{static} int counter
+void {abstract} start(int timeout)
}
note right of A::counter

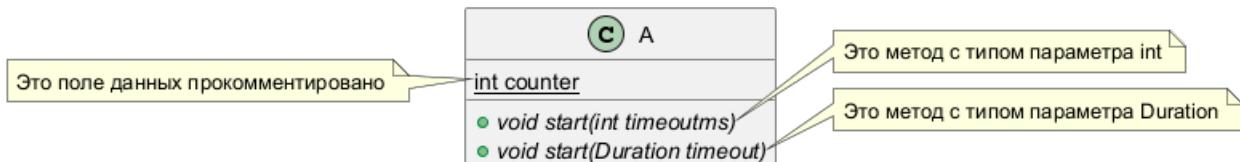
end note
note right of A::start
    UML
end note
@enduml
```



3.11.3 Заметка для разных методов с одинаковым именем

```
@startuml
class A {
{static} int counter
+void {abstract} start(int timeoutms)
+void {abstract} start(Duration timeout)
}
note left of A::counter

end note
note right of A::"start(int timeoutms)"
    int
end note
note right of A::"start(Duration timeout)"
    Duration
end note
@enduml
```



[Ref. QA-3474 and QA-5835]

3.12 Заметки на взаимосвязях

Чтобы добавить заметку на связь, сразу после определения такой связи в следующей строке разместите команду `note on link`.

Вы также можете использовать `note left on link`, `note right on link`, `note top on link` или `note bottom on link` для того, чтобы добавить заметку и спозиционировать её относительно последней определенной связи.

```
@startuml
```

```
class Dummy
```



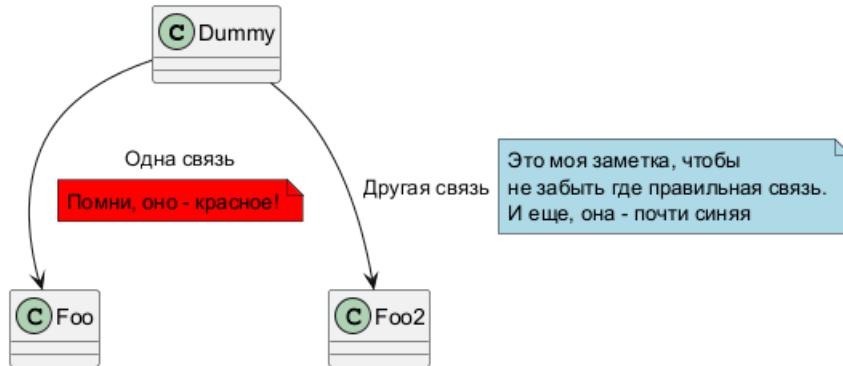
```

Dummy --> Foo :
note on link #red: , - !
  

Dummy --> Foo2 :
note right on link #lightblue
,
,
end note
  

@enduml

```



3.13 Абстрактные классы и интерфейсы

Вы можете определить класс как абстрактный, используя ключевые слова `abstract` или `abstract class`.

Такие классы будут нарисованы *курсивом*.

А еще Вы можете использовать ключевые слова `interface`, `annotation` и `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

```

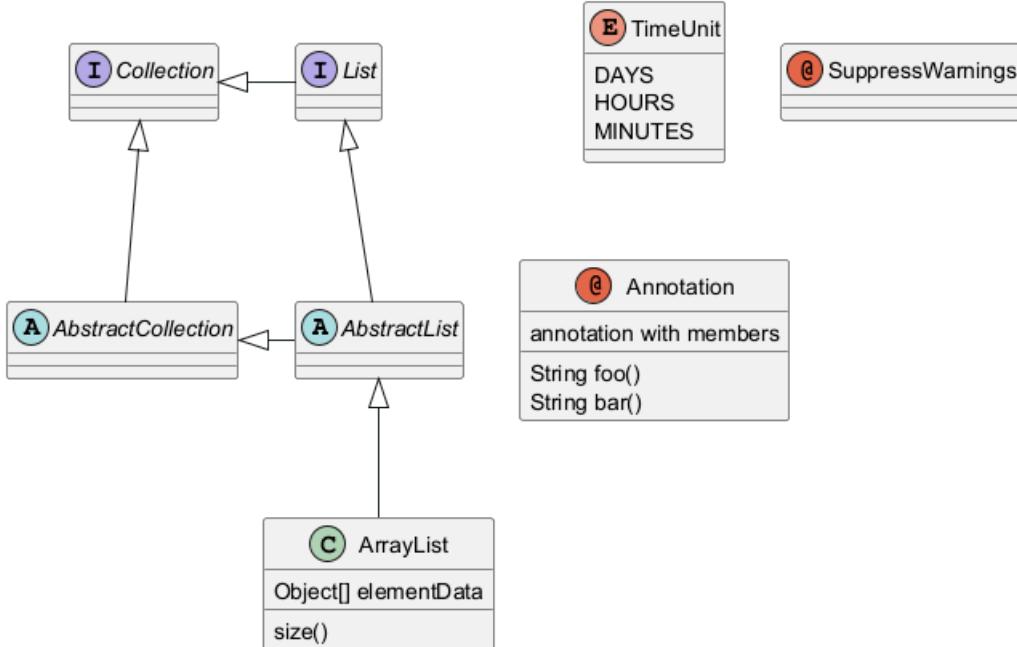


```

annotation Annotation {
    annotation with members
    String foo()
    String bar()
}

@enduml

```



[Ref. 'Annotation with members' Issue#458]

3.14 Скрыть атрибуты, методы...

Вы можете параметризовать отображение классов с помощью команды `hide/show`.

Основная команда: `hide empty members`. Эта команда скроет атрибуты или методы, если они пусты.

Вместо `empty members` можно использовать:

- `empty fields` или `empty attributes` для пустых полей,
- `empty methods` для пустых методов,
- `fields` или `attributes`, которая скроет поля, даже если они описаны,
- `methods` которая скроет методы, даже если они описаны,
- `members` которая скроет поля иметоды, даже если они описаны,
- `circle` для обведенного символа перед именем класса,
- `stereotype` для стереотипа.

Вы также можете указать, сразу после ключевого слова `hide` или `show`:

- `class` для всех классов,
- `interface` для всех интерфейсов,
- `enum` для всех перечислений,
- `<<foo1>>` для классов, стереотип которых `foo1`,
- существующее имя класса.

Вы можете использовать несколько команд `show/hide` для определения правил и исключений



```
@startuml
```

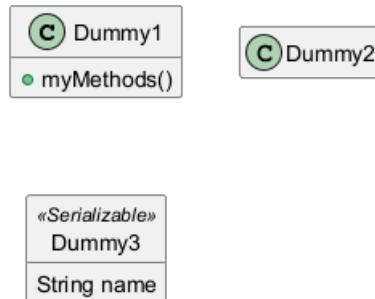
```
class Dummy1 {
    +myMethods()
}

class Dummy2 {
    +hiddenMethod()
}

class Dummy3 <<Serializable>> {
String name
}

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

@enduml
```



[Ref. QA-2913]

3.15 Скрытие классов

Вы также можете использовать команду `show/hide`, чтобы скрывать классы.

Это может быть полезно, если Вы подключили большой файл, и хотите скрыть некоторые классы из этого файла после его включения. Сам класс и связи, которые к нему вели будут скрыты, но место, которое он занимал на экране по прежнему будет им занято.

```
@startuml
```

```
class Foo1
class Foo2
class Foo3

Foo2 *-- Foo1
Foo3 *-- Foo1
Foo3 *-- Foo2

hide Foo1
hide Foo2

@enduml
```





3.16 Удаление классов

Вы можете использовать команду `remove`, чтобы удалять классы.

Это может быть полезно, если Вы подключили большой файл, и хотите полностью удалить некоторые классы из этого файла после его включения. В отличие от команды `hide`, которая только скрывает класс при отображении его на экране (и при этом этот скрытый класс продолжает занимать на экране место), при использовании команды `remove` класс и его связи будут полностью удалены перед формированием изображения, как будто они даже не определялись.

```
@startuml
class Foo1
class Foo2
class Foo3

Foo2 *-- Foo1
Foo3 *-- Foo1
Foo3 *-- Foo2

remove Foo1
remove Foo2

@enduml
```



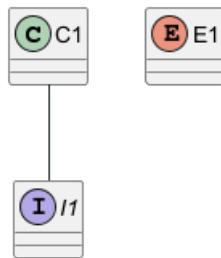
3.17 Hide, Remove or Restore tagged element or wildcard

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

By default, all components are displayed:

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





But you can:

- hide \$tag13 components:

```

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

hide $tag13
@enduml

```



- or remove \$tag13 components:

```

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

remove $tag13
@enduml

```



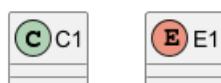
- or remove \$tag13 and restore \$tag1 components:

```

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

remove $tag13
restore $tag1
@enduml

```



- or remove * and restore \$tag1 components:

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

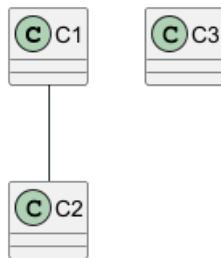
remove *
restore $tag1
@enduml
```



3.18 Скрытие или Удаление классов баз связей

По умолчанию отображаются все классы:

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



Но вы можете:

- скрывать классы без связей при помощи команды `hide @unlinked`:

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

hide @unlinked
@enduml
```



- полностью удалять классы без связей при помощи команды `remove @unlinked`:

```
@startuml
class C1
```



```

class C2
class C3
C1 -- C2

remove @unlinked
@enduml

```



[Adapted from QA-11052]

3.19 Использование дженериков

Вы можете использовать угловые скобки < и > чтобы указать на использование дженериков в классе.

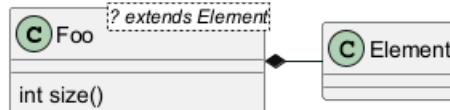
```
@startuml
```

```

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

```

```
@enduml
```



Также Вы можете отключить отрисовку этих элементов, используя команду `skipparam genericDisplay old`.

3.20 Определение метки

Обычно, метка с буквой (C, I, E or A) используется для классов, интерфейсов, перечислений и абстрактных классов.

Но Вы можете создать свою собственную метку для класса при создании шаблона задав для неё собственную букву и цвет (используя html-код для цвета или название цвета).

Русские буквы также можно использовать.

```
@startuml
```

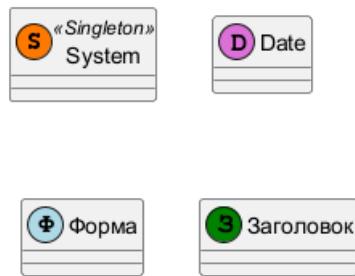
```

class System << (S,#FF7700) Singleton >>
class Date << (D,orchid) >>
class     << (,lightblue) >>
class     << (,green) >>

```

```
@enduml
```





3.21 Пакеты

Вы можете создать пакет, используя ключевое слово `package`, также можно дополнительно указать для него цвет фона (используя html-код для цвета или название цвета).

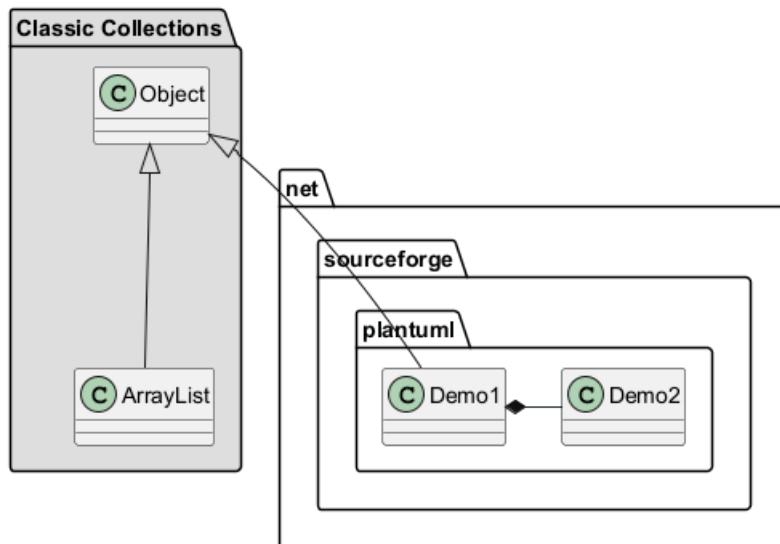
Обратите внимание, что пакеты могут быть вложенными.

```
@startuml
```

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

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

```
@enduml
```



3.22 Стили пакетов

Доступны различные стили для пакетов.

Можно глобально задать стиль по умолчанию с помощью команды: `skinparam packageStyle` , или указать стиль для каждого пакета по отдельности при помощи определения для него шаблона `package <<-->>`.

На примере ниже можно посмотреть доступные варианты стилей для пакетов:

```
@startuml
scale 750 width
package foo1 <<Node>> {
```



```

class Class1
}

package foo2 <<Rectangle>> {
    class Class2
}

package foo3 <<Folder>> {
    class Class3
}

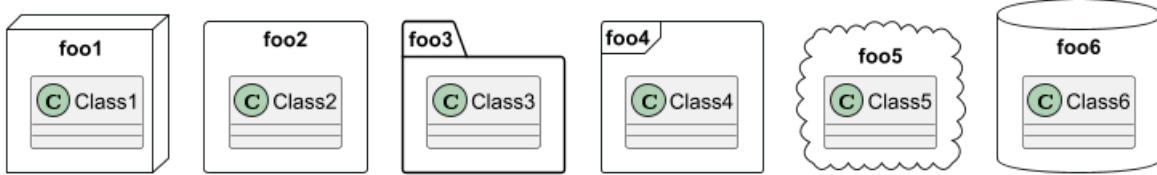
package foo4 <<Frame>> {
    class Class4
}

package foo5 <<Cloud>> {
    class Class5
}

package foo6 <<Database>> {
    class Class6
}

@enduml

```



Между пакетами также можно указывать взаимосвязи:

```

@startuml

skinparam packageStyle rectangle

package foo1.foo2 {

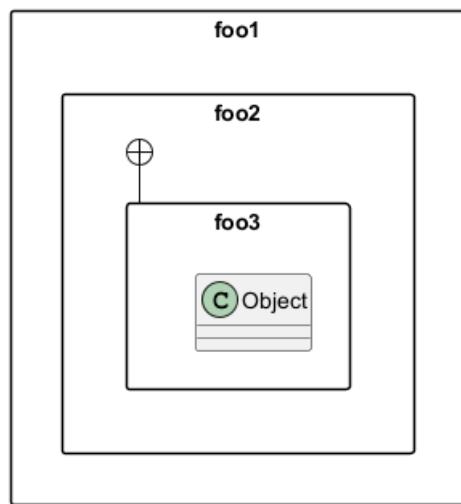
package foo1.foo2.foo3 {
    class Object
}

foo1.foo2 +-- foo1.foo2.foo3

@enduml

```





3.23 Пространства имён

Даже если класс располагается внутри пакета, его имя остается уникальным идентификатором для этого класса. Это значит, что у Вас не может быть двух классов в разных пакетах, но с одинаковым именем.

В этом случае, вам следует использовать пространства имён вместо пакетов.

Вы можете ссылаться на классы из других пространств имён по их полному определению. Классы без пространства имён определяются по наличию точки перед их именами.

Обратите внимание, что вы не обязаны явно создавать пространство имён: правильно определенный класс, в имени которого через точку указано пространство имён и затем его имя автоматически попадает в правильное пространство имён.

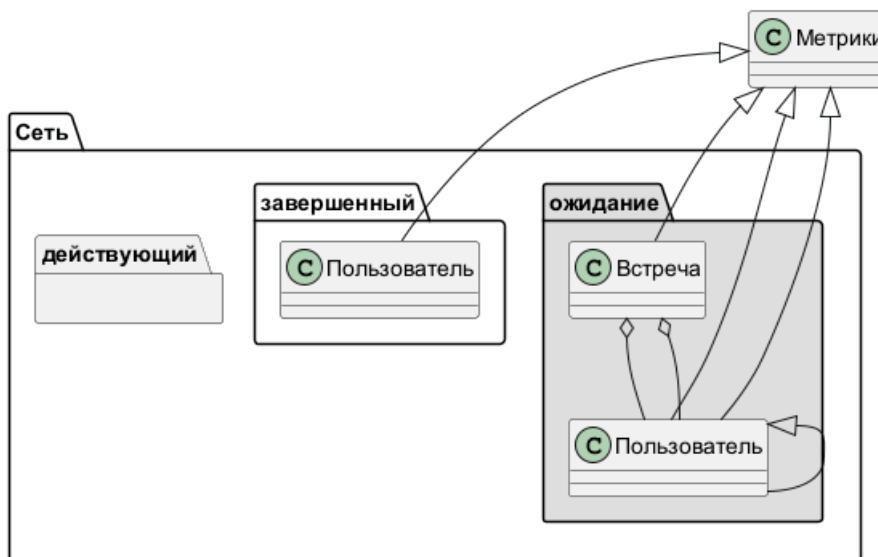
@startuml

```

class
namespace . #DDDDDD {
    .
    <|-- .
    o-- .
    .
    <|-
}
namespace . {
    .
    <|-
    <|-
    .
    o-- .
}
<|-- .
@enduml
  
```

@enduml





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

3.24 Настройка создания пространств имён

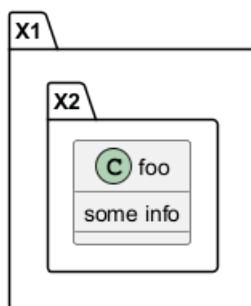
Вы можете задать другой разделитель (не точку) используя команду:

- set namespaceSeparator

@startuml

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

@enduml



также Вы можете отключить автоматическое создание пакетов используя команду:

- set namespaceSeparator none.

@startuml

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

@enduml



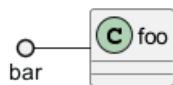


3.25 Lollipop интерфейс

Вы можете задавать интерфейсы в виде "lollipop" для классов, используя следующий синтаксис:

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

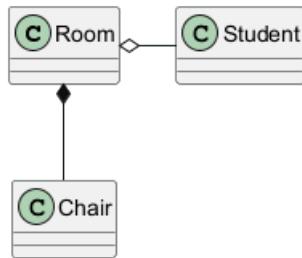
```
@startuml
class foo
bar ()- foo
@enduml
```



3.26 Изменение направления стрелок

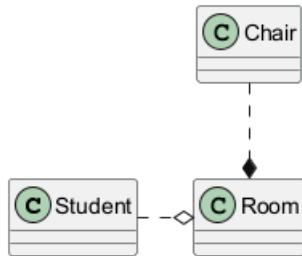
По умолчанию, связи между элементами задаются при помощи двух тире -- для сплошных линий или при помощи двух точек .. для пунктирных линий и ориентированы вертикально. Можно указать, что связь необходимо изобразить горизонтально, для этого необходимо использовать одиночное тире - для сплошной линии или одну точку . для штриховой линии.

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



Изменить направление стрелки можно, переместив символ окончания стрелки на другую сторону:

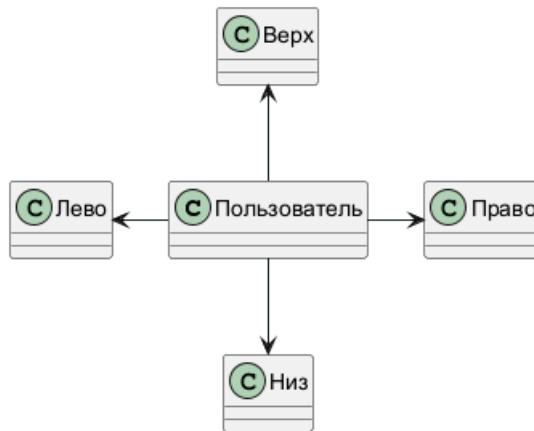
```
@startuml
Student .o Room
Chair ..* Room
@enduml
```



Можно управлять направлением связи добавляя ключевые слова left, right, up или down внутри стрелки:



```
@startuml
-left->
-right->
-up->
-down->
@enduml
```

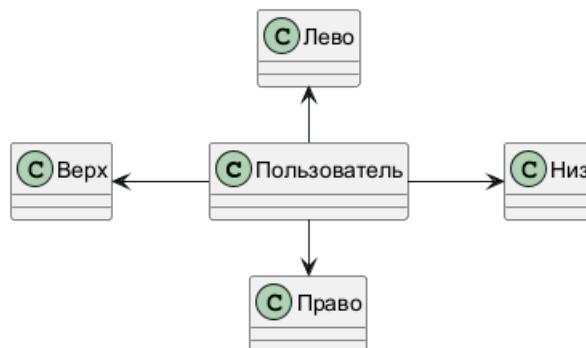


Можно использовать более короткий синтаксис используя только первый символ указанных выше ключевых слов, например -d- или -do- вместо полного слова -down-.

Правда злоупотреблять такой короткой записью не стоит, так как *GraphViz* обычно дает более хороший результат, когда такая короткая запись НЕ используется.

И наконец, можно поменять все направления местами используя ключевое слово `left to right direction`. Для этого можно сравнить предыдущую диаграмму с диаграммой, представленной ниже:

```
@startuml
left to right direction
-left->
-right->
-up->
-down->
@enduml
```



3.27 Ассоциация классов

Вы можете задать *ассоциацию класса* после того, как была задана связь между двумя классами, как в примере:

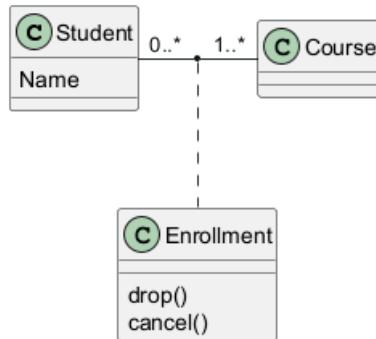
```
@startuml
class Student {
    Name
}
Student "0..*" - "1..*" Course
```



(Student, Course) .. Enrollment

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

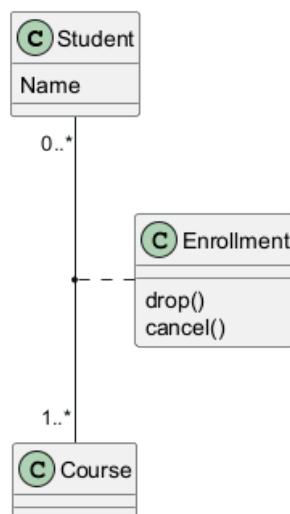
@enduml



Вы можете задать это в другом направлении:

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

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



3.28 Ассоциация в одном и том же классе

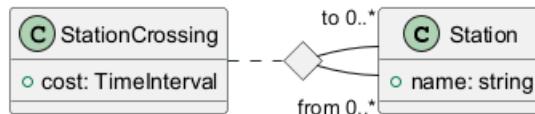
```
@startuml
class Station {
    +name: string
}
```



```
class StationCrossing {
    +cost: TimeInterval
}
```

<> diamond

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



[Ref. Incubation: Associations]

3.29 Параметры отображения

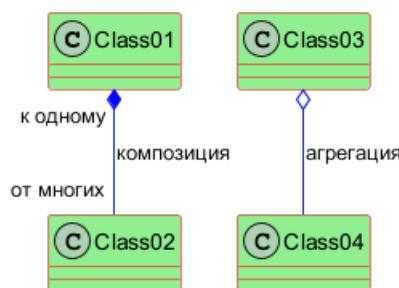
Вы можете использовать команду `skinparam` для глобального изменения шрифтов и цветов различных элементов диаграммы.

Вы можете использовать данную команду :

- При определении диаграммы, как любую другую команду,
- При подключении файла,
- В конфигурационном файле, указанном в командной строке или в задании ANT.

`@startuml`

```
'  
skinparam class {  
    '  
    BackgroundColor LightGreen  
    '  
    ArrowColor Blue  
    '  
    BorderColor Red  
}  
  
'  
skinparam stereotypeBackgroundColor Magenta  
  
Class01 " *-- " " Class02 :  
  
Class03 o-- Class04 :  
  
@enduml
```



3.30 Параметры отображения для шаблонов

Расширяя пример выше, можно задавать специальные параметры для определенных шаблонов (и применять эти шаблоны затем к тем или иным классам).

```
@startuml

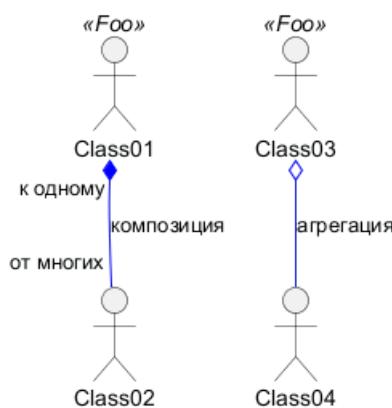
skinparam class {
    'BackgroundColor LightGreen
    'ArrowColor     Blue
    'BorderColor   Red
    '              ,           Foo
    BackgroundColor<<Foo>> Green
    '              ,           Foo
    BorderColor<<Foo>> Orange
}

skinparam stereotypeCBackgroundColor Magenta
'              ,           Foo
skinparam stereotypeCBackgroundColor<< Foo >> Gray

Class01 <<Foo>>
Class03 <<Foo>>

Class01 "      " *-- "      " Class02 :
Class03 o-- Class04 :

@enduml
```



Any of the spaces shown as ‘_’ below will cause **all** skinparams to be ignored, see [discord discussion](<https://discord.com/channels/1000000000000000000>) and [issue #1932](<https://github.com/plantuml/plantuml/issues/1932>):

- ‘BackgroundColor_«Foo» Wheat’
- ‘skinparam stereotypeCBackgroundColor_«Foo» DimGray’

3.31 Цветовой градиент

Можно объявить индивидуальный цвет для классов или примечаний, используя символ #.

Можно использовать как стандартные названия цветов, так и их RGB-коды в различных вариантах записи.



Так же, возможно задания градиента для фона, задав вместо одного цвета пару цветов и разделив их одним из символов градиента (разные символы отвечают за разные направления градиента):

- |,
- /,
- \,
- -

Для примера:

```
@startuml

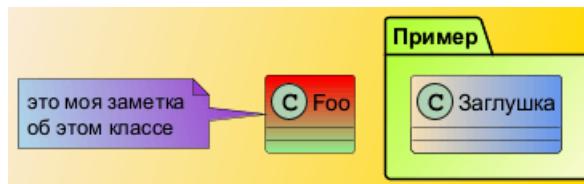
skinparam backgroundcolor AntiqueWhite/Gold
skinparam classBackgroundColor Wheat|CornflowerBlue

class Foo #red-lightgreen
note left of Foo #lightblue\9932CC

end note

package      #GreenYellow/LightGoldenRodYellow {
    class
}

@enduml
```



3.32 Помощь в расположении классов

Иногда диаграмма, формируемая по умолчанию выходит не совсем удачной...

Вы можете объединять несколько классов вместе используя ключевое слово `together`, тогда рендер постараётся расположить объединенные таким образом классы рядом друг с другом так, как если бы они находились в одном пакете.

Для этих же целей можно связать классы между собой связями и скрыть эти связи установив им соответствующее свойство `hidden`. Тогда рендер также постараётся расположить такие связанные классы вместе (причем длиной такой скрытой связи можно регулировать расстояние между классами, а направлением связи можно регулировать взаимное расположение).

```
@startuml

class Bar1
class Bar2
together {
    class Together1
    class Together2
    class Together3
}

Together2 -[hidden]--> Bar1
Bar1      -[hidden]>   Bar2

@enduml
```



3.33 Разделение больших файлов

Иногда могут получиться очень большие файлы изображений.

Чтобы разделить создаваемое изображение на несколько файлов (страниц), Вы можете использовать команду `page (hpages)x(vpages)`, где:

- `hpages` - это число горизонтальных страниц,
- `vpages` - это число вертикальных страниц..

С этой командой также можно использовать команды `skinparam` страниц, и размеров их границы (смотри пример).

для настроек цвета разделённых

```

@startuml
    page 2x2
    skinparam pageMargin 10
    skinparam pageExternalColor gray
    skinparam pageBorderColor black

    class

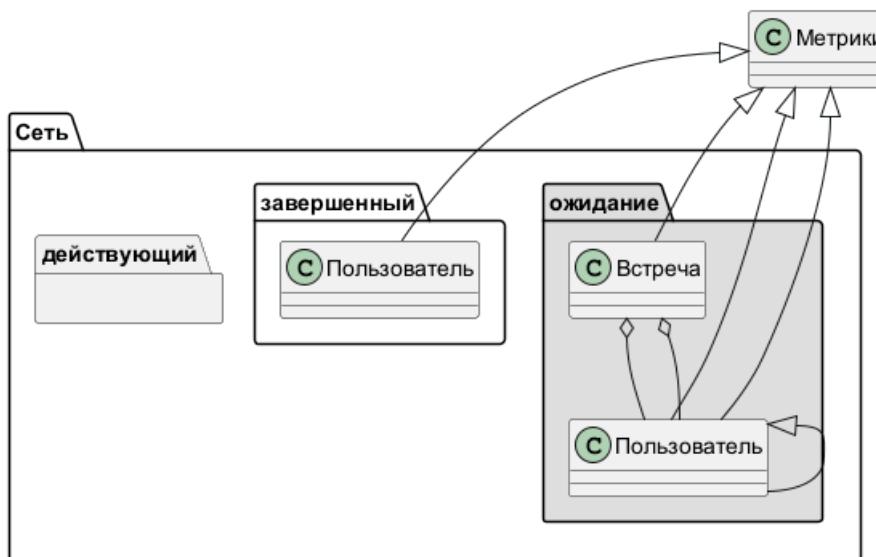
    namespace .      #DDDDDD {
        .
        <|-- .
        o-- .
        .
        <|- .
    }

    namespace .      {
        .
        <|- .
        <|-- .
        .
        o-- .
        <|-- .
    }

    @enduml

```



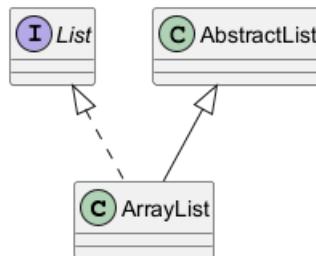


3.34 Наследование и Реализация (Имплементация)

Вы можете использовать ключевые слова `extends` и `implements` для создания связей наследования и реализации (имплементации).

```

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



[Ref. [QA-2239](https://forum.plantuml.net/2239/is-multiple-inheritance-or-implementation-possible)]

3.35 Изменение цвета и стиля связей или стрелок (синтаксис квадратных скобочек)

Есть два разных синтаксиса для изменения цвета и стиля связей и стрелок. Возможности этого синтаксиса квадратных скобочек несколько больше, чем возможности встраиваемого синтаксиса (подробнее о нем будет ниже), так как здесь можно полностью скрывать связи, а также задавать их толщину.

3.35.1 Стиль линий

Можно задавать следующие стили для связей: `bold` - жирный, `dashed` - штриховой, `dotted` - пунктирный, `hidden` - скрытый, `plain` - непрерывный:

- пример для каждого стиля без заметок на связях:

```

@startuml
title
class foo
class bar
bar1 : [bold]
bar2 : [dashed]
bar3 : [dotted]
bar4 : [hidden]
  
```



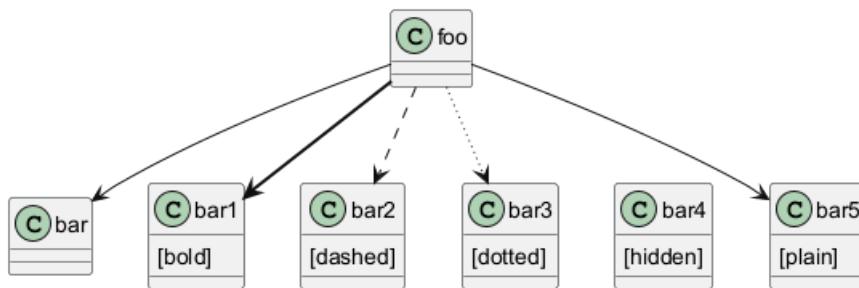
```

bar5 : [plain]

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

```

Синтаксис квадратных скобочек без заметок



- пример для каждого стиля с заметками на связях:

```

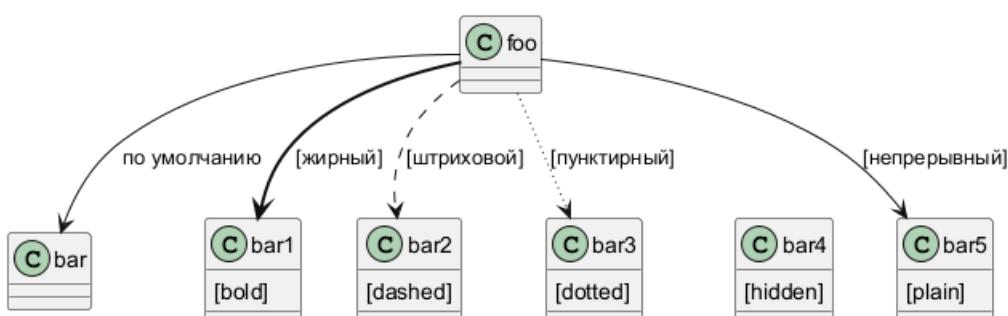
@startuml
title
class foo
class bar
bar1 : [bold]
bar2 : [dashed]
bar3 : [dotted]
bar4 : [hidden]
bar5 : [plain]

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

@enduml

```

Синтаксис квадратных скобочек с заметками



[Adapted from QA-4181]



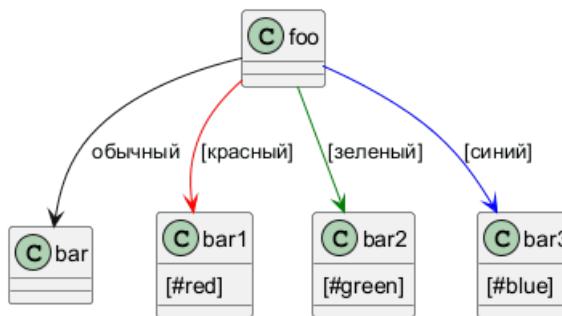
3.35.2 Цвет линий

Можно задавать цвет для связей:

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

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

Задание цвета при помощи синтаксиса квадратных скобочек



3.35.3 Толщина линий

Можно задать толщину связей:

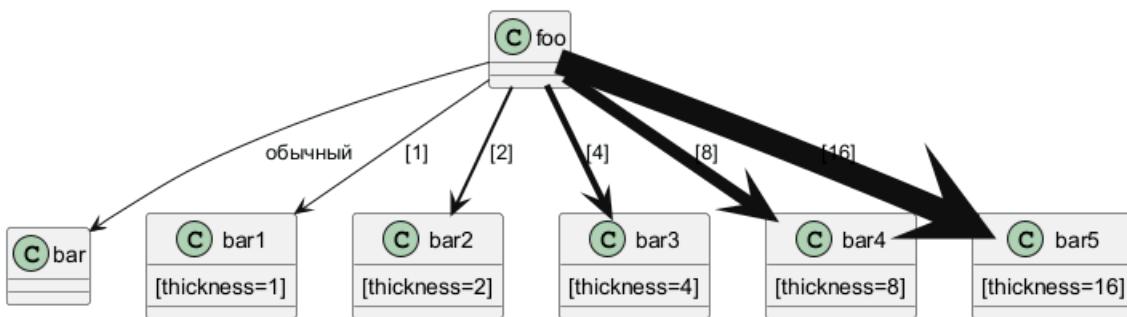
```
@startuml
title
class foo
class bar
bar1 : [thickness=1]
bar2 : [thickness=2]
bar3 : [thickness=4]
bar4 : [thickness=8]
bar5 : [thickness=16]

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
```



Задание толщины при помощи синтаксиса квадратных скобочек



[Ref. QA-4949]

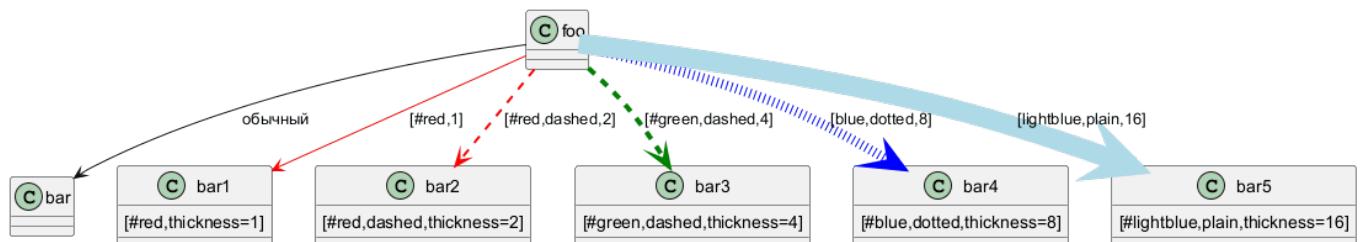
3.35.4 Смешивание различных параметров

```

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

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

Смешивание параметров при использовании синтаксиса квадратных скобочек



3.36 Изменение цвета и стиля связей или стрелок (встраиваемый синтаксис)

Возможности этого синтаксиса немного меньше, чем синтаксиса квадратных скобочек, который был рассмотрен выше. В частности, здесь нельзя задавать толщину линий, а также нельзя полностью скрывать линии.

Вы можете изменить цвет или стиль индивидуально для каждой связи или стрелочки используя следующий встраиваемый синтаксис:

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

где:

- # - начало режима конфигурирования,
- color - цвет стрелки,



- ; - разделитель между конфигурационными параметрами,
- line.type - тип линии. type может принимать значения: bold - жирная, dashed - штриховая, dotted - пунктирная,
- text:color - цвет текста заметки на линии.

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



Пример установки одних и тех же параметров при помощи двух различных синтаксисов можно посмотреть на примере ниже:

```
@startuml

class foo1
class foo2

foo1 -[#red,bold]>     bar1
foo1 -[#blue,dashed]-> bar2 :

foo2 -> bar3 #red;line.bold
foo2 --> bar4 #blue;line.dashed :
```

@enduml



[See similar feature on deployment]

3.37 Изменение цвета и стиля класса (встраиваемый синтаксис)

Вы можете изменить цвет или стиль индивидуально для каждой стрелочки используя следующие 2 варианта встраиваемого синтаксиса:

1. `#color ##[style]color`

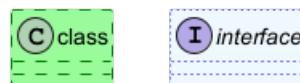
где:

- `#color` - цвет фона элемента,



- `##[style]color` - стиль отображения и цвет линий элемента.

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



[Ref. QA-1487]

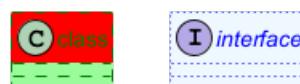
2. `#[color|back:color];header:color;line:color;line.[bold|dashed|dotted];text:color`

где:

- `#` - начало режима конфигурирования,
- `color` или `back:color` - основной цвет фона,
- `;` - разделитель между конфигурационными параметрами,
- `header:color` - дополнительный цвет фона для заголовка,
- `line:color` - цвет для контуров элемента,
- `line.[type]` - тип линий, из которых состоит контур элемента (type может принимать значения: `bold` - жирный, `dashed` - штриховой, `dotted` - пунктирный),
- `text:color` - цвет текста.

Как уже упоминалось выше, можно задавать градиенты для фона, если указывать вместо одного цвета два разных цвета, разделяя их знаками `\`, `!`, `/` или `-` (они отвечают за разное направление градиента). Например: `header:blue/red` или `back:DD0000-00FFFF`.

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



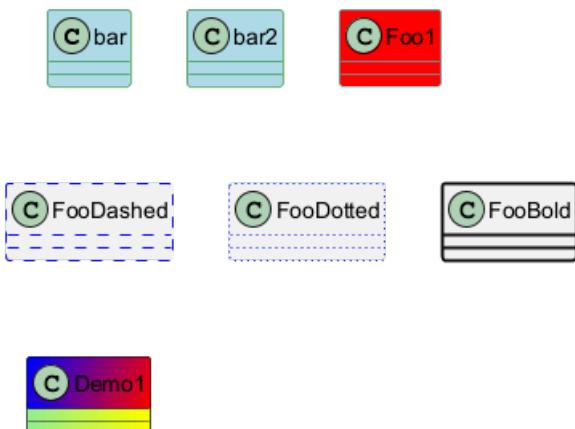
Еще один пример:

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

class Foo1 #back:red;line:00FFFF
```



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



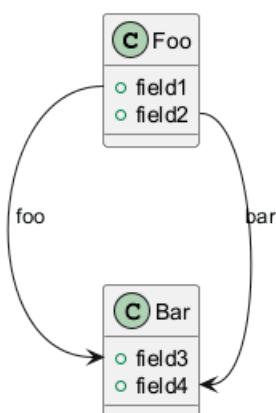
[Ref. QA-3770]

3.38 Связи от/на члены класса

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

class Bar {
+ field3
+ field4
}

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



[Ref. QA-3636]

```
@startuml
left to right direction

class User {
```



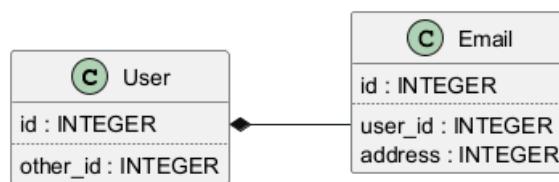
```

id : INTEGER
..
other_id : INTEGER
}

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

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

```



[Ref. QA-5261]

3.39 Группировка стрелок, ведущих к одному элементу

Вы можете включить объединение окончания стрелок, сходящихся к одному элементу, используя команду `skinparam groupInheritance X`, где X - это минимальное количество стрелок, при котором будет включаться данный режим.

3.39.1 Установка параметра в значение 1 (без группировки)

```

@startuml
skinparam groupInheritance 1

```

A1 <|-- B1

A2 <|-- B2

A2 <|-- C2

A3 <|-- B3

A3 <|-- C3

A3 <|-- D3

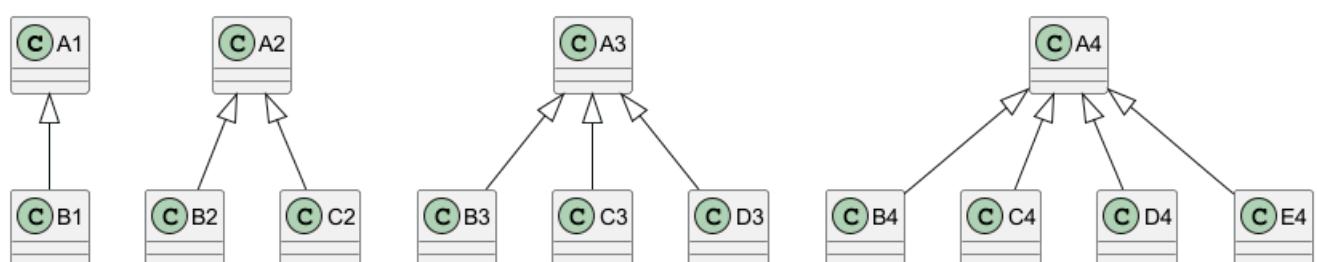
A4 <|-- B4

A4 <|-- C4

A4 <|-- D4

A4 <|-- E4

@enduml



3.39.2 Установка параметра в значение 2 (группировка если 2 и более)

```
@startuml
skinparam groupInheritance 2
```

A1 <|-- B1

A2 <|-- B2

A2 <|-- C2

A3 <|-- B3

A3 <|-- C3

A3 <|-- D3

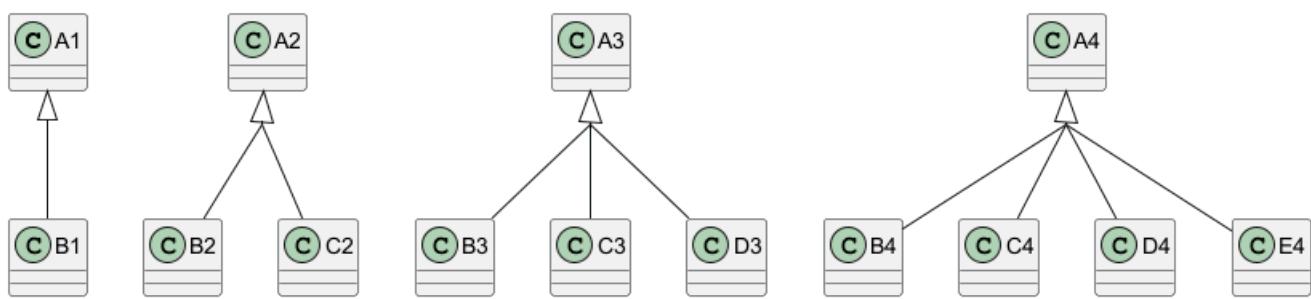
A4 <|-- B4

A4 <|-- C4

A4 <|-- D4

A4 <|-- E4

```
@enduml
```



3.39.3 Установка параметра в значение 3 (группировка если 3 и более)

```
@startuml
skinparam groupInheritance 3
```

A1 <|-- B1

A2 <|-- B2

A2 <|-- C2

A3 <|-- B3

A3 <|-- C3

A3 <|-- D3

A4 <|-- B4

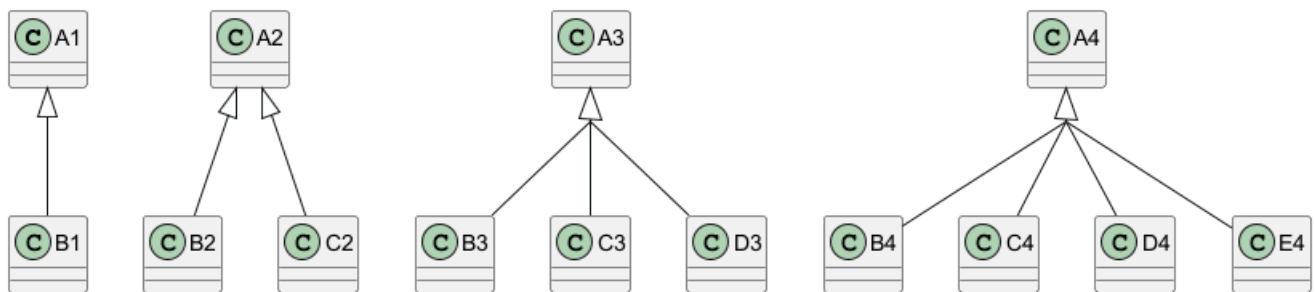
A4 <|-- C4

A4 <|-- D4

A4 <|-- E4

```
@enduml
```





3.39.4 Установка параметра в значение 4 (группировка если 4 и более)

```
@startuml
skinparam groupInheritance 4
```

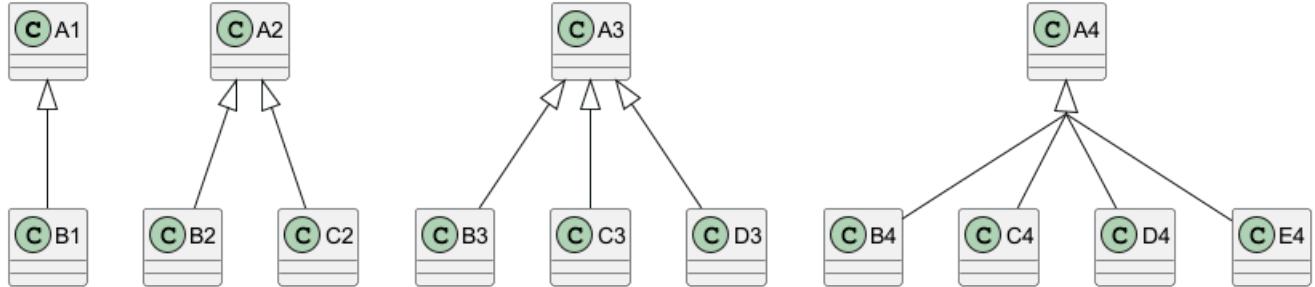
A1 <|-- B1

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

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

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

```
@enduml
```



[Ссылка на QA-3193, и дефект QA-13532]

3.40 Display JSON Data on Class or Object diagram

3.40.1 Simple example

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





JSON	
fruit	Apple
size	Large
color	Red
	Green

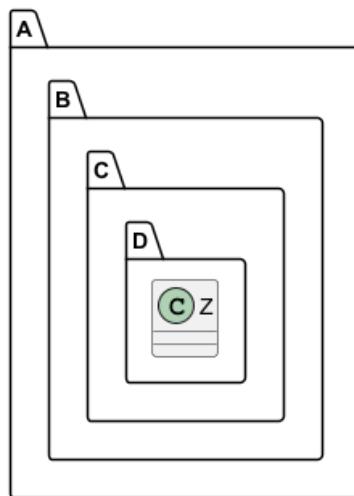
[Ref. QA-15481]

For another example, see on JSON page.

3.41 Packages and Namespaces Enhancement

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

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

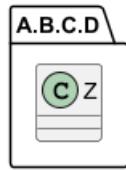


```
@startuml
set separator none
class A.B.C.D.Z {
}
@enduml
```



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

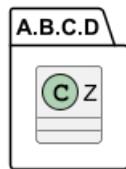




```

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

```



[Ref. GH-1352]

3.42 Qualified associations

3.42.1 Minimal example

```

@startuml
class class1
class class2

class1 [Qualifier] - class2
@enduml

```



[Ref. QA-16397, GH-1467]

3.42.2 Another example

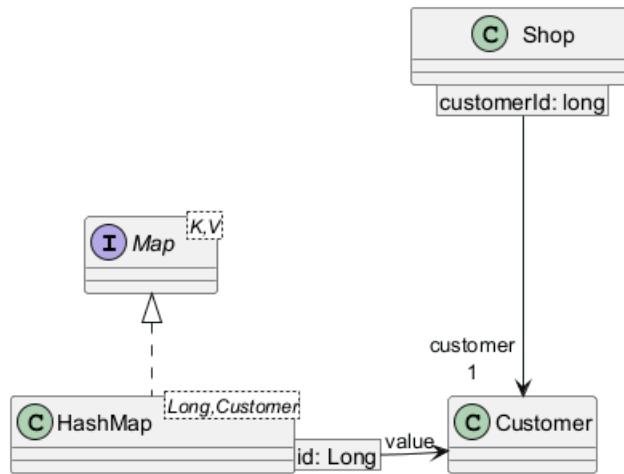
```

@startuml
    interface Map<K,V>
    class HashMap<Long, Customer>

    Map <|.. HashMap
    Shop [customerId: long] ---> "customer\n1" Customer
    HashMap [id: Long] -r-> "value" Customer
@enduml

```





3.43 Change diagram orientation

You can change (whole) diagram orientation with:

- top to bottom direction (by default)
- left to right direction

3.43.1 Top to bottom (by default)

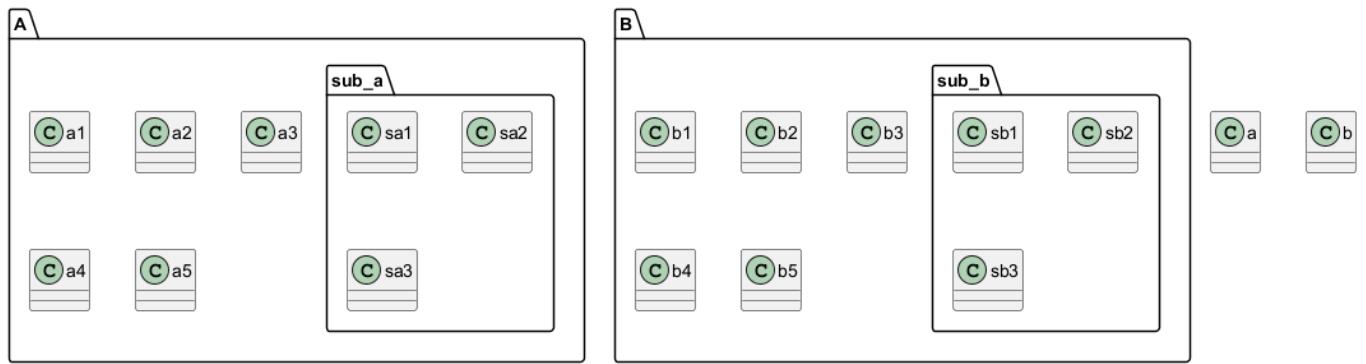
3.43.2 With Graphviz (layout engine by default)

The main rule is: Nested element first, then simple element.

```

@startuml
class a
class b
package A {
    class a1
    class a2
    class a3
    class a4
    class a5
    package sub_a {
        class sa1
        class sa2
        class sa3
    }
}
package B {
    class b1
    class b2
    class b3
    class b4
    class b5
    package sub_b {
        class sb1
        class sb2
        class sb3
    }
}
@enduml
  
```





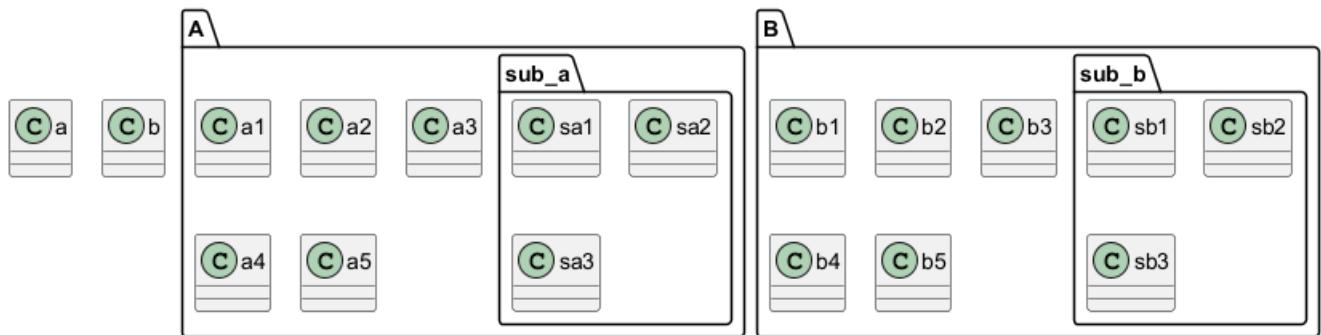
3.43.3 With Smetana (*internal layout engine*)

The main rule is the opposite: **Simple element first, then nested element.**

```
@startuml
!pragma layout smetana
class a
class b
package A {
    class a1
    class a2
    class a3
    class a4
    class a5
    package sub_a {
        class sa1
        class sa2
        class sa3
    }
}

package B {
    class b1
    class b2
    class b3
    class b4
    class b5
    package sub_b {
        class sb1
        class sb2
        class sb3
    }
}
@enduml
```





3.43.4 Left to right

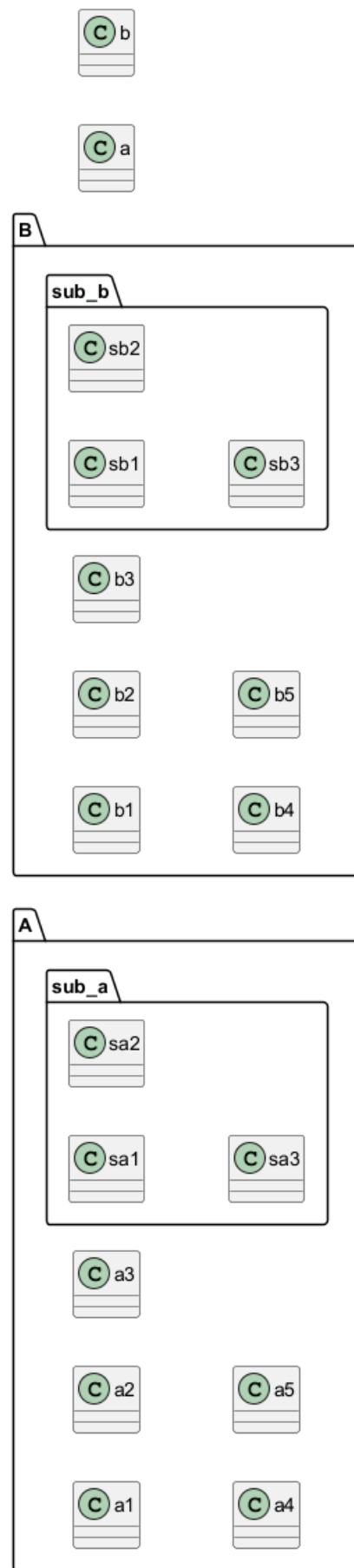
3.43.5 With Graphviz (*layout engine by default*)

```

@startuml
left to right direction
class a
class b
package A {
    class a1
    class a2
    class a3
    class a4
    class a5
    package sub_a {
        class sa1
        class sa2
        class sa3
    }
}
package B {
    class b1
    class b2
    class b3
    class b4
    class b5
    package sub_b {
        class sb1
        class sb2
        class sb3
    }
}
@enduml

```



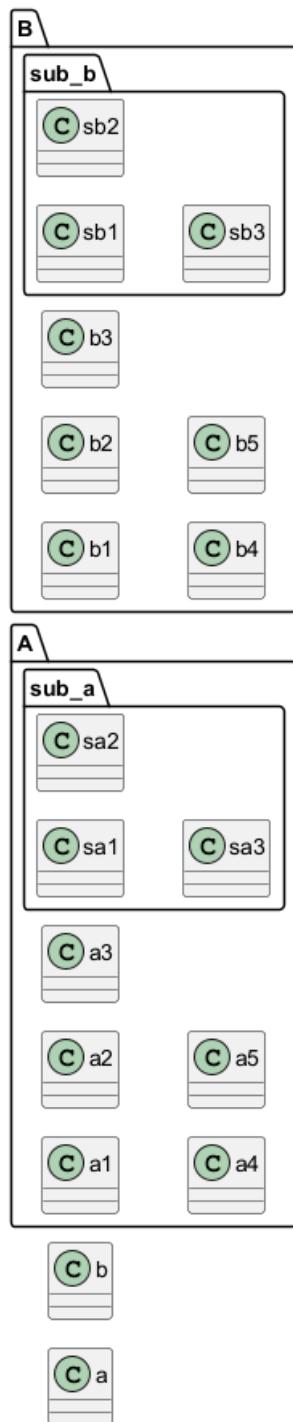


3.43.6 With Smetana (*internal layout engine*)

```
@startuml
!pragma layout smetana
left to right direction
class a
class b
package A {
    class a1
    class a2
    class a3
    class a4
    class a5
    package sub_a {
        class sa1
        class sa2
        class sa3
    }
}

package B {
    class b1
    class b2
    class b3
    class b4
    class b5
    package sub_b {
        class sb1
        class sb2
        class sb3
    }
}
@enduml
```





4 Диаграмма объектов

Диаграмма объектов - это графическое представление, демонстрирующее объекты и их взаимосвязи в определенный момент времени. Она дает представление о структуре системы, фиксируя статический вид присутствующих экземпляров и их связей.

PlantUML предлагает простой и интуитивно понятный способ создания объектных диаграмм с помощью обычного текста. Его удобный синтаксис позволяет быстро создавать диаграммы, не прибегая к сложным средствам графического интерфейса. Кроме того, на форуме PlantUML пользователи могут обсуждать, обмениваться опытом и обращаться за помощью, что способствует формированию коллективного сообщества. Выбирая PlantUML, пользователи получают преимущества как от эффективности построения диаграмм на основе markdown, так и от поддержки активного сообщества.

4.1 Определение объектов

Вы можете определить объекты используя ключевое слово `object`.

```
@startuml
object
object " " as o2
@enduml
```



4.2 Взаимосвязи между объектами

Взаимосвязи между объектами (согласно нотации UML) задаются с помощью следующих символов:

Тип	Символ	Назначение
Расширение	< --	Специализация класса в иерархии
Реализация	< ..	Реализация интерфейса классом
Композиция	*--	Часть не может существовать без целого
Агрегация	o--	Часть может существовать независимо от целого
Зависимость	-->	Объект использует другой объект
Зависимость	...>	Более слабая форма зависимости

4.2.1 Если рассматривать более универсально,

- есть два типа линий:
 - --- рисует сплошную линию
 - ... - рисует штриховую линию
- к любой стороне (или сразу к обоим сторонам) линии, соединяющей два элемента, можно добавить разные наконечники:
 - < - рисует заостренный наконечник стрелочки
 - <| или ^ - рисует наконечник стрелочки в виде треугольника
 - * - рисует неконечник стрелочки в виде сплошного ромба
 - o - рисует неконечник стрелочки в виде полого ромба
 - # - рисует неконечник стрелочки в виде полого квадратика
 - x - рисует неконечник стрелочки в виде крестика
 - } - рисует неконечник стрелочки в виде обратного треугольника
 - + - рисует неконечник стрелочки в виде кружочка с крестиком внутри

Чтобы добавить к связи текстовую метку, используйте символ `:` и за ним разместите текст метки:

- `object1 --> object2 : Метка`



Чтобы добавить текстовую метку к началу и/или к концу связи, разместите текст метки в двойных кавычках перед описанием связи и/или после описания связи:

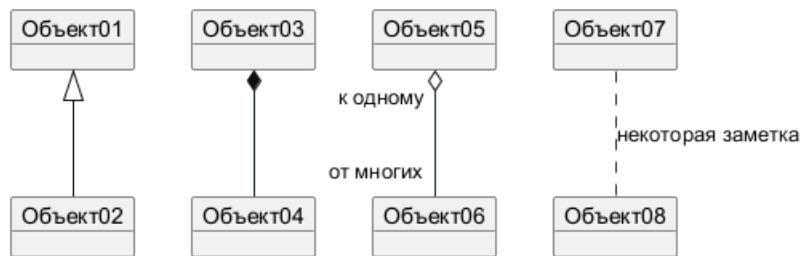
- object1 " " --> " " object2

4.2.2 Примеры:

```
@startuml
object 01
object 02
object 03
object 04
object 05
object 06
object 07
object 08

01 <|-- 02
03 *-- 04
05 " " o-- " " 06
07 .. 08 :

@enduml
```



4.2.3 Ещё больше примеров можно посмотреть:

- по созданию связей - в документации к Диаграмме классов, в разделе Взаимосвязи между классами
- по размещению меток - в документации к Диаграмме классов, в разделе Метки на взаимосвязях

4.3 Пример ассоциации объектов

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

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





4.4 Добавление полей

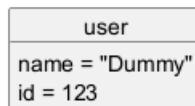
Для добавления полей (свойств) Вы можете использовать символ `:`. Для этого каждый раз указывайте имя объекта, затем символ `:` и затем имя поля, который Вы хотите добавить в этот объект.

```
@startuml
```

```
object user
```

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

```
@enduml
```

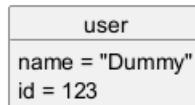


Чтобы не указывать каждый раз имя объекта, для которого Вы хотите добавить поле (свойство), можно все такие поля для одного объекта указать сразу перечислив их все внутри фигурных скобок `{}`.

```
@startuml
```

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

```
@enduml
```



4.5 Общие с диаграммами классов функции

- Видимость
- Задание меток
- Использование пакетов
- Стилизование вывода

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
```

CapitalCity	
UK	London
USA	Washington
Germany	Berlin

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

Map Country => CapitalCity	
UK	London
USA	Washington
Germany	Berlin

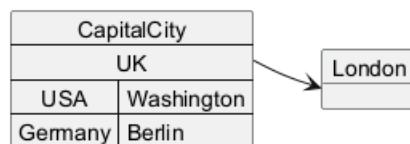
```
@startuml
map "map: Map<Integer, String>" as users {
    1 => Alice
    2 => Bob
    3 => Charlie
}
@enduml
```

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

And add link with object.

```
@startuml
object London

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



```
@startuml
```



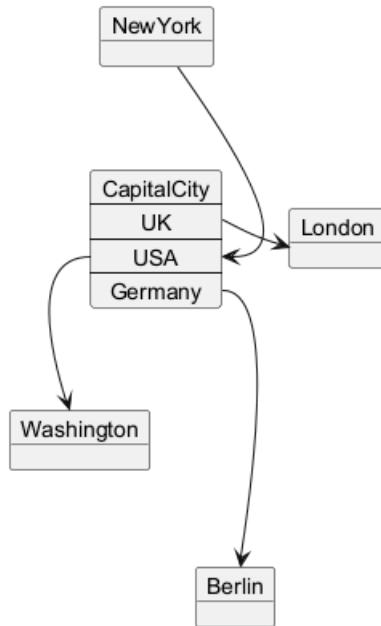
```

object London
object Washington
object Berlin
object NewYork

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

NewYork --> CapitalCity::USA
@enduml

```



[Ref. #307]

```

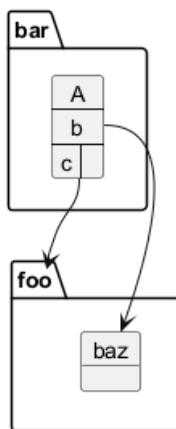
@startuml
package foo {
    object baz
}

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

A::c --> foo
@enduml

```





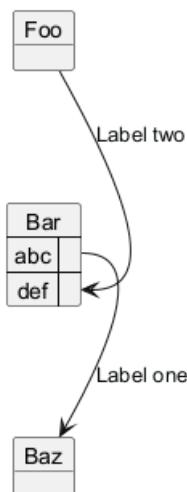
[Ref. QA-12934]

```

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

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

```



[Ref. #307]

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

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

```

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

map Kick.Off {
}
map task.1 {

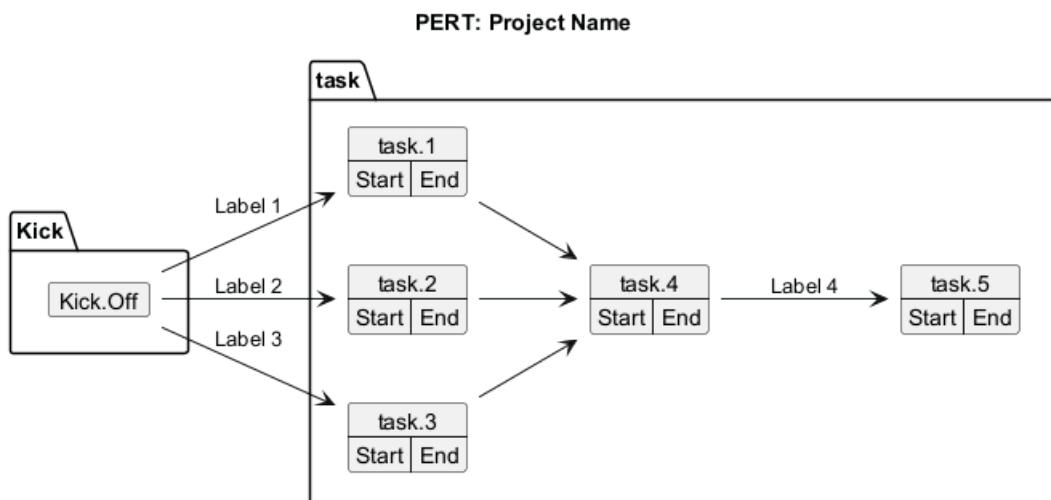
```



```

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

```



[Ref. QA-12337]

4.8 Display JSON Data on Class or Object diagram

4.8.1 Simple example

```

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

```





JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

For another example, see on JSON page.



5 Диаграмма деятельности

5.1 Простая деятельность

Вы можете использовать (*) для начальных и конечных точек диаграммы деятельности.

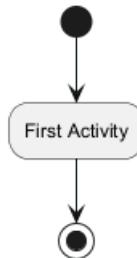
В некоторых случаях, вы можете использовать (*top) чтобы указать что начальная точка должна быть в верху диаграммы.

Используйте --> для стрелок.

```
@startuml
```

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

```
@enduml
```



5.2 Метка на стрелках

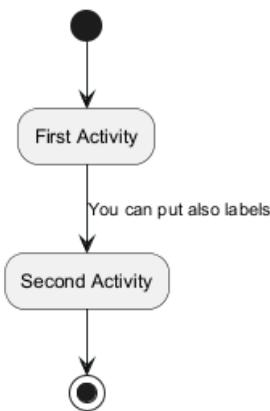
По умолчанию, стрелка начинается с последней использованной активности.

Вы можете пометить стрелку при помощи скобок [и] сразу после определения стрелки.

```
@startuml
```

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

```
@enduml
```



5.3 Изменение направления стрелки

Вы можете использовать -> для горизонтальных стрелок. Возможно задать направление стрелки используя следующий синтаксис:

- -down-> (default arrow)
- -right-> or ->

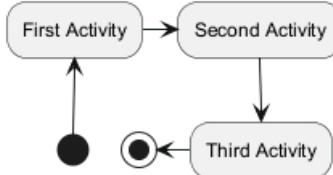


```

• -left->
• -up->
@startuml
(*) -up-> "First Activity"
-right-> "Second Activity"
--> "Third Activity"
-left-> (*)

@enduml

```



5.4 Ветвления

Вы можете использовать ключевые слова `if/then/else` чтобы определять ветви.

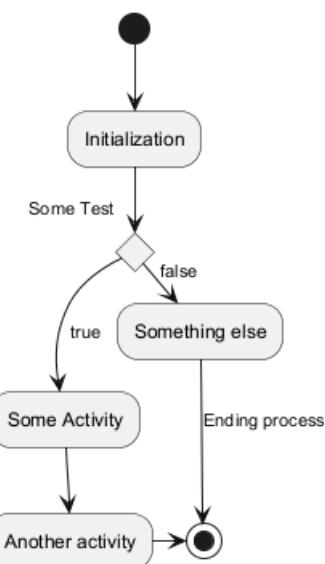
```

@startuml
(*) --> "Initialization"

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

@enduml

```



К сожалению, вам иногда придётся повторять ту же активность в тексте диаграммы:

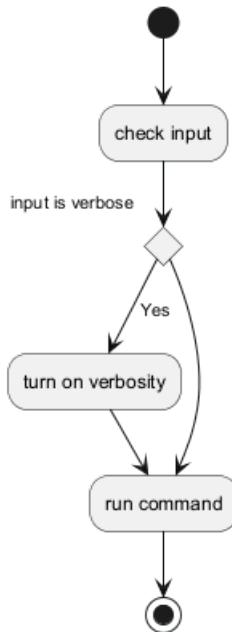
```

@startuml
(*) --> "check input"
If "input is verbose" then

```



```
--> [Yes] "turn on verbosity"
--> "run command"
else
--> "run command"
Endif
-->(*)
@enduml
```



5.5 Больше о ветках

По умолчанию, ветка соединена к последней заданной активности, но возможно переопределить это и задать связь с помощью ключевого слова `if`.

Также возможно создавать вложенные ветки.

```
@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"
```

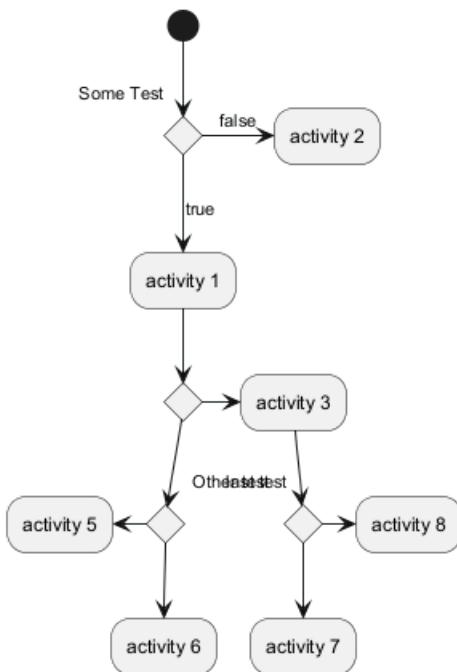


```

else
    -> "activity 8"
endif

@enduml

```



5.6 Синхронизация

Вы можете использовать === code ===, чтобы отобразить барьеры синхронизации.

```
@startuml
```

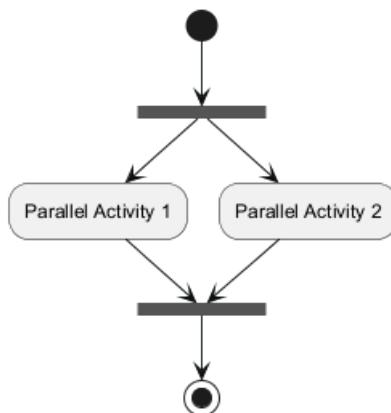
```

(*) --> ===B1===
--> "Parallel Activity 1"
--> ===B2===
 
 ===B1=== --> "Parallel Activity 2"
--> ===B2===

--> (*)

```

```
@enduml
```



5.7 Длинное описание активности

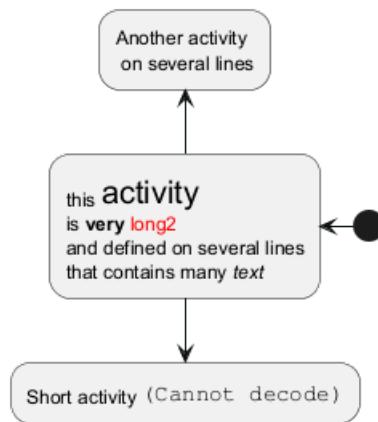
Когда вы задаёте активность, вы можете разделить её описание на несколько линий. Вы также можете добавить \n в описание.

Вы также можете задать короткий код активности с помощью ключевого слова as. Этот код может быть использован позже в описании диаграммы.

```
@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 Заметки

Вы можете добавить заметки к активности используя команды note left, note right, note top or note bottom, сразу после описания активности, к которой вы хотите прикрепить заметку.

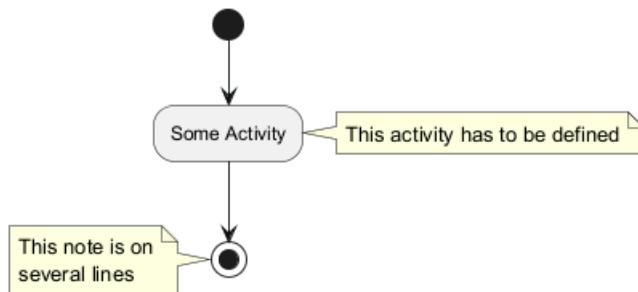
Если вы хотите прикрепить заметку к точке начала, задайте метку в самом начале описания диаграммы.

Вы также можете создать заметку на нескольких линиях, используя ключевое слово 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 Разделы

Вы можете задать раздел используя ключевое слово `partition`, и опционально задать цвет фона для своего раздела (Используя код цвета html или название цвета)

Когда вы задаёте активность, они автоматически попадают в последнюю заданную активность.

Вы можете закрыть раздел используя закрывающую скобку }.

`@startuml`

```

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

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

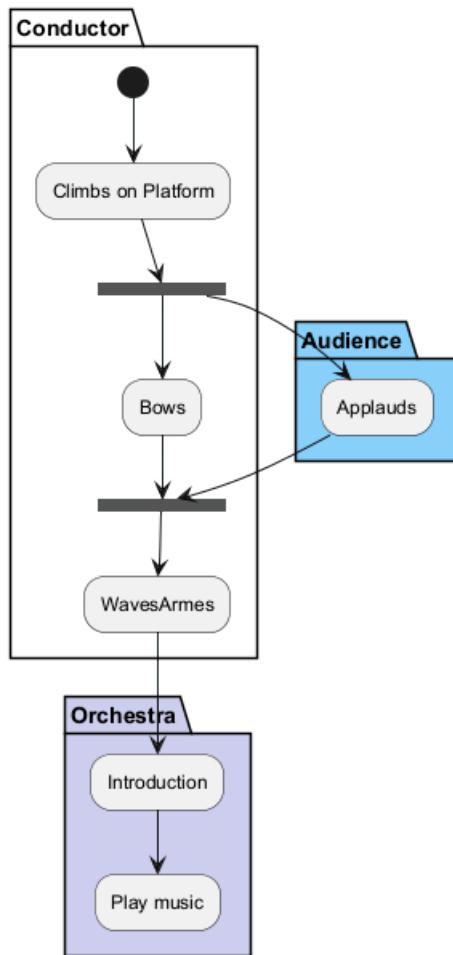
partition Conductor {
    Bows --> === S2 ===
    --> WavesArmes
    Applauds --> === S2 ===
}

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

```

`@enduml`





5.10 Skinparam

Вы можете использовать команду `skinparam` чтобы изменить цвет и шрифт рисования.

Вы можете использовать команду :

- В определении диаграммы, как любую другую команду,
- В подключаемом файле,
- В конфигурационном файле, подставленный в командной строке ANT задания.

Вы можете задать определённый цвет и шрифт для активностей с шаблоном.

`@startuml`

```

skinparam backgroundColor #AFFFFF
skinparam activity {
    StartColor red
    BarColor SaddleBrown
    EndColor Silver
    BackgroundColor Peru
    BackgroundColor<< Begin >> Olive
    BorderColor Peru
    FontName Impact
}

```

```

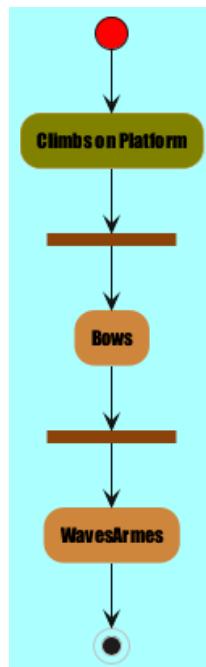
(*) --> "Climbs on Platform" << Begin >>
--> === S1 ===
--> Bows

```



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

@enduml



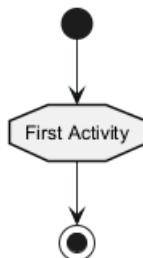
5.11 Восьмиугольник

Вы можете изменить форму активностей на восьмиугольник, используя команду `skinparam activityShape octagon`.

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

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

@enduml



5.12 Полноценный пример

```
@startuml
title Servlet Container

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

if "Page.onSecurityCheck" then
```



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

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

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

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

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

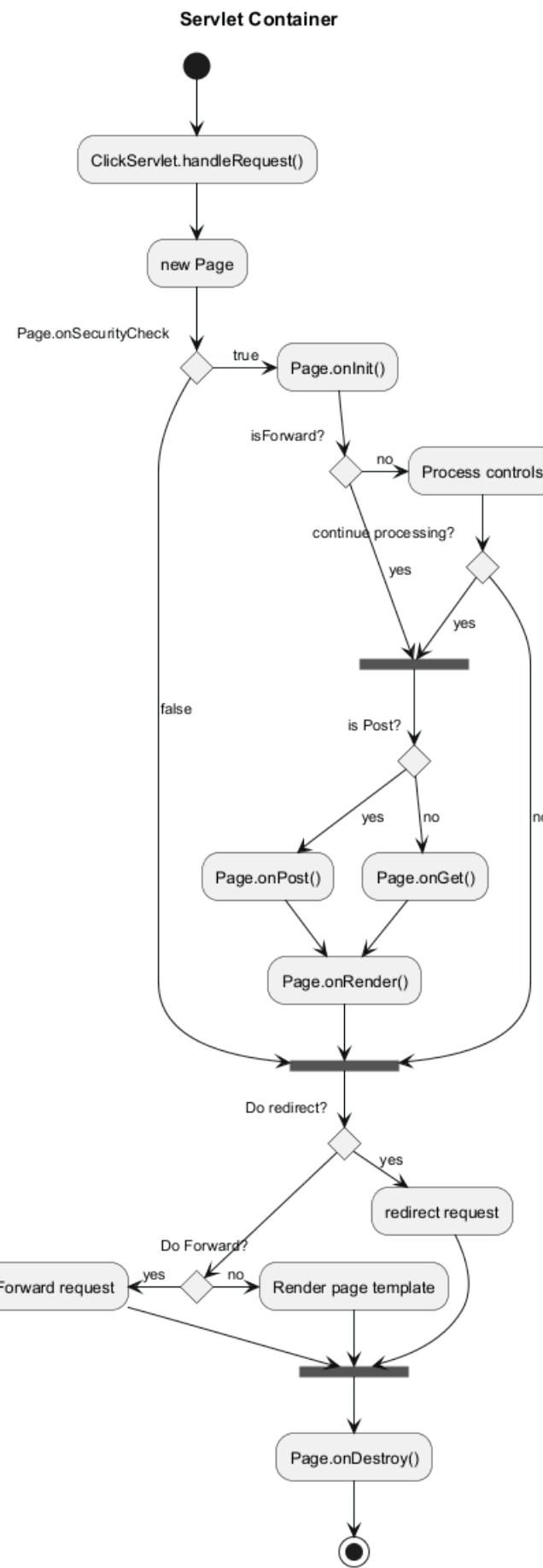
else
-->[false] ===REDIRECT_CHECK===
endif

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

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

@enduml





6 Диаграмма деятельности (новый синтаксис)

Прежний синтаксис, использовавшийся для диаграмм деятельности, имел ряд ограничений и проблем с поддержкой. Признавая эти недостатки, мы представили полностью переработанный синтаксис и реализацию, которые не только удобны для пользователя, но и более стабильны.

6.0.1 Преимущества нового синтаксиса

- Отсутствие зависимости от Graphviz: Как и в случае с диаграммами последовательностей, новый синтаксис исключает необходимость установки Graphviz, упрощая тем самым процесс настройки.
- Простота обслуживания: Интуитивный характер нового синтаксиса облегчает управление и сопровождение диаграмм.

6.0.2 Переход на новый синтаксис

Хотя мы будем продолжать поддерживать старый синтаксис для сохранения совместимости, мы настоятельно рекомендуем пользователям перейти на новый синтаксис, чтобы использовать его расширенные возможности и преимущества.

Сделайте этот переход сегодня и почувствуйте более рациональный и эффективный процесс создания диаграмм с помощью нового синтаксиса диаграмм деятельности.

6.1 Простое действие

Описания действий начинаются со знака : и заканчиваются —;.

Возможно использовать синтаксис creole wiki syntax для форматирования текста.

Действия автоматически связываются по порядку их определения.

```
@startuml
:      !;
:
**      **      ;
@enduml
```



6.2 Старт активности, Стоп активности, Стоп поток, Быстрый стоп

Следующие ключевые слова используются для:

- start - для обозначения начального состояния активности,
- stop - для обозначения конечного состояния активности,
- end - для обозначения конечного состояния потока,
- kill или detach - для быстрого прерывания активности без обозначения каких либо значков.

Использование любого из них не обязательно, диаграммы активности можно строить и без указания начального и/или конечного состояний start, stop или end, как было продемонстрировано на примерах выше. Чтобы прервать какую-либо активность без отображения каких-либо значков, нужно использовать идентичные по своему действию ключевые слова kill или detach.

```
@startuml
start
:      !;
:
```



```
**      **      ;
stop
@enduml
```



```
@startuml
start
:      !;
:
**      **      ;
end
@enduml
```



```
@startuml
if ( ) then
: \nkill;
kill
else
: ;
endif
: \nstop;
stop
@enduml
```



6.3 Условия (оператор if)

Можно использовать ключевые слова `if`, `then`, `else` и `endif`, чтобы добавить добавить проверку условия на диаграмму.

Доступно 3 равнозначных варианта записи условия (квадратными скобками отмечены опциональные поля, которые можно указывать, а можно не указывать):

6.3.1 1 вариант записи:

- `if () then [(])]`
- `else [()]`
- `endif`

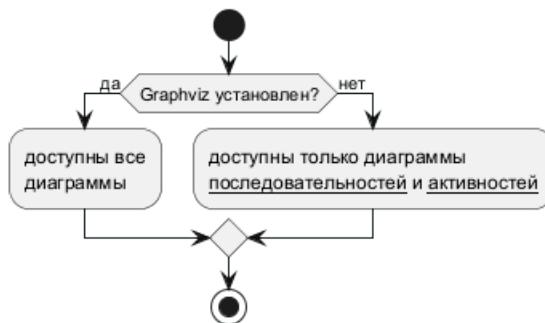
```
@startuml
```

```
start

if (Graphviz      ?) then ( )
:      \n      ;
else ( )
:
--      --      --      --;
endif

stop

@enduml
```



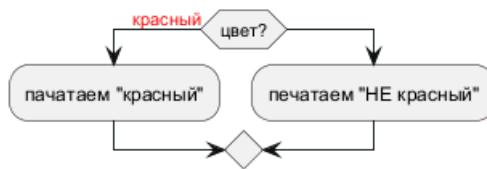
6.3.2 2 вариант записи

- `if () is () then`
- `else [()]`
- `endif`

```
@startuml
```

```
if ( ?) is (<color:red>      ) then
:      "      ";
else
:      "      ";
endif
@enduml
```





6.3.3 3 вариант записи

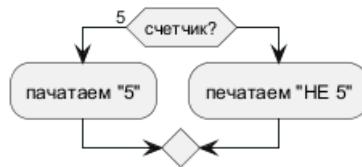
- if () equals () then
- else [()]
- endif

@startuml

```

if ( ?) is (5) then
:      "5";
else
:      " 5";
endif
@enduml

```



[Ref. QA-301]

6.3.4 Ряд последовательных проверок условий (горизонтальное отображение)

Вы можете использовать ключевое слово elseif, чтобы создать несколько последовательных проверок условий (по умолчанию используется режим горизонтального отображения):

@startuml

```

start

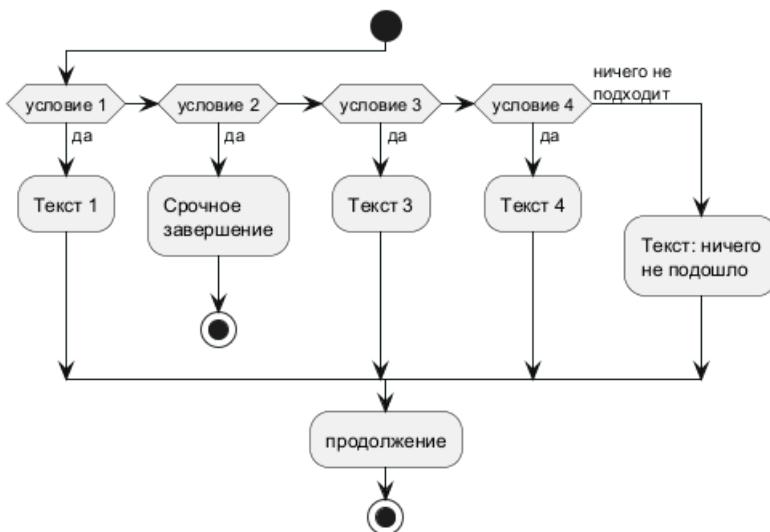
if      ( 1) then ( )
:      1;
elseif ( 2) then ( )
:      \n      ;
stop
elseif ( 3) then ( )
:      3;
elseif ( 4) then ( )
:      4;
else (      \n      )
:      :      \n      ;
endif

:      ;
stop

@enduml

```





6.3.5 Ряд последовательных проверок условий (вертикальное отображение)

Используйте опцию `!pragma useVerticalIf on` для отображения нескольких последовательных проверок условий в вертикальном режиме:

```
@startuml
```

```

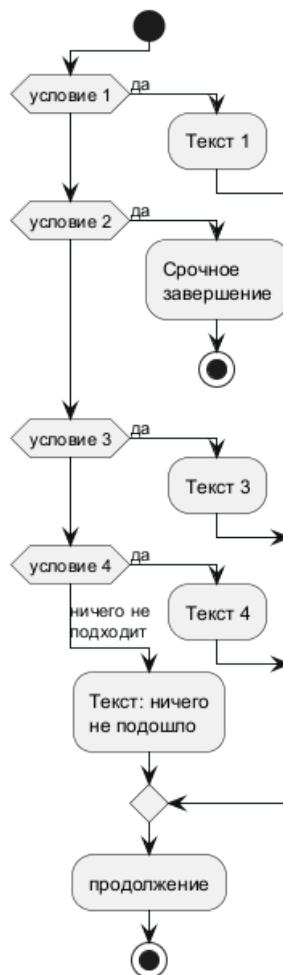
!pragma useVerticalIf on
start

if      (      1) then ( )
:      1;
elseif (      2) then ( )
:      \n      ;
stop
elseif (      3) then ( )
:      3;
elseif (      4) then ( )
:      4;
else (      \n      )
:      :      \n      ;
endif

:      ;
stop
@enduml

```





Также, можно использовать параметр `-P` для установки этой опции для диаграмм по умолчанию из командной строки:

```
java -jar plantuml.jar -PuseVerticalIf=on
```

[Refs. QA-3931, issue-582]

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

6.4 Перебор множества условий для определенного параметра (оператор switch)

Для добавления в диаграмму перебора множества условий для какого-то одного параметра (конструкция `switch-case`), нужно использовать следующие ключевые слова: `switch`, `case`, `endswitch`.

Условно можно изобразить синтаксис этой конструкции следующим образом:

```

switch ( )
  case ( 1 ? )
    :
    1;
  case ( 2 ? )
    :
    2;
  ...
endswitch

@startuml

start
  
```



```

switch ( )
case ( 1 )
:1 ;
case ( 2 )
:2 ;
case ( 3 )
:3 ;
case ( 4 )
:4
-----

```

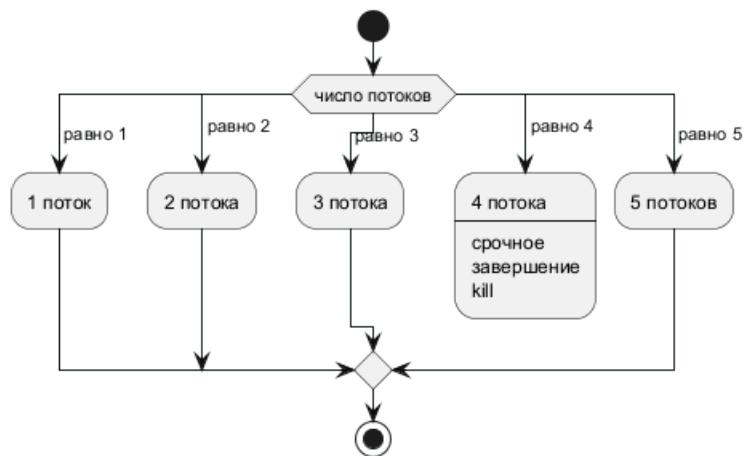
```

kill;
kill
case ( 5 )
:5 ;
endswitch

stop

@enduml

```



6.5 Остановка активности в теле условия

Как уже было продемонстрировано ранее, можно остановить активность на конкретном действии, используя ключевые слова `stop` для отображения остановки деятельности или `end` для отображения остановки потока.

```
@startuml
```

```

if ( ? ) then
: ;
stop

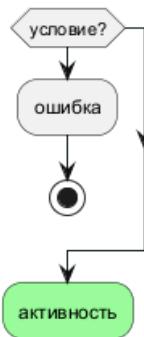
endif

#palegreen : ;

```

```
@enduml
```





Если нужно прервать активность на каком-то конкретном действии без отображения каких-либо значков, нужно использовать идентичные по своему действию ключевые слова `kill` или `detach`:

- `kill`

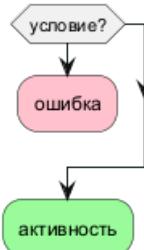
```
@startuml
```

```
if (      ?) then
    #pink :      ;
    kill
```

```
endif
```

```
#palegreen :      ;
```

```
@enduml
```



[Ref. QA-265]

- `detach`

```
@startuml
```

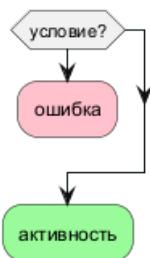
```
if (      ?) then
    #pink :      ;
    detach
```

```
endif
```

```
#palegreen :      ;
```

```
@enduml
```





6.6 Повторяющийся цикл repeat (с проверкой условия в конце цикла)

Для отображения циклов нужно использовать идентичные по своему действию ключевые слова repeat или repeatwhile:

```
@startuml
```

```
start
```

```
repeat
```

```
: ;  
: ;
```

```
repeat while ( ? )
```

```
stop
```

```
@enduml
```



Чтобы добавить подписи к стрелочкам после проверки условия, нужно использовать следующую конструкцию:

- repeat while () is () not ()

```
@startuml
```

```
start
```

```
repeat
```

```
: ;  
: ;
```



```
repeat while (      ?) is ( ) not ( )
stop
@enduml
```



Можно превратить начало цикла в полноценное действие, если после ключевого слова `repeat` в той же строке указать это действие при помощи конструкции: ;

```
@startuml

start

repeat :      ;
:      ;
:      ;
repeat while (      ?)
stop
@enduml
```



Чтобы добавить к циклу обратное действие, используйте ключевое слово `backward` и сразу в той же строке укажите это действие при помощи конструкции: ;

```
@startuml
```



```

start

repeat :      ;
:      ;
:      ;
backward :      ;
repeat while (      ?)

stop

@enduml

```



[Ref. QA-5826]

6.7 Прерывание цикла repeat при помощи ключевого слова break

Чтобы прервать цикл после определенного действия, нужно использовать ключевое слово break

@startuml

```

start

repeat

:      ;

if ( -      ?) then ( )
  #palegreen :      ;
  break
endif

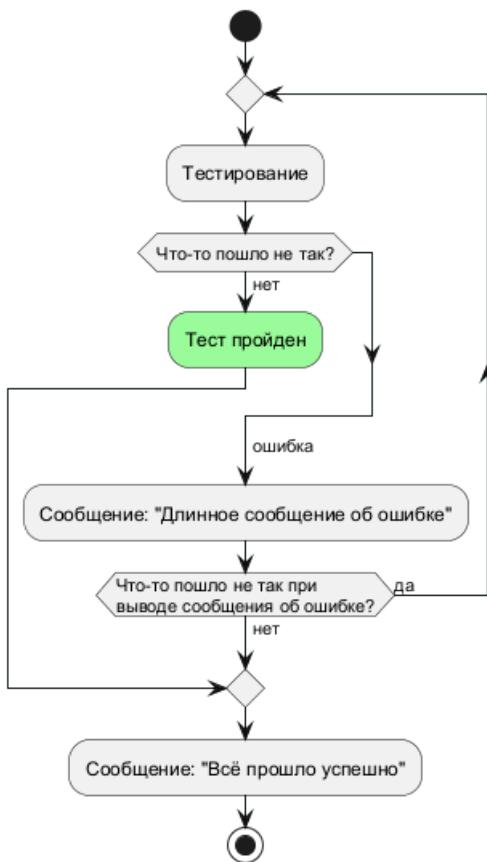
->      ;

:      : "      ";
repeat while ( -      \n      ?) is ( ) not ( )
:      : "      ";
stop

@enduml

```





[Ref. QA-6105]

6.8 Обработка Goto и меток [label, goto]

□ В настоящее время это только экспериментальный вариант □

Для обозначения обработки goto можно использовать ключевые слова label и goto, причем:

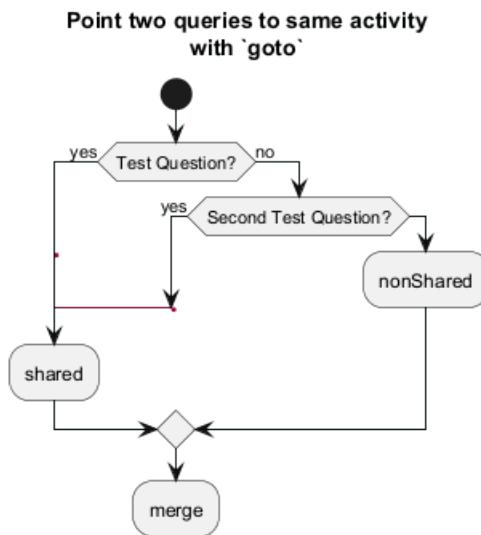
- label <label_name>
- goto <label_name>

```

@startuml
title Point two queries to same activity\nwith `goto`
start
if (Test Question?) then (yes)
'space label only for alignment
label sp_lab0
label sp_lab1
'real label
label lab
:shared;
else (no)
if (Second Test Question?) then (yes)
label sp_lab2
goto sp_lab1
else
:nonShared;
endif
endif
:merge;
@enduml

```





[ссылка на QA-15026, QA-12526 и первоначально QA-1626]

6.9 Повторяющийся цикл while (с проверкой условия в начале цикла)

Для организации цикла с проверкой условия в начале используйте ключевые слова `while` и `end while`.

`@startuml`

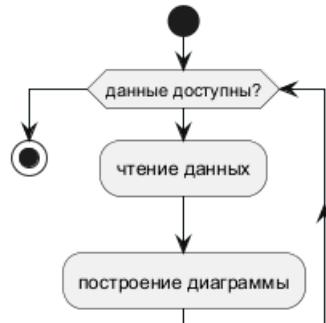
```

start

while (      ?)
:      ;
:      ;
endwhile
  
```

`stop`

`@enduml`



Чтобы добавить пояснения на стрелочки после проверки условия, нужно использовать следующую конструкцию:

```

while (    ) is (      )
...
endwhile (      )
  
```

Пример:

`@startuml`

```
while (      ?) is ( )
```



```

:      ;
:      ;
endwhile ( )

:      ;
@enduml

```



Чтобы добавить к циклу обратное действие, используйте ключевое слово `backward` и сразу в той же строке укажите это действие при помощи конструкции : ;

```

@startuml

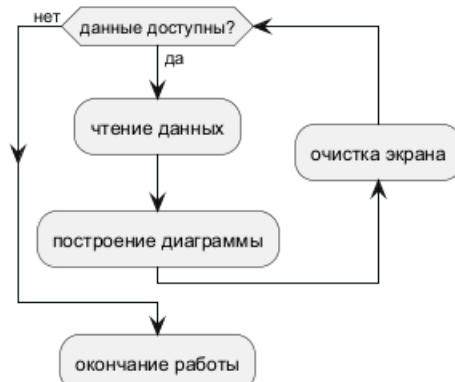
while (      ?) is ( )

:      ;
:      ;

backward :      ;
endwhile ( )

:      ;
@enduml

```



Чтобы прервать выполнение цикла, используйте ключевое слово `break`:

```

@startuml

while (      ?) is ( )

:      ;
@enduml

```



```

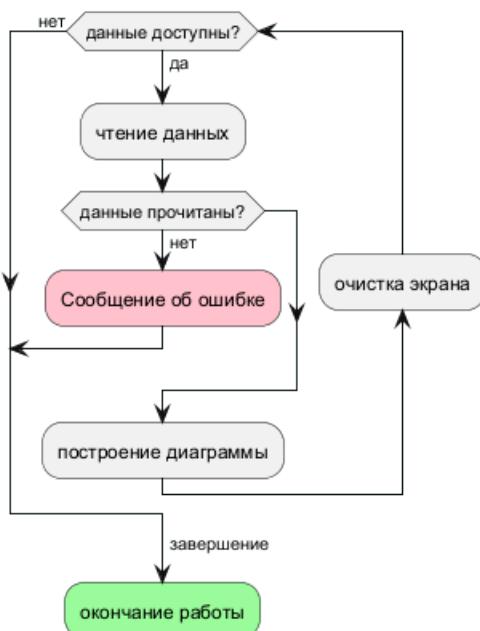
:           ;
if (      ?) then ( )
#pink :           ;
break
endif

:           ;
backward :           ;
endwhile ( )

->       ;

#palegreen :           ;
@enduml

```



6.10 Распараллеливание активности [режим многопоточности: fork, fork again, end fork и merge]

Для разделения одной активности на несколько параллельных активностей можно использовать режим многопоточности. Для этого необходимо использовать ключевые слова: `fork`, `fork again` и `end fork` или `end merge`.

(Также для этих целей можно использовать режим разделения (*split*), который будет рассмотрен более подробно ниже).

6.10.1 Простой режим многопоточности

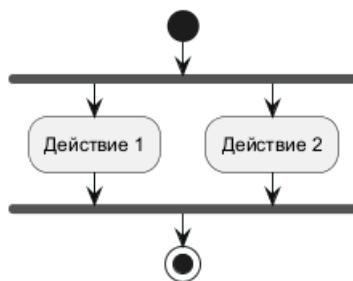
```

@startuml
start
fork
    :   1;
fork again
    :   2;
end fork
stop

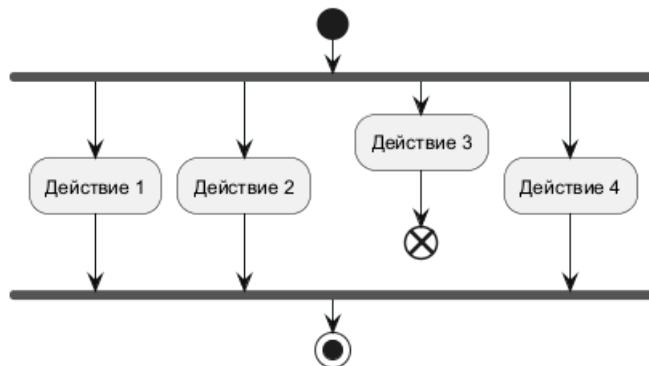
```



@enduml



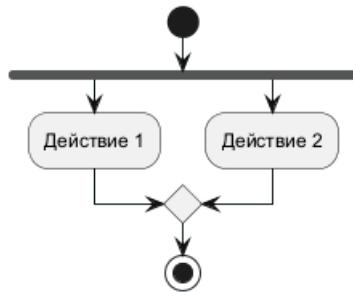
```
@startuml
start
fork
    : 1;
fork again
    : 2;
fork again
    : 3;
end
fork again
    : 4;
end fork
stop
@enduml
```



6.10.2 Режим многопоточности с объединением потоков в конце

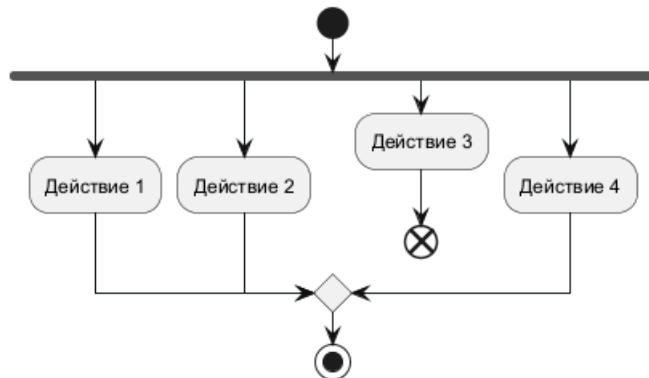
```
@startuml
start
fork
    : 1;
fork again
    : 2;
end merge
stop
@enduml
```





```

@startuml
start
fork
: 1;
fork again
: 2;
fork again
: 3;
end
fork again
: 4;
end merge
stop
@enduml
  
```



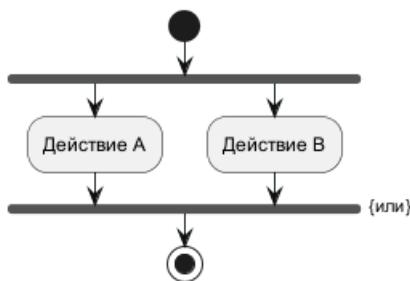
[Ref. QA-13731]

6.10.3 Размещение комментария при возвращении к однопоточному режиму (реализация синтаксиса UML):

```

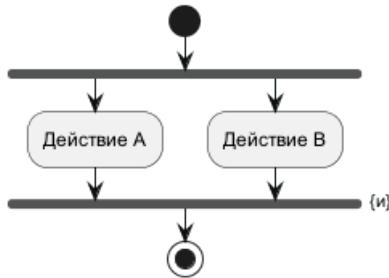
@startuml
start
fork
: A;
fork again
: B;
end fork { }
stop
@enduml
  
```





```

@startuml
start
fork
  :      A;
fork again
  :      B;
end fork { }
stop
@enduml
  
```



[Ref. QA-5346]

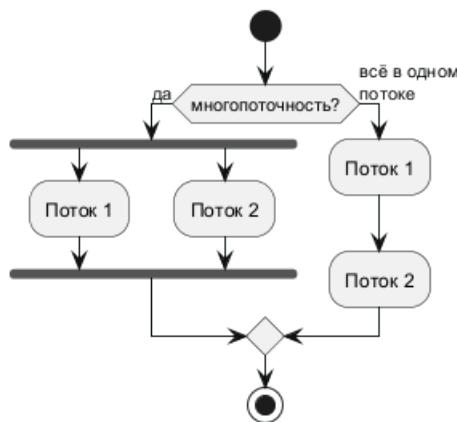
6.10.4 Еще один пример

```

@startuml
start
if (      ?) then ( )
  fork
    :      1;
  fork again
    :      2;
  end fork
else (      \n      )
  :      1;
  :      2;
endif

stop
@enduml
  
```





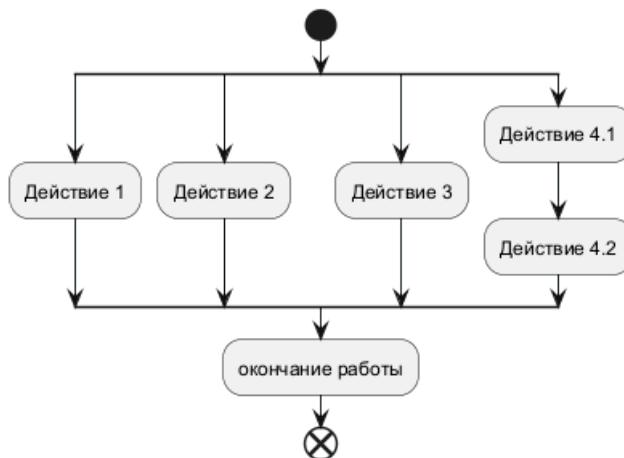
6.11 Распараллеливание активности [режим разделения: split, split again и end split]

6.11.1 Разделение

Чтобы распараллелить активность на несколько одновременно идущих активностей, можно также использовать режим разделения. Для этого необходимо использовать следующие ключевые слова: `split`, `split again` и `end split`.

```

@startuml
start
split
  : 1;
split again
  : 2;
split again
  : 3;
split again
  : 4.1;
  : 4.2;
end split
: ;
end
@enduml
  
```



6.11.2 Несколько входов

При помощи режима разделения можно изобразить несколько независимых начальных действий. Для этого необходимо организовать распараллеливание активности на несколько параллельных и при этом скрыть



входные стрелочки при помощи добавления конструкции `-[hidden]->` перед каждой активностью:

```
@startuml
split
  -[hidden]->
  :    1;
split again
  -[hidden]->
  :    2;
split again
  -[hidden]->
  :    3;
end split
:
@enduml
```



Более сложный пример:

```
@startuml
split
  -[hidden]->
  :    1;
  -> 1;
split again
  -[hidden]->
  :    2.1;
  :    2.2;
  -> 2;
split again
  -[hidden]->
  ()
  -> 3;
end split
:
@enduml
```



[Ref. QA-8662]



6.11.3 Несколько выходов

Таким же образом можно организовать несколько параллельно завершаемых действий. Для этого после каждого такого действия нужно указать ключевые слова `kill` или `detach`:

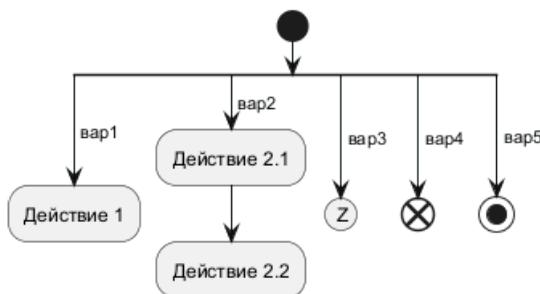
```
@startuml
start
split
:    1;
kill
split again
:    2;
detach
split again
:    3;
kill
end split
@enduml
```



Более сложный пример:

```
@startuml
start
split
-> 1;
:    1;
kill
split again
-> 2;
:    2.1;
:    2.2;
detach
split again
-> 3;
(Z)
detach
split again
-> 4;
end
split again
-> 5;
stop
end split
@enduml
```





6.12 Заметки

На диаграмму активностей (также, как и на остальные диаграммы) можно добавлять заметки.

Текст внутри заметок можно дополнительного форматировать используя creole и HTML синтаксис.

По умолчанию создаются заметки, которые явно указывают на элемент, который был объявлен непосредственно перед объявлением заметки. Чтобы создать отсоединенную заметку, нужно использовать ключевое слово `floating` (она по прежнему будет находиться рядом с элементом, но у неё не будет "хвостика" указателя).

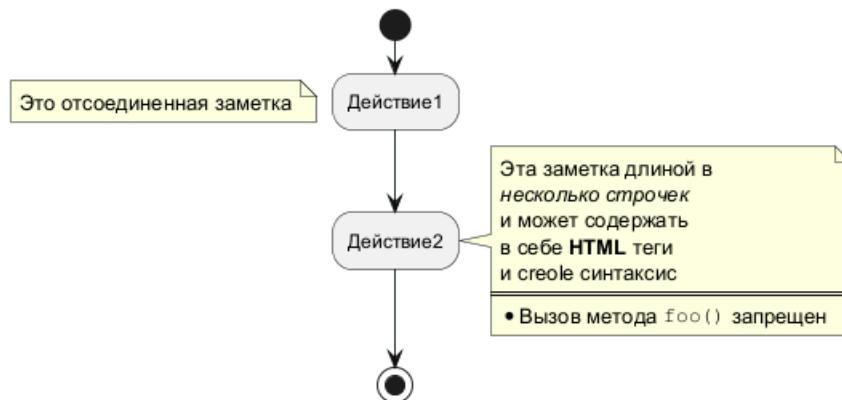
```
@startuml
```

```
start
: 1;
floating note left:
: 2;
note right
```

```
//      //
```

```
<b>HTML</b>
creole
=====
*      ""foo()"""
end note
stop
```

```
@enduml
```



Можно добавлять заметки к обратным действиям в циклах:

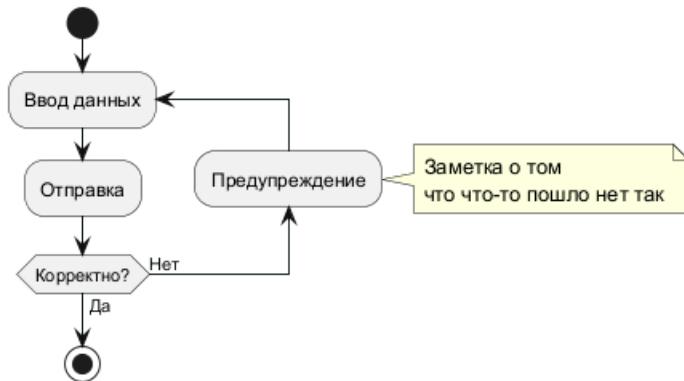
```
@startuml
start
repeat :      ;
:      ;
backward :     ;
```



```

    note right:      \n  -
repeat while (      ?) is ( ) not ( )
stop
@enduml

```



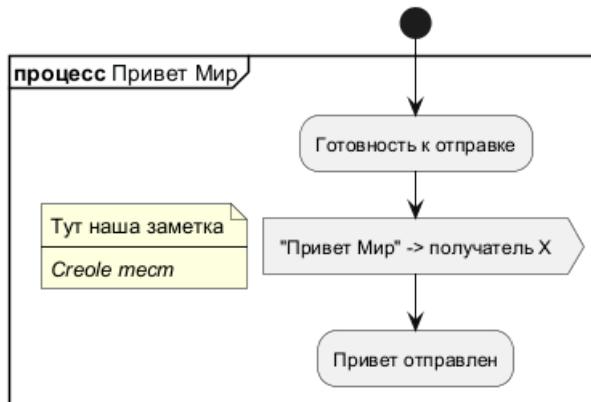
[Ref. QA-11788]

Можно добавлять заметки к разделам:

```

@startuml
start
partition "***" {
    note
        -----
        //Creole  //
    end note
    :
    :"      " ->      >
    :
}
@enduml

```



[Ref. QA-2398]

6.13 Цвета

Можно задавать цвета для активностей

```
@startuml
```

```
:      ;
```

```
#HotPink:
```

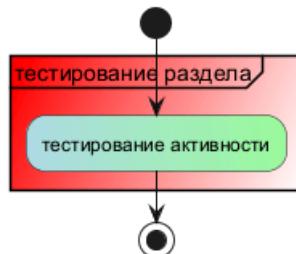


```
----  
! ;  
  
#AAAAAA : ;  
  
@enduml
```



Также можно задавать заливку градиентом:

```
@startuml  
start  
  
partition #red/white " " {  
    #lightblue\palegreen :  
}  
  
stop  
@enduml
```



[Ref. QA-4906]

6.14 Линии вместо стрелок

Можно превратить все стрелки на диаграмме в линии, если указать команду `skinparam ArrowHeadColor none` в начале диаграммы.

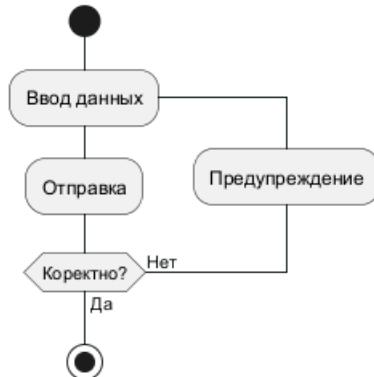
```
@startuml  
Skinparam ArrowHeadColor none  
start  
: !;  
: -  
** **;  
stop  
@enduml
```





```

@startuml
skinparam ArrowHeadColor none
start
repeat :      ;
:      ;
backward :      ;
repeat while (      ?) is ( ) not ( )
stop
@enduml
  
```



6.15 Изменение цвета и стиля стрелок (синтаксис квадратных скобочек)

Явно объявляя стрелку и указывая для неё дополнительные параметры, можно управлять отображением стрелок между элементами диаграммы.

6.15.1 Стиль линий

Можно задавать различные стили отображения линий при помощи следующей конструкции:

- -[#]-->

где [#] может принимать значения:

- bold - жирный,
- dashed - штриховой,
- dotted - пунктирный,
- hidden - скрытый,
- plain - непрерывный.

6.15.2 Заметки на линиях

Можно добавлять текстовые подписи к линиям, используя следующую конструкцию:

- -->



6.15.3 Цвет линий

Можно задавать цвет линий при помощи следующей конструкции:

- -[#] ->

где # может принимать текстовые обозначения цветов или их числовое представление.

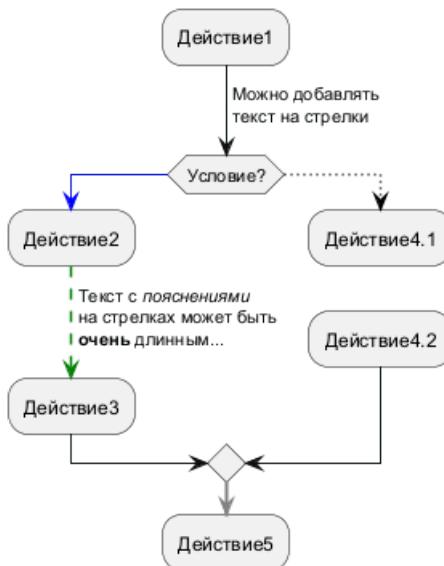
6.15.4 Полный синтаксис

Можно смешивать все указанные выше параметры, следующим образом:

- -[# ,]->

6.15.5 Комплексный пример

```
@startuml
: 1;
-> \n ;
if ( ?) then
-[#blue]->
: 2;
-[#green,dashed]-> // //
** ** ... ;
: 3;
else
-[#black,dotted]->
: 4.1;
-[hidden]->
: 4.2;
endif
-[#gray,bold]->
: 5;
@enduml
```



6.16 Коннектор

Можно добавлять на диаграмму специальный элемент - коннектор. Для этого необходимо использовать круглые скобки. Чтобы убрать связь между двумя коннекторами, нужно использовать ключевое слово `detach`.



```
@startuml
start
:
;
(A)
detach
(A)
:
;
@enduml
```

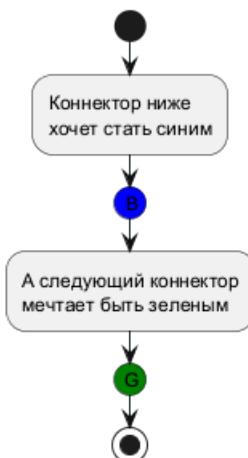


6.17 Цвет коннектора

Можно указать цвет для коннектора.

```
@startuml
start
:
;
:#blue:(B)
:
;
:#green:(G)
stop
@enduml
```





[Ref. QA-10077]

6.18 Объединение (группировка) активностей

Можно объединять (группировать) активности на диаграмме используя:

- group - группы;
- partition - разделы;
- package - пакеты;
- rectangle - прямоугольники;
- card - карточки.

отличаются эти варианты только внешним видом (а для group еще немного другим синтаксисом записи и отсутствием возможности задавать цвет).

6.18.1 Группы

Итак, можно объединять активности используя группы:

```

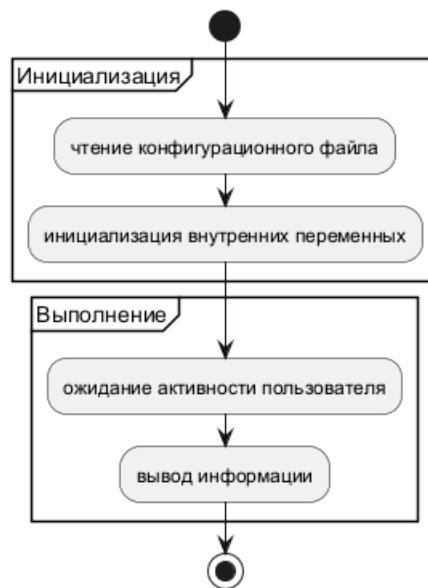
group
  ...
end group
  ...
@startuml
start

group
  :
  ;
  :
end group
group
  :
  ;
  :
end group

stop
@enduml

```





6.18.2 Разделы, Пакеты, Прямоугольники и Карточки

Также, можно объединять активности используя Разделы, Пакеты, Прямоугольники и Карточки. Синтаксис у всех этих вариантов одинаковый, отличие состоит только в форме отображения.

```

partition/package/rectangle/card [# ] { ... }

}

@startuml
start

group : ;
end group
floating note:

partition {
: ;
}
floating note:

package {
: ;
}
floating note:

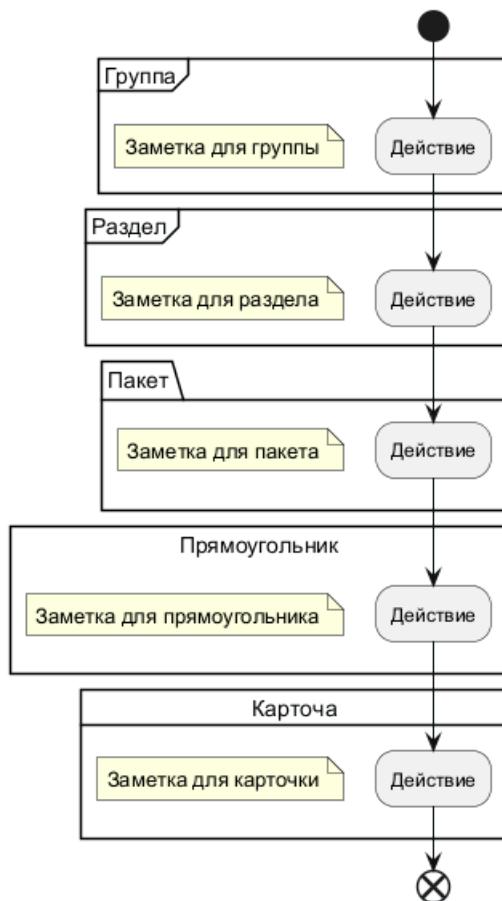
rectangle {
: ;
}
floating note:

card {
: ;
}
floating note:

end
@enduml

```





Для Разделов, Пакетов, Прямоугольников и Карточек можно задавать цвет отображения (для Групп этого сделать нельзя):

```

@startuml
start

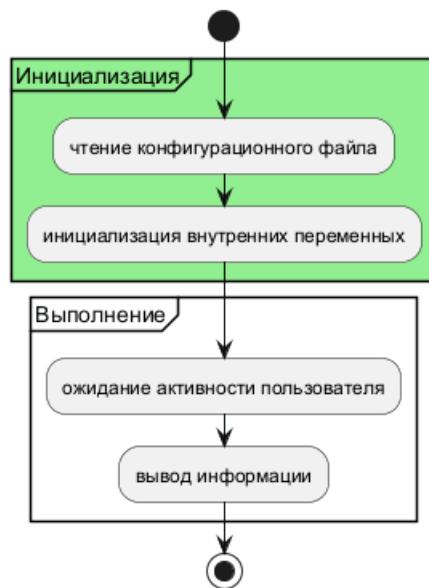
partition #lightGreen {
    :
    ;
}

partition {
    :
    ;
}

stop
@enduml

```





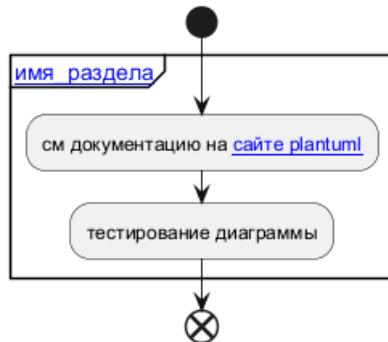
[Ref. QA-2793]

Также, в названиях можно использовать гиперссылки:

```

@startuml
start
partition "[[http://plantuml.com _ ]]" {
    : [[http://plantuml.com plantuml]];
    ;
}
end
@enduml

```



[Ref. QA-542]

6.19 Дорожки

Можно задать дорожки используя символ |. Также для задаваемых дорожек можно задавать цвета.

```

@startuml

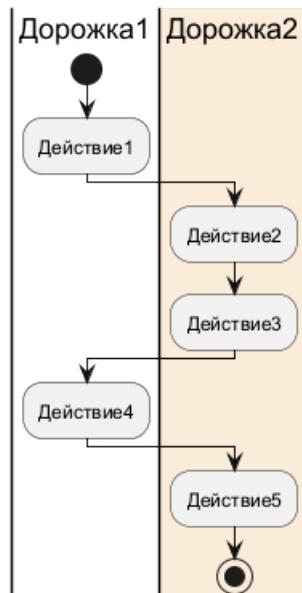
| 1|
start
: 1;

|#AntiqueWhite| 2|
: 2;
: 3;

```



```
|    1|
:    4;
|    2|
:    5;
stop
@enduml
```



Можно использовать дорожки вместе с Условиями и Циклами.

```
@startuml
```

```
|#pink| _ _ |
start
if ( ?) is ( ) then
#pink :**          **;
:    1;
else (      )

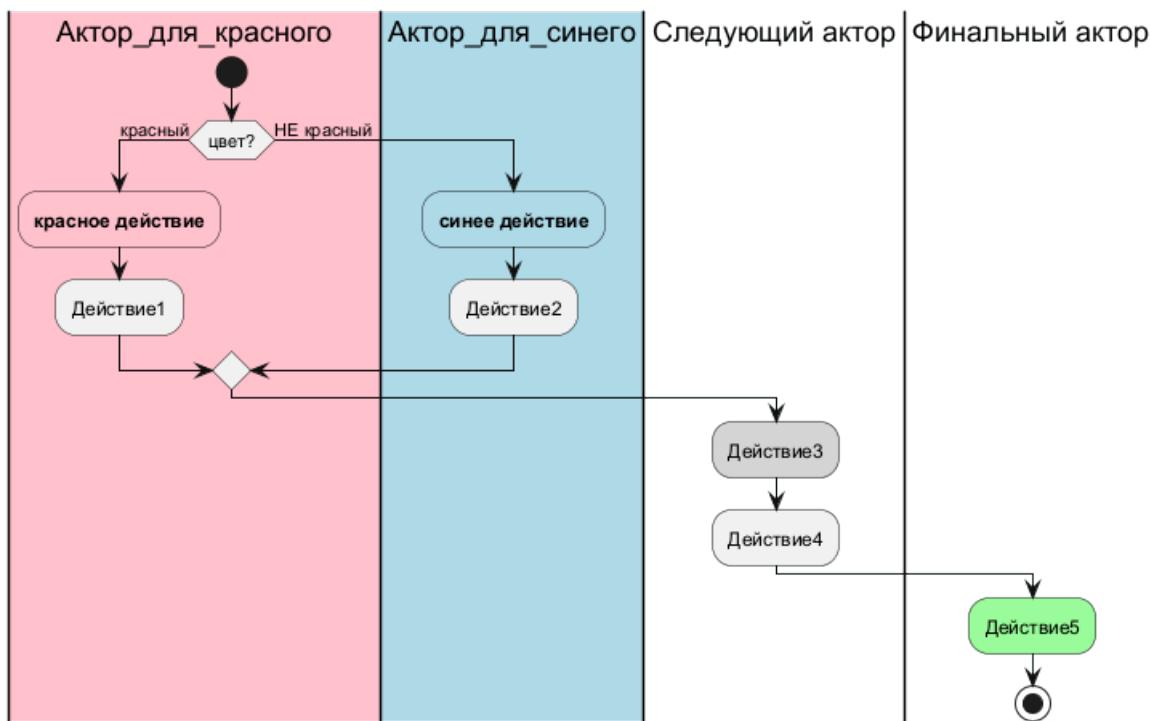
|#lightblue| _ _ |
#lightblue :**          **;
:    2;
endif

|      |
#lightgray :    3;
:    4;

|      |
#palegreen :    5;
stop
```

```
@enduml
```





Также, для того, чтобы каждый раз не писать длинное имя Дорожки, можно задать для дорожки короткое имя и далее использовать везде его. Сделать это можно используя ключевое слово `alias` при первом определении Дорожки:

- | [#<color>|] < > | < >

`@startuml`

| #palegreen | |

| |

| #gold | |

| |

`start`

: ;

| |

: ;

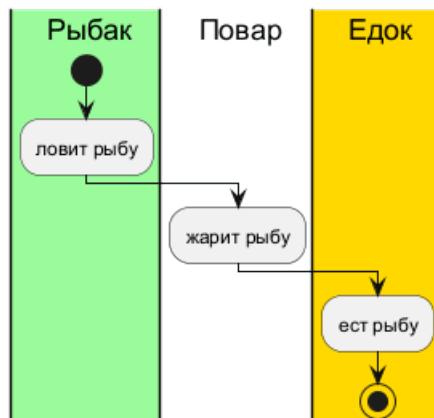
| |

: ;

`stop`

`@enduml`





[Ref. QA-2681]

6.20 Прерывание активности

Можно прервать активность (удалить стрелку) используя идентичные по своему действию ключевые слова `detach` или `kill`:

- `detach`

```
@startuml

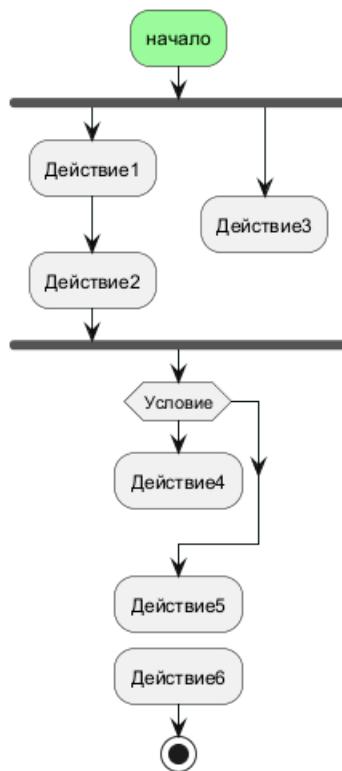
#palegreen :     ;
fork
:     1;
:     2;

fork again
:     3;
detach
endfork

if (     ) then
:     4;
detach
endif

:     5;
detach
:     6;
stop
@enduml
```





- kill

@startuml

```

#palegreen :      ;
fork
:      1;
:      2;

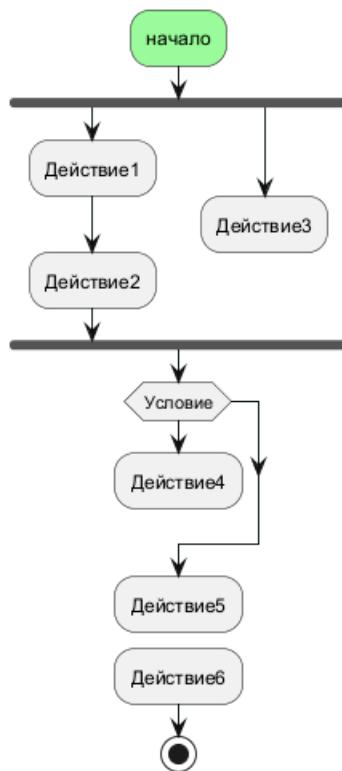
fork again
:      3;
kill
endfork

if (      ) then
:      4;
detach
endif

:      5;
detach
:      6;
stop
@enduml

```





6.21 SDL (язык спецификаций с формальной семантикой)

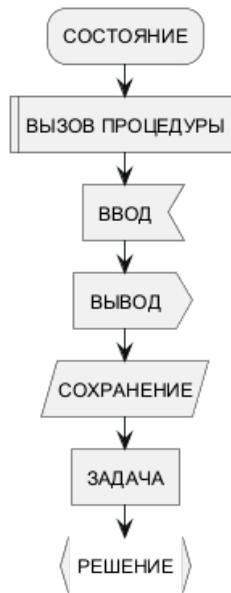
(еще на русский переводится как "язык спецификаций и описаний")

Заменяя символ ;, обозначающий конец описания действия, на символы из списка ниже, можно изменять внешний вид действия. Ниже перечислены доступные варианты с указанием обозначаемых ими типов:

- ; - STATE,
- | - CALL,
- < - INPUT,
- > - OUTPUT,
- / - SAVE,
-] - TASK,
- } - DECISION,

```
@startuml
:      ;
:      |
:  <
:  >
:  /
:  ]
:  }
@enduml
```





Еще один пример:

```

@startuml
:   ;
:   _ (o)|
:   ;

split
:   (i)<
:   (o)>

split again
:   (i)<
:   _ (o)
|_
:i := i + 1]
:   (o)>

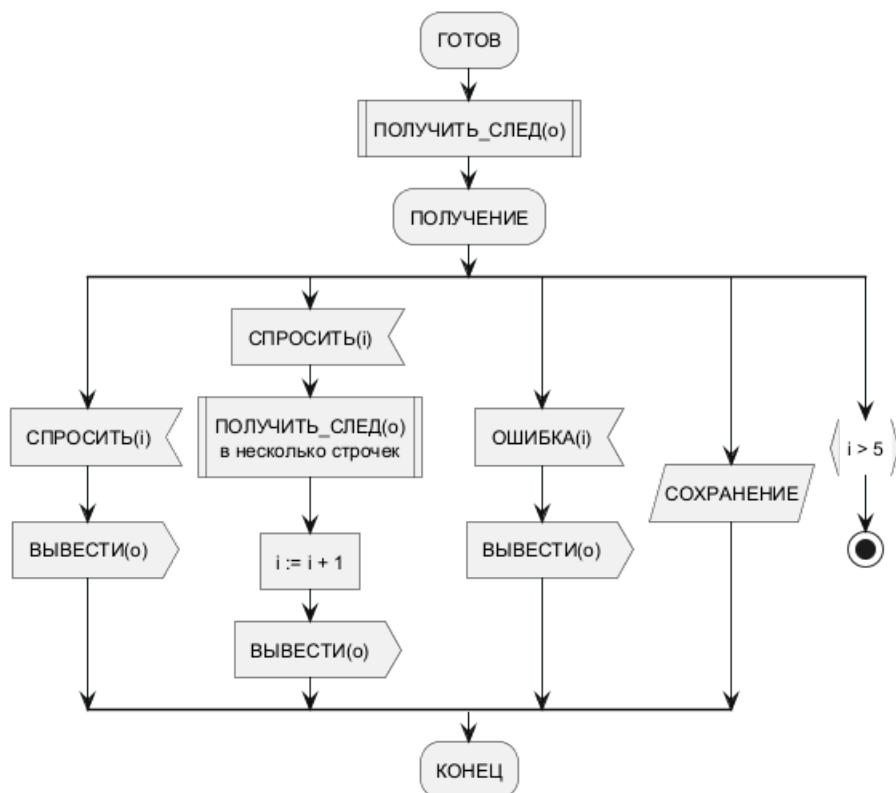
split again
:   (i)<
:   (o)>

split again
:   /
|_
split again
:i > 5}
stop

end split

:   ;
@enduml
  
```





6.22 Полноценный пример

@startuml

```

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

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

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



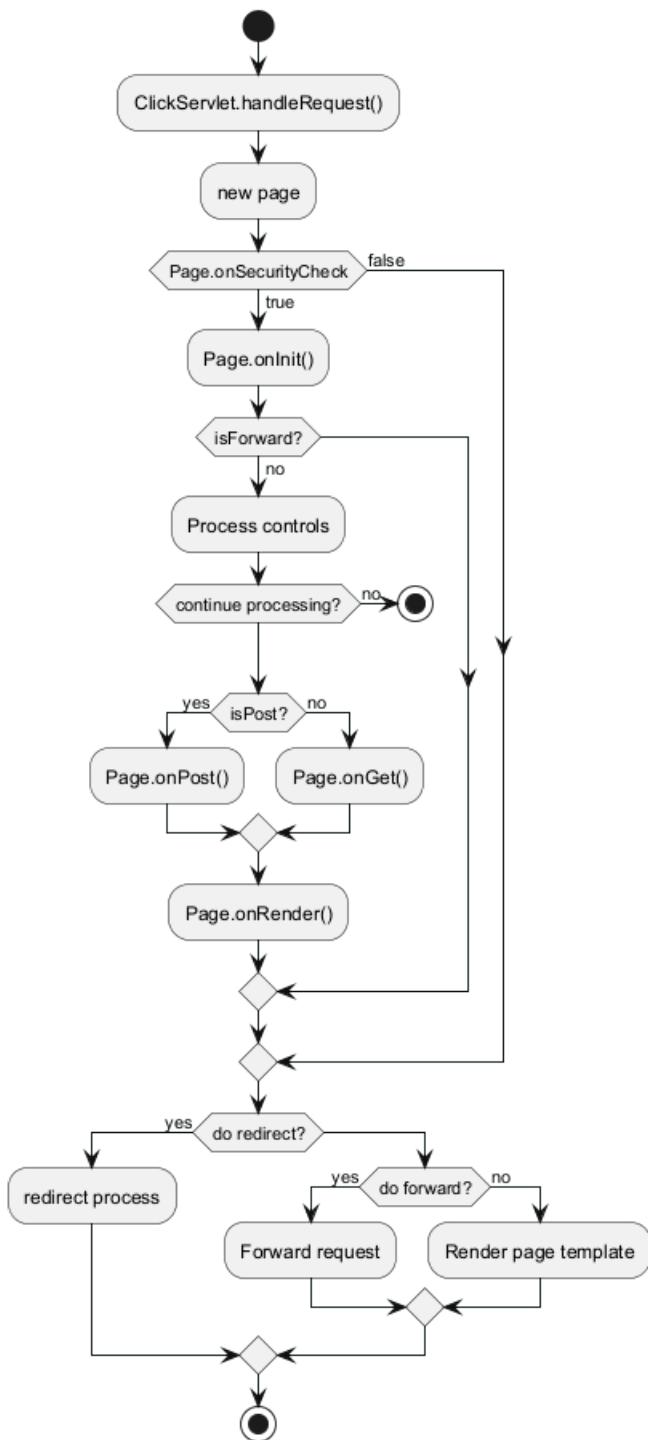
```

endif
endif

stop

@enduml

```



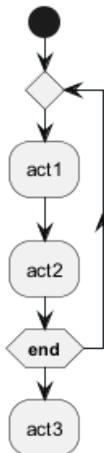
6.23 Condition Style

6.23.1 Inside style (by default)

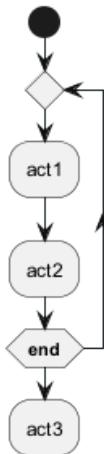
```
@startuml
```



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



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

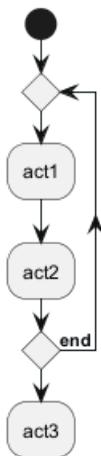


6.23.2 Diamond style

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

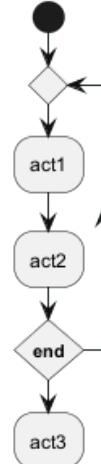


```
@enduml
```



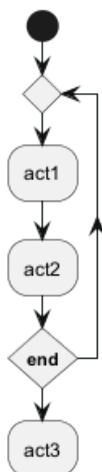
6.23.3 InsideDiamond (or *Foo1*) style

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



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





[Ref. QA-1290 and #400]

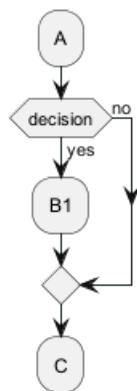
6.24 Condition End Style

6.24.1 Diamond style (by default)

- With one branch

```

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



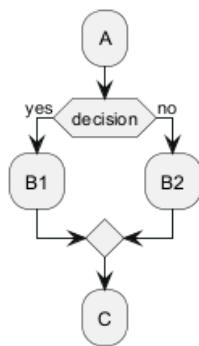
- With two branches (B1, B2)

```

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



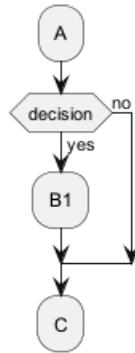
@enduml

**6.24.2 Horizontal line (hline) style**

- With one branch

```

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

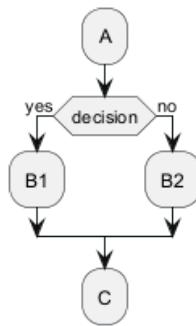


- With two branches (B1, B2)

```

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





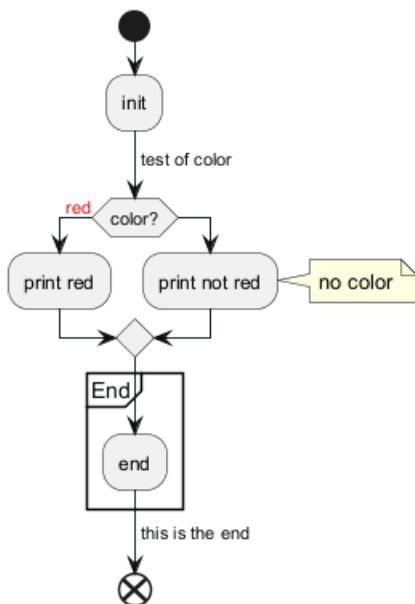
[Ref. QA-4015]

6.25 Using (global) style

6.25.1 Without style (by default)

```

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



6.25.2 With style

You can use style to change rendering of elements.

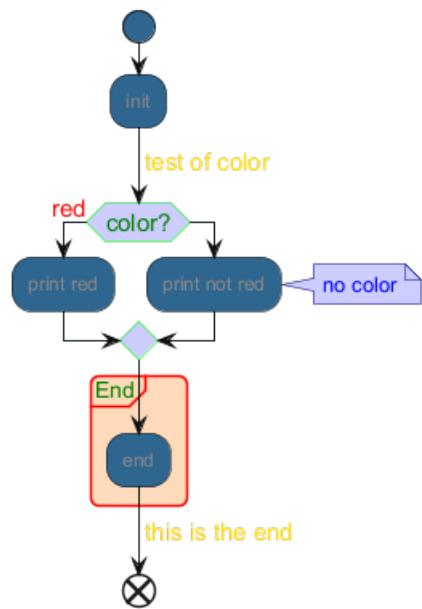
```
@startuml
```



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

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





7 Диаграмма компонентов

Диаграмма компонентов: Диаграмма компонентов - это тип структурной диаграммы, используемой в языке UML (Unified Modeling Language) для визуализации организаций и взаимосвязей компонентов системы. Эти диаграммы помогают разбить сложную систему на управляемые компоненты, показать их взаимозависимость и обеспечить эффективное проектирование и архитектуру системы.

Преимущества PlantUML:

- **Простота:** С помощью PlantUML можно создавать диаграммы компонентов, используя простые и интуитивно понятные текстовые описания, что избавляет от необходимости использовать сложные средства рисования.
- **Интеграция:** PlantUML легко интегрируется с различными инструментами и платформами, что делает его универсальным выбором для разработчиков и архитекторов.
- **Совместная работа:** Форум PlantUML предоставляет пользователям возможность обсуждать, делиться и обращаться за помощью к своим диаграммам, создавая тем самым сообщество для совместной работы.

7.1 Компоненты

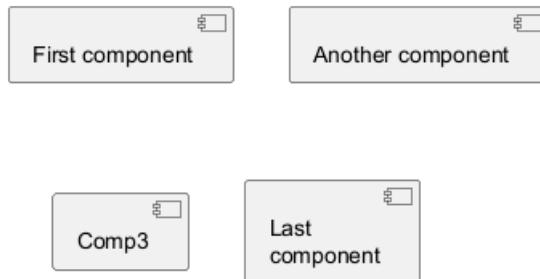
Обозначения компонентов должны быть заключены в квадратные скобки.

Также можно использовать ключевое слово `component` для объявления компонента. Вы можете объявить алиас с помощью ключевого слова `as`. Этот алиас может быть использован позже, при объявлении связей.

```
@startuml
```

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

```
@enduml
```



7.2 Интерфейсы

Для обозначения интерфейса используется символ () (потому что он выглядит как круг).

Также возможно использование ключевого слова `interface` для объявления интерфейса. Вы можете объявить алиас с помощью ключевого слова `as`. Этот алиас может быть использован позднее, когда будут задаваться связи.

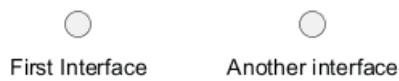
Далее мы увидим, что задание интерфейсов опционально.

```
@startuml
```

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



@enduml



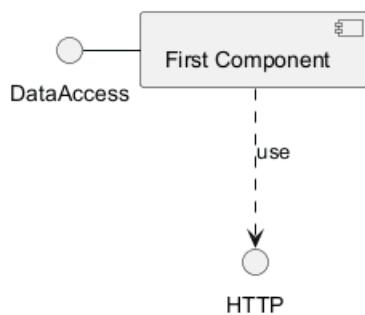
7.3 Простой пример

Отношения между элементами создаются с помощью комбинации точечных линий (..), прямых линий (--) и стрелок (-->).

@startuml

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

@enduml



7.4 Использование заметок

Вы можете использовать ключевые слова `note left of`, `note right of`, `note top of`, `note bottom of` чтобы задать метки, относящиеся к одному объекту.

Заметка также может быть задана не прикреплённой, используя ключевое слово `note`, а затем прикреплена к другим объектам, используя символ ...

@startuml

```
interface "Data Access" as DA

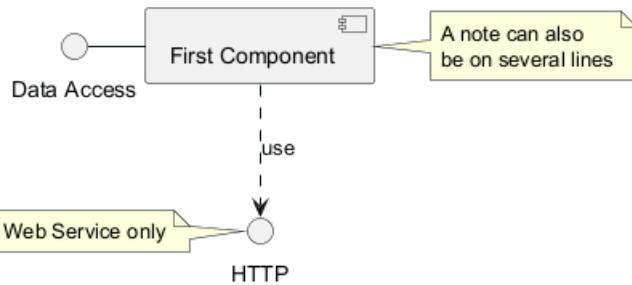
DA - [First Component]
[First Component] ..> HTTP : use

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 Группирование компонентов

Вы можете использовать несколько ключевых слов package, чтобы группировать компоненты и интерфейсы вместе.

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

@startuml

```

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

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

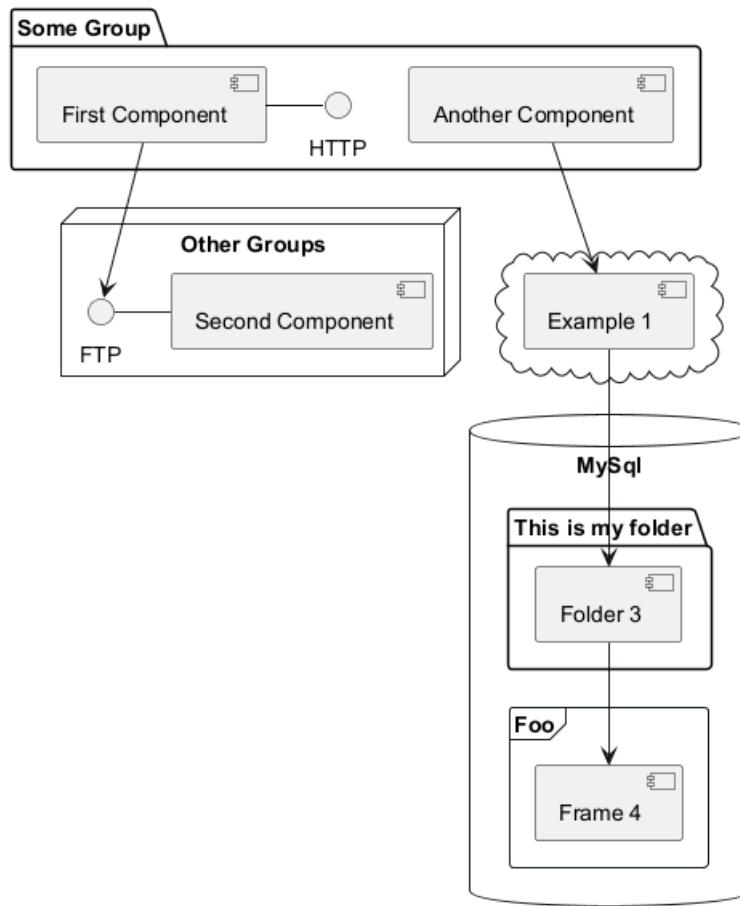
cloud {
    [Example 1]
}

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
  
```

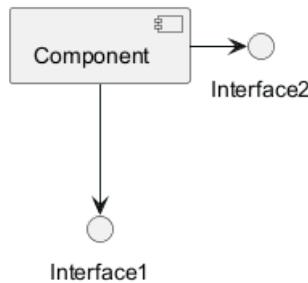




7.6 Изменение направления стрелок

По умолчанию, связи между классами имею два тире -- и ориентированы вертикально. Можно создавать горизонтальные связи с помощью одного тире (или точки), вот так:

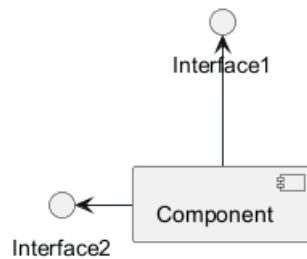
```
@startuml
[Component] --> Interface1
[Component] -> Interface2
@enduml
```



Вы также можете изменять направления, перевернув связь:

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

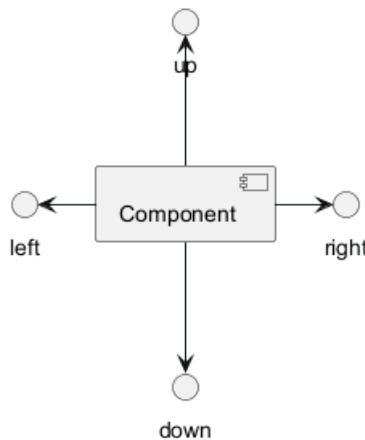




Также, можно изменить направление стрелки добавлением ключевых слов `left`, `right`, `up` или `down` внутри стрелки:

```

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



Вы можете сократить запись, используя только первую букву направления (например, `-d-` вместо `-down-`) или две первые буквы (`-do-`).

Пожалуйста, заметьте, что не стоит использовать эту функциональность без особой надобности: *Graphviz* обычно даёт хорошие результаты без дополнительной настройки.

[See also 'Change diagram orientation' on \[Deployment diagram\]\(deployment-diagram\) page.](#)

7.7 Use UML2 notation

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

```

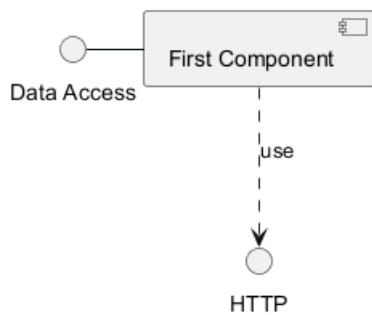
@startuml

interface "Data Access" as DA

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

@enduml
  
```





7.8 Использование нотации UML1

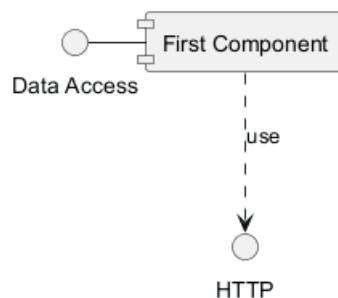
Команда `skinparam componentStyle uml1` используется, чтобы переключиться на нотацию UML1.

```
@startuml
skinparam componentStyle uml1
```

```
interface "Data Access" as DA
```

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

```
@enduml
```



7.9 Use rectangle notation (remove UML notation)

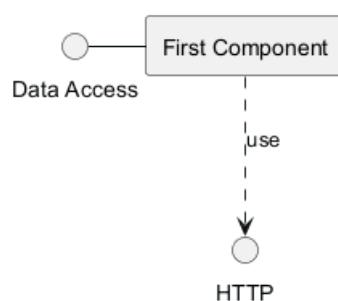
The `skinparam componentStyle rectangle` command is used to switch to rectangle notation (*without any UML notation*).

```
@startuml
skinparam componentStyle rectangle
```

```
interface "Data Access" as DA
```

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

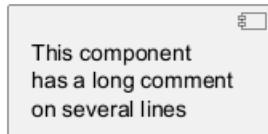
```
@enduml
```



7.10 Длинное описание

Для помещения многострочного текста в тело компонента используются квадратные скобки

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



7.11 Индивидуальные цвета

Вы можете задать цвет после определения компонента.

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

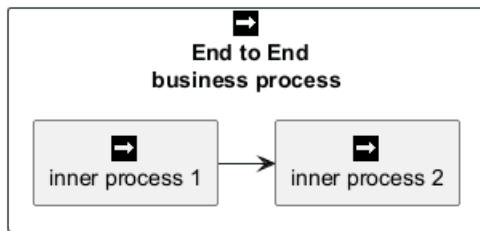


7.12 Использование Sprite в стереотипах

Можно использовать спрайты внутри компонентов стереотипа.

```
@startuml
sprite $businessProcess [16x16/16] {
FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFOFFFFF
FFFFFFFFFFFFFO0FFFF
FF000000000000FF
FF000000000000FF
FF000000000000FF
FFFFFFFFF00FFFF
FFFFFFFFF0FFFF
FFFFFFFFF0FFFF
FFFFFFFFF0FFFF
FFFFFFFFF0FFFF
FFFFFFFFF0FFFF
}
rectangle " End to End\nbusiness process" <<$businessProcess>> {
rectangle "inner process 1" <<$businessProcess>> as src
rectangle "inner process 2" <<$businessProcess>> as tgt
src -> tgt
}
@enduml
```





7.13 Skinparam

Вы можете использовать команду skinparam для изменения шрифтов и цветов диаграммы

Вы можете использовать данную команду :

- В определении диаграммы, как любую другую команду,
- В подключенном файле,
- В конфигурационном файле, указанном в командной строке в задании ANT.

Вы можете задать цвет и шрифт для компонентов и интерфейсов с заданными шаблонами.

@startuml

```

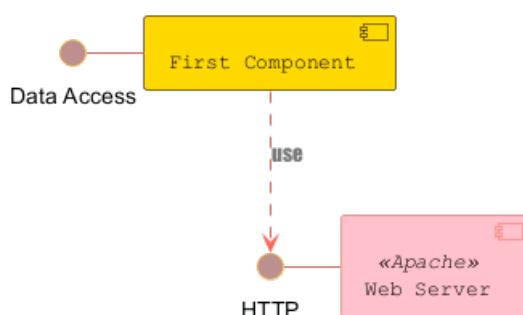
skinparam interface {
    backgroundColor RosyBrown
    borderColor orange
}

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

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

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

@enduml



@startuml

```

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

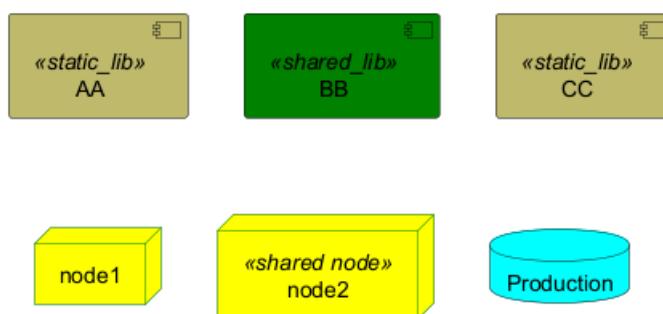
skinparam node {
borderColor Green
backgroundColor Yellow
backgroundColor<<shared_node>> Magenta
}
skinparam databaseBackgroundColor Aqua

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

node node1
node node2 <<shared_node>>
database Production

```

@enduml



7.14 Specific SkinParameter

7.14.1 componentStyle

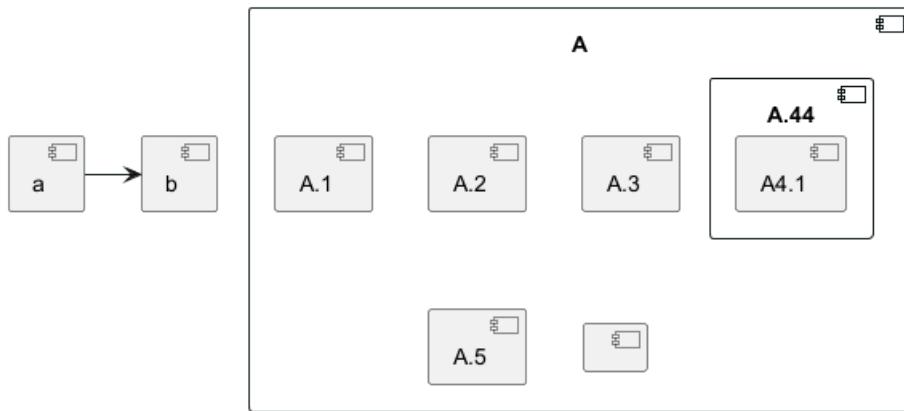
- By default (or with `skinparam componentStyle uml2`), you have an icon for component

```

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

```



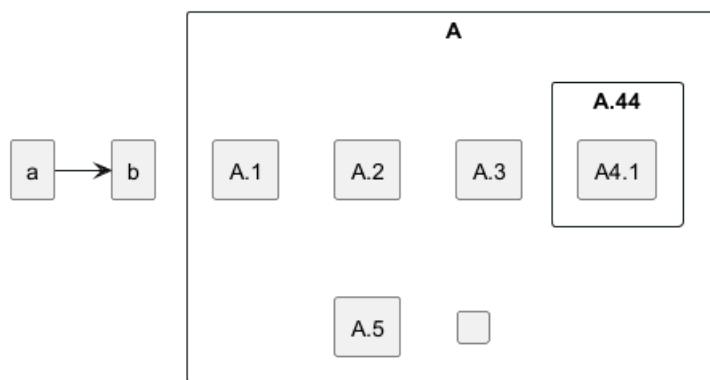


- If you want to suppress it, and to have only the rectangle, you can use `skinparam componentStyle rectangle`

```

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

```



[Ref. 10798]

7.15 Hide or Remove unlinked component

By default, all components are displayed:

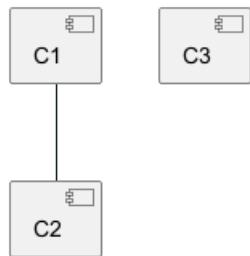
```

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

```



```
@enduml
```

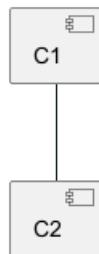


But you can:

- hide @unlinked components:

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

```
hide @unlinked
@enduml
```



- or remove @unlinked components:

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

```
remove @unlinked
@enduml
```



[Ref. QA-11052]

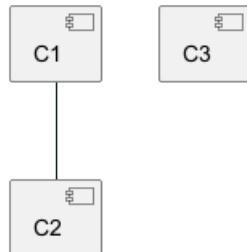
7.16 Hide, Remove or Restore tagged component or wildcard

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

By default, all components are displayed:



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



But you can:

- hide \$tag13 components:

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

hide $tag13
@enduml
```



- or remove \$tag13 components:

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

remove $tag13
@enduml
```



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

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

remove $tag13
```



```
restore $tag1
@enduml
```



- or remove * and restore \$tag1 components:

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

remove *
restore $tag1
@enduml
```



[Ref. QA-7337 and QA-11052]

7.17 Display JSON Data on Component diagram

7.17.1 Simple example

```
@startuml
allowmixing

component Component
()           Interface

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



JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]



For another example, see on JSON page.

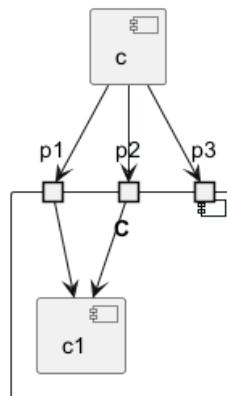
7.18 Port [port, portIn, portOut]

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

7.18.1 Port

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

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

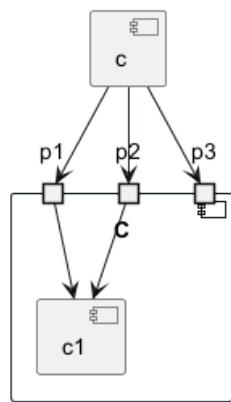


7.18.2 PortIn

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

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

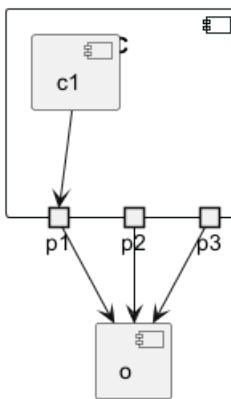




7.18.3 PortOut

```

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



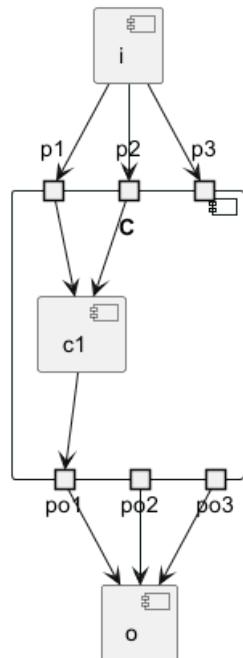
7.18.4 Mixing PortIn & PortOut

```

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



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



8 Диаграмма развертывания

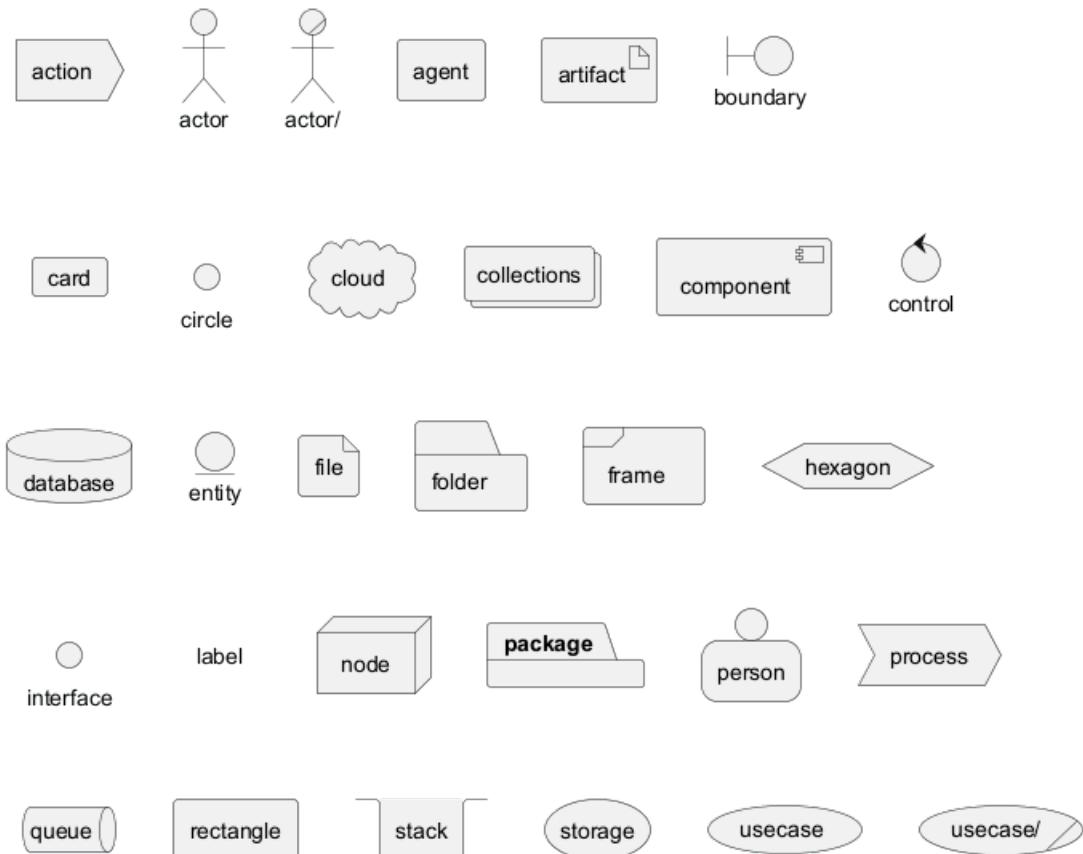
Диаграмма развертывания - это тип диаграммы, визуализирующий архитектуру систем, показывающий, как программные компоненты развертываются на аппаратном обеспечении. Она дает четкое представление о распределении компонентов по различным узлам, таким как серверы, рабочие станции и устройства.

С помощью PlantUML создание диаграмм развертывания становится простым делом. Платформа предлагает простой и интуитивно понятный способ создания таких диаграмм с использованием обычного текста, что обеспечивает быстроту итераций и удобство контроля версий. Кроме того, форум PlantUML представляет собой активное сообщество, где пользователи могут обращаться за помощью, делиться идеями и совместно решать проблемы, связанные с созданием диаграмм. Одним из ключевых преимуществ PlantUML является его способность легко интегрироваться с различными инструментами и платформами, что делает его предпочтительным выбором как для профессионалов, так и для энтузиастов.

8.1 Объявление элемента

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





Для длинного описания можно дополнительно поместить текст с помощью скобок [].

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

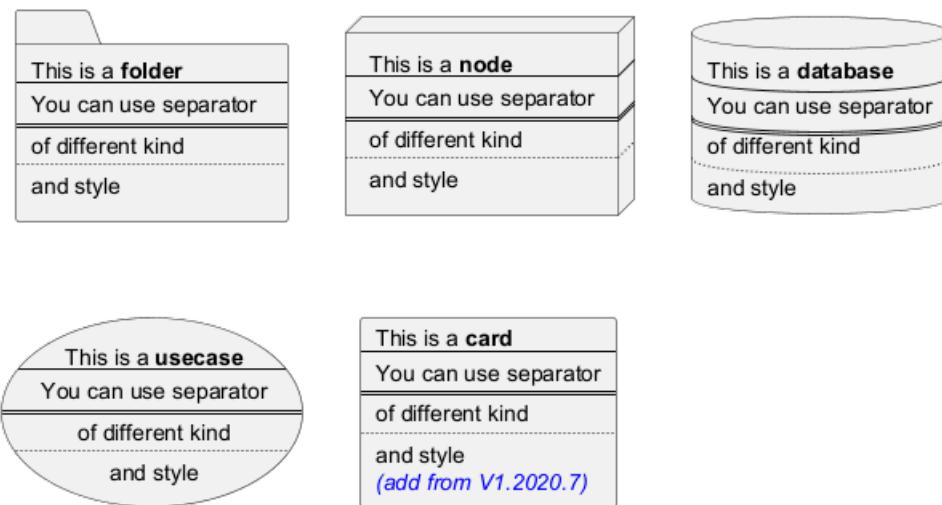
```
node node [
This is a <b>node
-----
You can use separator
=====
of different kind
....
and style
]
```

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



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

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



8.2 Declaring element (using short form)

We can declare element using some short forms.

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

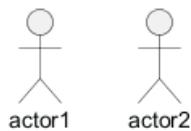
8.2.1 Actor

```
@startuml
```

```
actor actor1
:actor2:
```

```
@enduml
```





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

8.2.2 Component

```
@startuml
```

```
component component1
[component2]
```

```
@enduml
```



8.2.3 Interface

```
@startuml
```

```
interface interface1
() "interface2"

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



interface example

8.2.4 Usecase

```
@startuml
```

```
usecase usecase1
(usecase2)

@enduml
```



8.3 Linking or arrow

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

```
@startuml
```

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

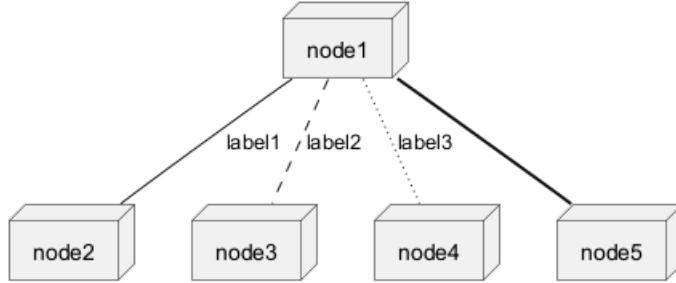


```

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

@enduml

```



It is possible to use several types of links:

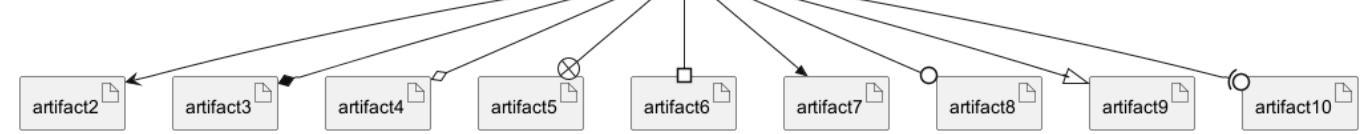
```

@startuml

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

@enduml

```



You can also have the following types:

```
@startuml
```

```

cloud cloud1
cloud cloud2
cloud cloud3
cloud cloud4

```

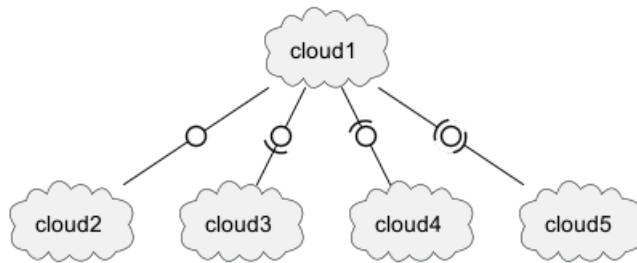


```

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

```

@enduml



or another example:

```

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

```

```

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

```

```
actor1 -0)- actor2
```

```

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

```

```

boundary b1
control c1
b1 -(0)- c1

```

```

component comp1
interface interf1
comp1 #~~( interf1

```

```

:mode1actor: -0)- fooa1
:mode1actor1: -ri0)- foo11

```

```

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

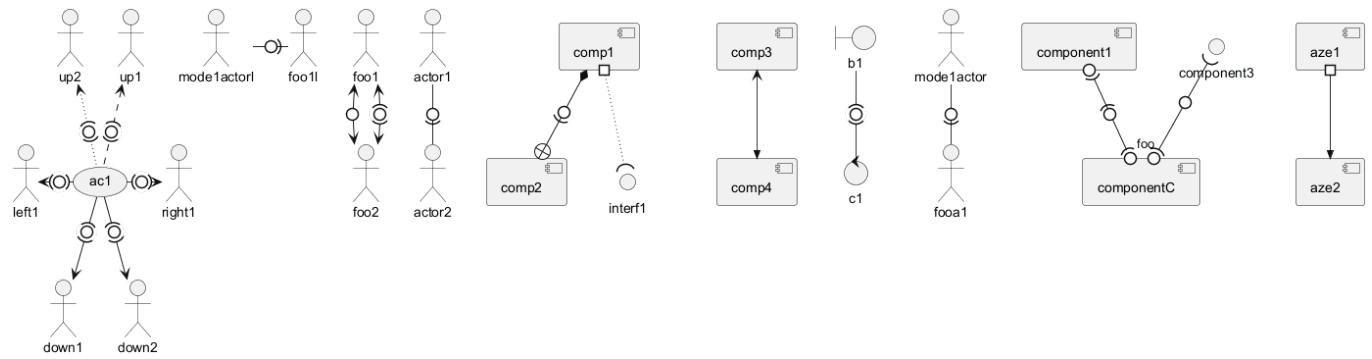
```

```

[aze1] #-->> [aze2]
@enduml

```





[Ref. QA-547 and QA-1736]

□ See all type on [Appendix](#).

8.4 Bracketed arrow style

Similar as Bracketed class relations (linking or arrow) style

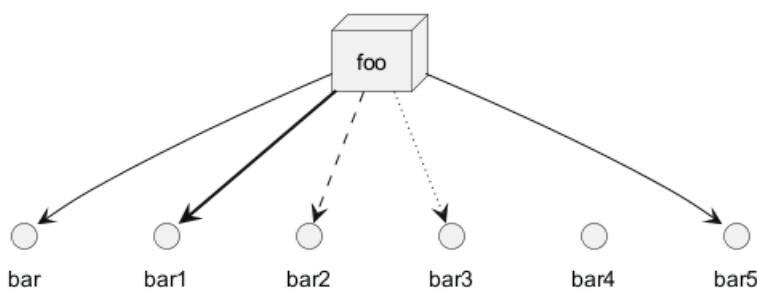
8.4.1 Line style

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

- without label

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

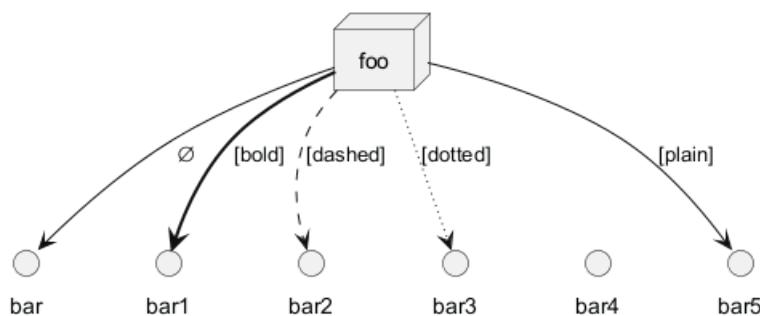
Bracketed line style without label



- with label

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



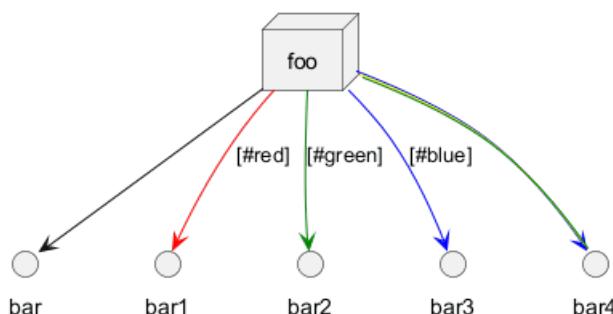
Bracketed line style with label

[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
  
```

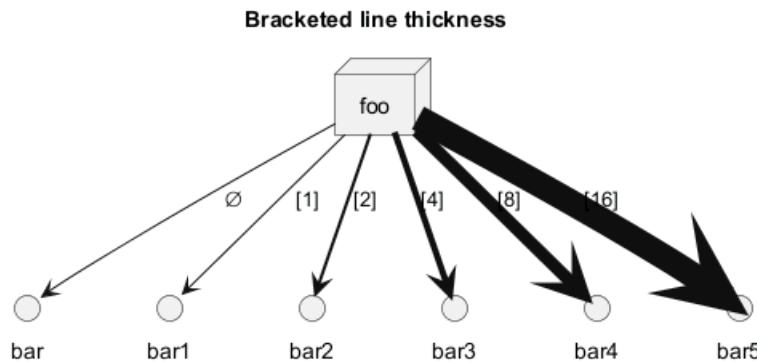
Bracketed line color

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
  
```



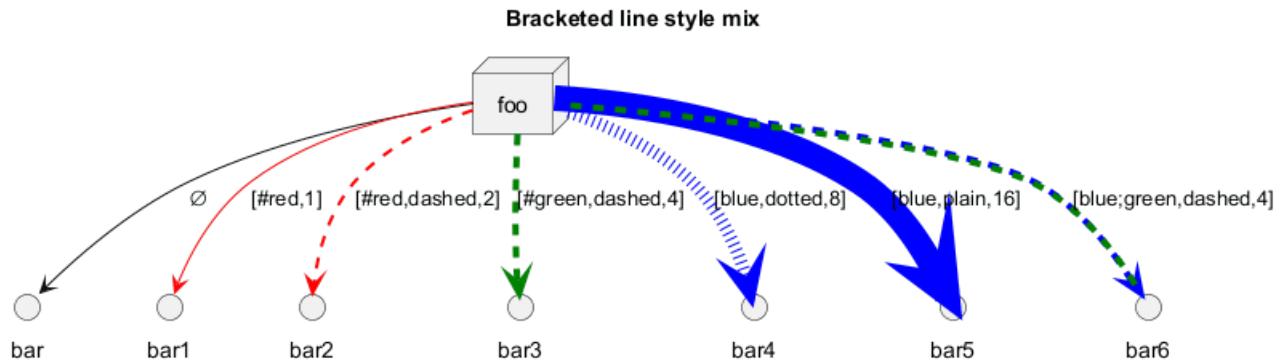


[Adapted from QA-4949]

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)

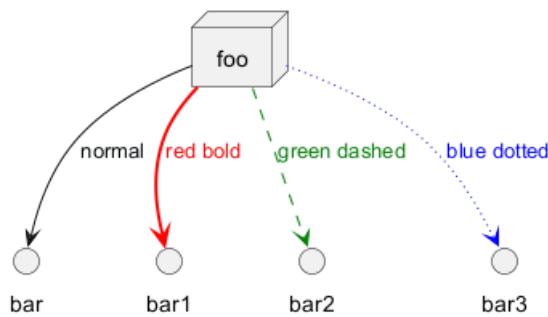
You can change the color or style of individual arrows using the inline following notation:

- #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)

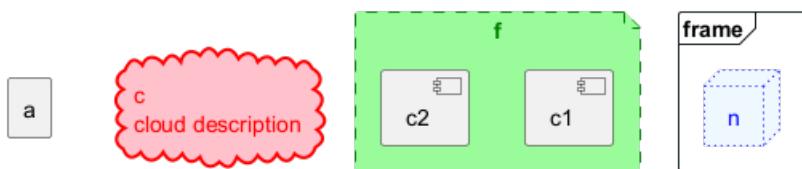
You can change the color or style of individual element using the following notation:

- # [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
action action {
}
artifact artifact {
}
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
}
node node {
}
package package {
}
process process {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml
```



8.8 Packages and nested elements

8.8.1 Example with one level

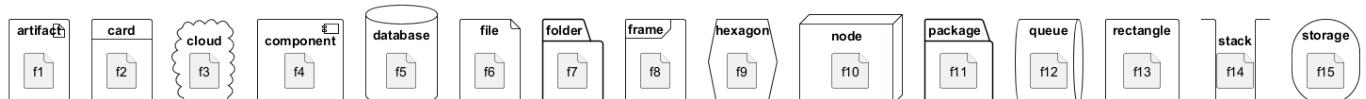
```
@startuml
artifact      artifactVeryL00000000000000000000000g      as "artifact" {
file f1
}
card         cardVeryL00000000000000000000000g      as "card" {
file f2
}
cloud        cloudVeryL00000000000000000000000g      as "cloud" {
file f3
}
component    componentVeryL00000000000000000000000g  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



```

file f4
}
database databaseVeryL00000000000000000000g as "database" {
file f5
}
file fileVeryL00000000000000000000g as "file" {
file f6
}
folder folderVeryL00000000000000000000g as "folder" {
file f7
}
frame frameVeryL00000000000000000000g as "frame" {
file f8
}
hexagon hexagonVeryL00000000000000000000g as "hexagon" {
file f9
}
node nodeVeryL00000000000000000000g as "node" {
file f10
}
package packageVeryL00000000000000000000g as "package" {
file f11
}
queue queueVeryL00000000000000000000g as "queue" {
file f12
}
rectangle rectangleVeryL00000000000000000000g as "rectangle" {
file f13
}
stack stackVeryL00000000000000000000g as "stack" {
file f14
}
storage storageVeryL00000000000000000000g 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 {

```

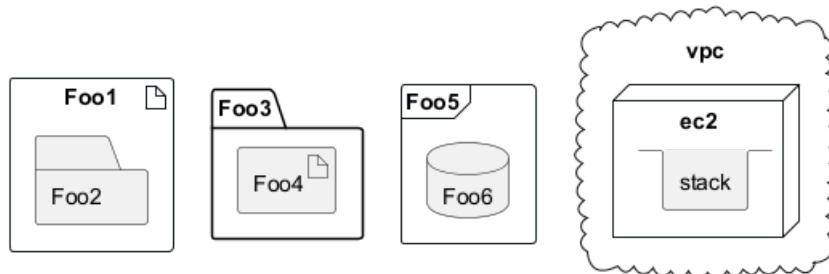


```

node ec2 {
    stack stack
}
}

@enduml

```



```

@startuml
node Foo1 {
    cloud Foo2
}

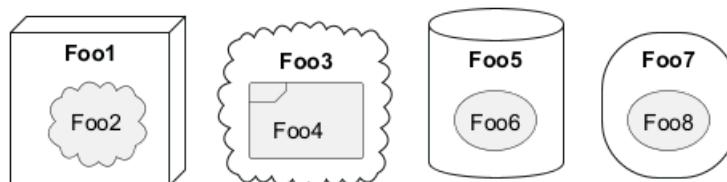
cloud Foo3 {
    frame Foo4
}

database Foo5 {
    storage Foo6
}

storage Foo7 {
    storage Foo8
}
}

@enduml

```



8.8.3 Full nesting

Here is all the nested elements:

- by alphabetical order:

```

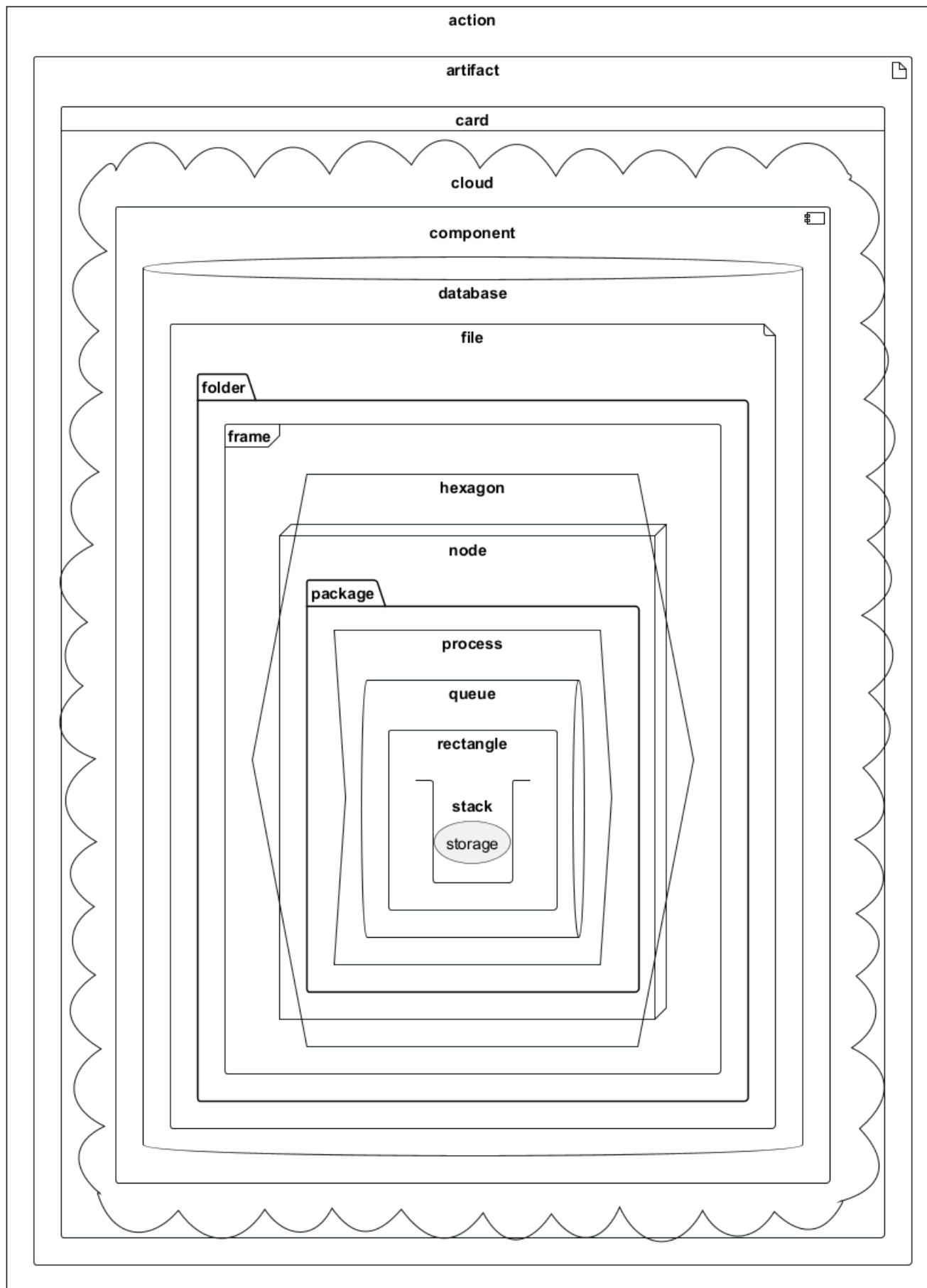
@startuml
action action {
artifact artifact {
card card {
cloud cloud {
component component {
database database {
file file {
folder folder {
frame frame {
hexagon hexagon {
node node {

```



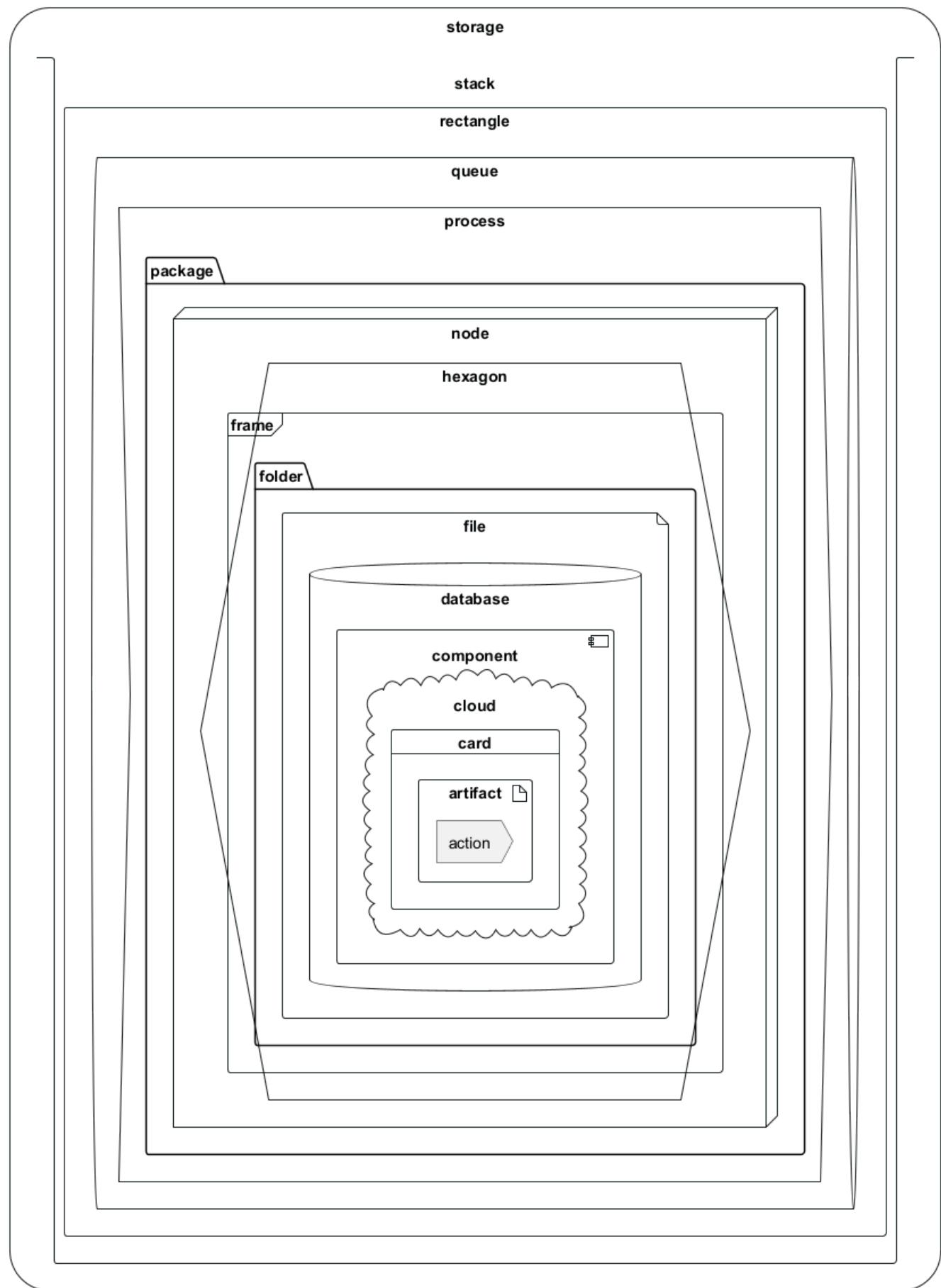
```
package package {  
process process {  
queue queue {  
rectangle rectangle {  
stack stack {  
storage storage {  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
@enduml
```





- or reverse alphabetical order



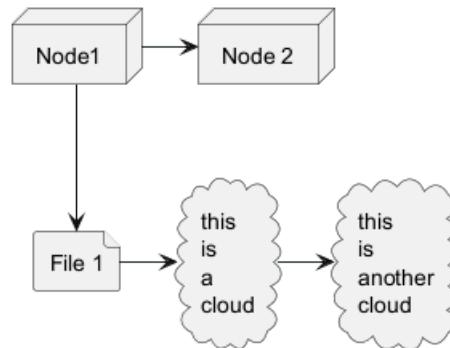


8.9 Alias

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 actorVeryLongXXXXXXXXXXXXXXXXXXXXg
agent      "agent"       as agentVeryLongXXXXXXXXXXXXXXXXXXXXg
artifact   "artifact"    as artifactVeryLongXXXXXXXXXXXXXXXXXXXXg
boundary   "boundary"   as boundaryVeryLongXXXXXXXXXXXXXXXXXXXXg
card       "card"        as cardVeryLongXXXXXXXXXXXXXXXXXXXXg
cloud      "cloud"       as cloudVeryLongXXXXXXXXXXXXXXXXXXXXg
collections "collections" as collectionsVeryLongXXXXXXXXXXXXXXXXXXXXg
component   "component"  as componentVeryLongXXXXXXXXXXXXXXXXXXXXg
control     "control"    as controlVeryLongXXXXXXXXXXXXXXXXXXXXg
database   "database"   as databaseVeryLongXXXXXXXXXXXXXXXXXXXXg
entity      "entity"     as entityVeryLongXXXXXXXXXXXXXXXXXXXXg
file        "file"        as fileVeryLongXXXXXXXXXXXXXXXXXXXXg
folder      "folder"      as folderVeryLongXXXXXXXXXXXXXXXXXXXXg
frame       "frame"       as frameVeryLongXXXXXXXXXXXXXXXXXXXXg
hexagon     "hexagon"    as hexagonVeryLongXXXXXXXXXXXXXXXXXXXXg
interface   "interface"  as interfaceVeryLongXXXXXXXXXXXXXXXXXXXXg
label       "label"       as labelVeryLongXXXXXXXXXXXXXXXXXXXXg
node        "node"        as nodeVeryLongXXXXXXXXXXXXXXXXXXXXg
package    "package"    as packageVeryLongXXXXXXXXXXXXXXXXXXXXg
person     "person"     as personVeryLongXXXXXXXXXXXXXXXXXXXXg
queue      "queue"      as queueVeryLongXXXXXXXXXXXXXXXXXXXXg
```



```

stack      "stack"      as stackVeryL00000000000000000000000g
rectangle  "rectangle"  as rectangleVeryL00000000000000000000000g
storage    "storage"   as storageVeryL00000000000000000000000g
usecase    "usecase"   as usecaseVeryL00000000000000000000000g
@enduml

```



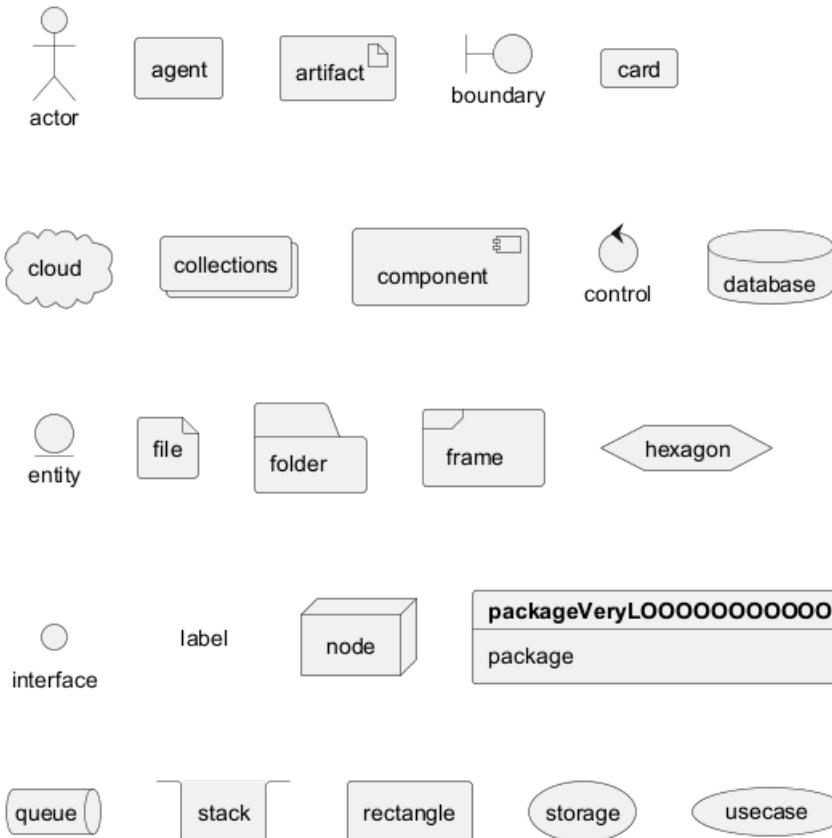
```

@startuml
actor      actorVeryL00000000000000000000000g      as "actor"
agent      agentVeryL00000000000000000000000g      as "agent"
artifact   artifactVeryL00000000000000000000000g     as "artifact"
boundary   boundaryVeryL00000000000000000000000g    as "boundary"
card       cardVeryL00000000000000000000000g       as "card"
cloud      cloudVeryL00000000000000000000000g      as "cloud"
collections collectionsVeryL00000000000000000000000g as "collections"
component  componentVeryL00000000000000000000000g   as "component"
control    controlVeryL00000000000000000000000g    as "control"
database   databaseVeryL00000000000000000000000g   as "database"
entity     entityVeryL00000000000000000000000g     as "entity"
file       fileVeryL00000000000000000000000g      as "file"
folder     folderVeryL00000000000000000000000g     as "folder"
frame      frameVeryL00000000000000000000000g      as "frame"
hexagon    hexagonVeryL00000000000000000000000g     as "hexagon"
interface  interfaceVeryL00000000000000000000000g   as "interface"
label      labelVeryL00000000000000000000000g      as "label"
node       nodeVeryL00000000000000000000000g      as "node"
package    packageVeryL00000000000000000000000g     as "package"
person    personVeryL00000000000000000000000g      as "person"
queue     queueVeryL00000000000000000000000g      as "queue"
stack     stackVeryL00000000000000000000000g      as "stack"
rectangle rectangleVeryL00000000000000000000000g   as "rectangle"
storage   storageVeryL00000000000000000000000g     as "storage"

```



```
usecase      usecaseVeryL000000000000000000g      as "usecase"
@enduml
```

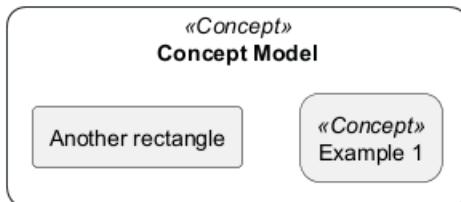


[Ref. QA-12082]

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

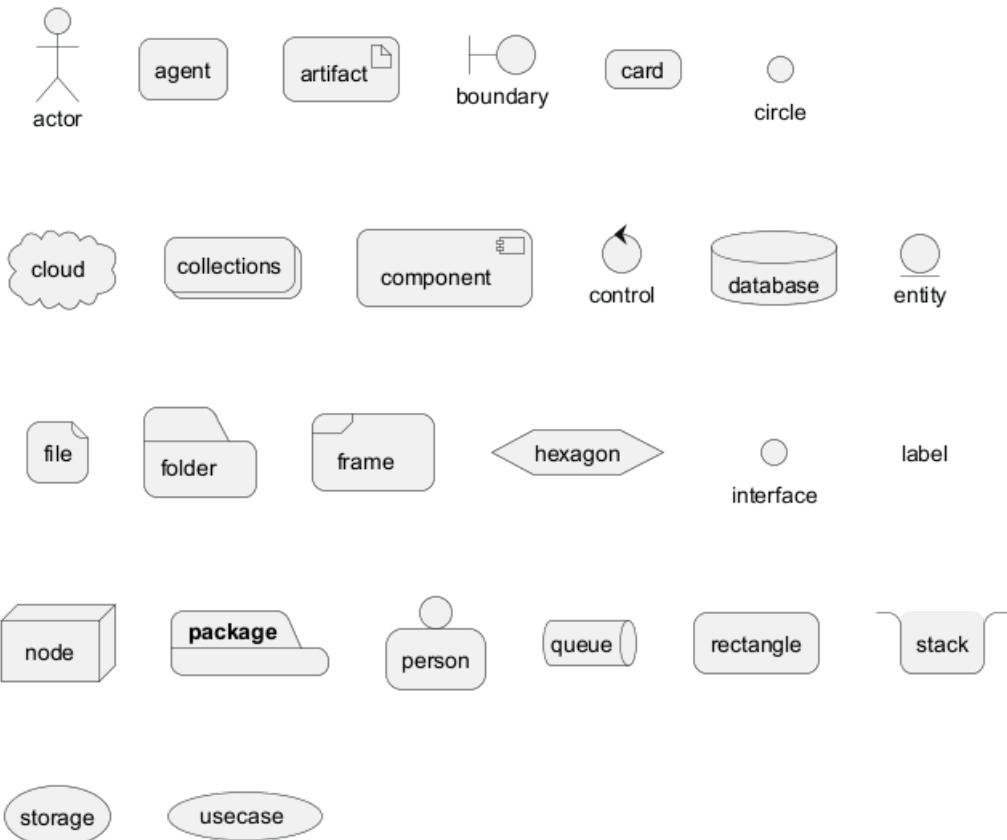
```
@startuml
skinparam roundCorner 15
actor actor
```



```

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
person person
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
@enduml

```



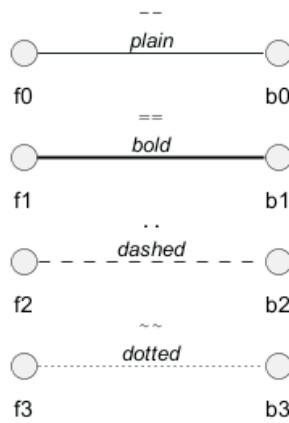
[Ref. QA-5299, QA-6915, QA-11943]



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
```



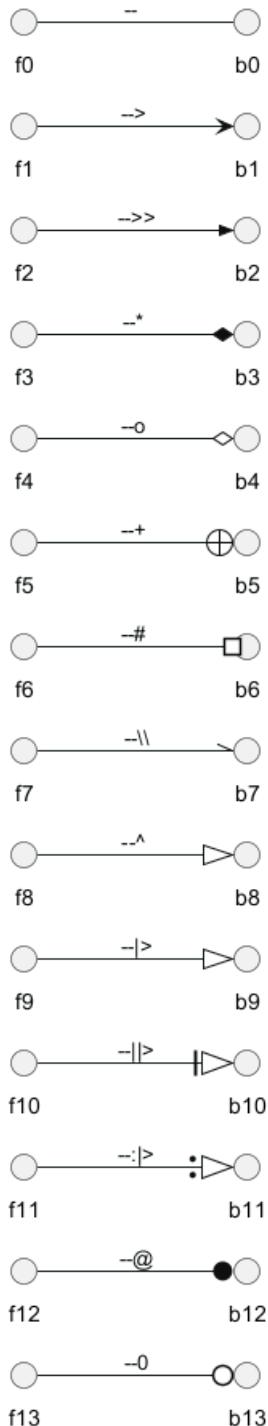
8.13 Appendix: All type of arrow head or '0' arrow

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

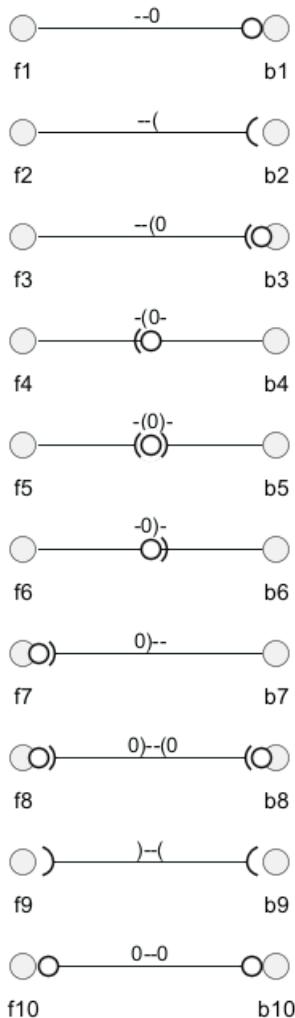
f10 0--0 b10 : "" 0--0 """
f9 )--( b9 : "" )--( """
f8 0)--(0 b8 : "" 0)--(0"""
f7 0)-- b7 : "" 0)-- """
f6 -0)- b6 : "" -0)- """
f5 -(0)- b5 : "" -(0)-"""

```



```
f4 -(0- b4 : "" -(0- ""
f3 --(0 b3 : "" --(0 ""
f2 --( b2 : "" --(  ""
f1 --0 b1 : "" --0  ""

@enduml
```



8.14 Appendix: Test of inline style on all element

8.14.1 Simple element

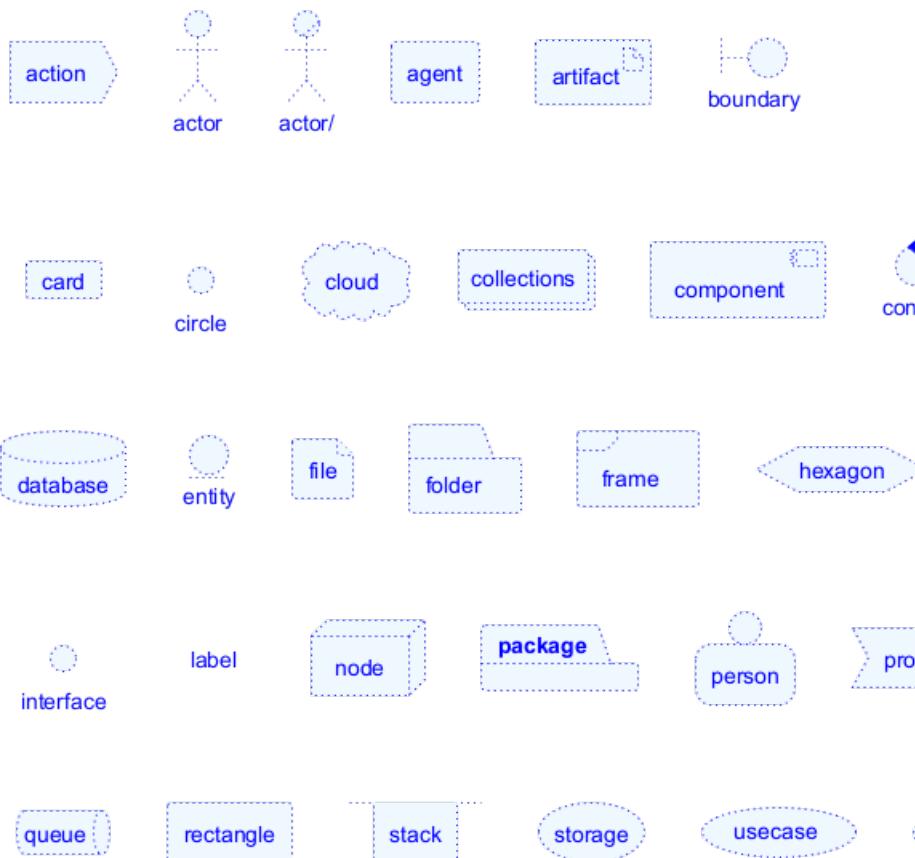
```
@startuml
action action           #aliceblue;line:blue;line.dotted;text:blue
actor actor            #aliceblue;line:blue;line.dotted;text:blue
actor/ "actor/"        #aliceblue;line:blue;line.dotted;text:blue
agent agent             #aliceblue;line:blue;line.dotted;text:blue
artifact artifact      #aliceblue;line:blue;line.dotted;text:blue
boundary boundary      #aliceblue;line:blue;line.dotted;text:blue
card card               #aliceblue;line:blue;line.dotted;text:blue
circle circle           #aliceblue;line:blue;line.dotted;text:blue
cloud cloud             #aliceblue;line:blue;line.dotted;text:blue
collections collections #aliceblue;line:blue;line.dotted;text:blue
component component    #aliceblue;line:blue;line.dotted;text:blue
control control         #aliceblue;line:blue;line.dotted;text:blue
database database       #aliceblue;line:blue;line.dotted;text:blue
entity entity           #aliceblue;line:blue;line.dotted;text:blue
file file               #aliceblue;line:blue;line.dotted;text:blue
```



```

folder folder          #aliceblue;line:blue;line.dotted;text:blue
frame frame           #aliceblue;line:blue;line.dotted;text:blue
hexagon hexagon       #aliceblue;line:blue;line.dotted;text:blue
interface interface   #aliceblue;line:blue;line.dotted;text:blue
label label            #aliceblue;line:blue;line.dotted;text:blue
node node              #aliceblue;line:blue;line.dotted;text:blue
package package        #aliceblue;line:blue;line.dotted;text:blue
person person          #aliceblue;line:blue;line.dotted;text:blue
process process        #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
action action #aliceblue;line:blue;line.dotted;text:blue {
}
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 {
}
process process #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
action      actionVeryL0000000000000000000g      as "action" #aliceblue;line:blue;line.dotted;text:blue
file f1
}
artifact    artifactVeryL0000000000000000000g     as "artifact" #aliceblue;line:blue;line.dotted;text:blue
file f1
}
card        cardVeryL0000000000000000000g       as "card" #aliceblue;line:blue;line.dotted;text:blue
file f2
}
cloud       cloudVeryL0000000000000000000g      as "cloud" #aliceblue;line:blue;line.dotted;text:blue
file f3
}
component   componentVeryL0000000000000000000g   as "component" #aliceblue;line:blue;line.dotted;text:blue
file f4
}
database   databaseVeryL0000000000000000000g    as "database" #aliceblue;line:blue;line.dotted;text:blue
file f5
}
file       fileVeryL0000000000000000000g       as "file" #aliceblue;line:blue;line.dotted;text:blue
file f6
}
folder     folderVeryL0000000000000000000g      as "folder" #aliceblue;line:blue;line.dotted;text:blue
file f7
}
frame      frameVeryL0000000000000000000g       as "frame" #aliceblue;line:blue;line.dotted;text:blue

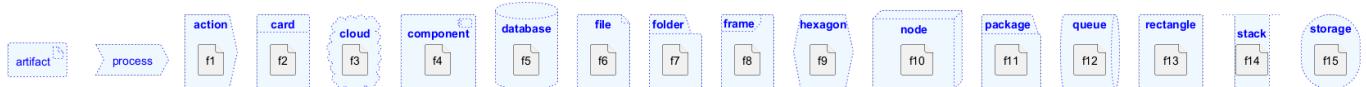
```



```

file f8
}
hexagon    hexagonVeryL00000000000000000000g      as "hexagon" #aliceblue;line:blue;line.dotted;text:blue
file f9
}
node       nodeVeryL00000000000000000000g      as "node" #aliceblue;line:blue;line.dotted;text:blue
file f10
}
package    packageVeryL00000000000000000000g     as "package" #aliceblue;line:blue;line.dotted;text:blue
file f11
}
process    processVeryL00000000000000000000g     as "process" #aliceblue;line:blue;line.dotted;text:blue
file f11
}
queue      queueVeryL00000000000000000000g      as "queue" #aliceblue;line:blue;line.dotted;text:blue
file f12
}
rectangle  rectangleVeryL00000000000000000000g    as "rectangle" #aliceblue;line:blue;line.dotted;text:blue
file f13
}
stack      stackVeryL00000000000000000000g      as "stack" #aliceblue;line:blue;line.dotted;text:blue
file f14
}
storage    storageVeryL00000000000000000000g     as "storage" #aliceblue;line:blue;line.dotted;text:blue
file f15
}
@enduml

```



8.15 Appendix: Test of style on all element

8.15.1 Simple element

8.15.2 Global style (on componentDiagram)

```

@startuml
<style>
componentDiagram {
    BackGroundColor palegreen
    LineThickness 1
    LineColor red
}
document {
    BackGroundColor white
}
</style>
actor actor
actor/ "actor/"
agent agent
artifact artifact
boundary boundary
card card
circle circle
cloud cloud
collections collections

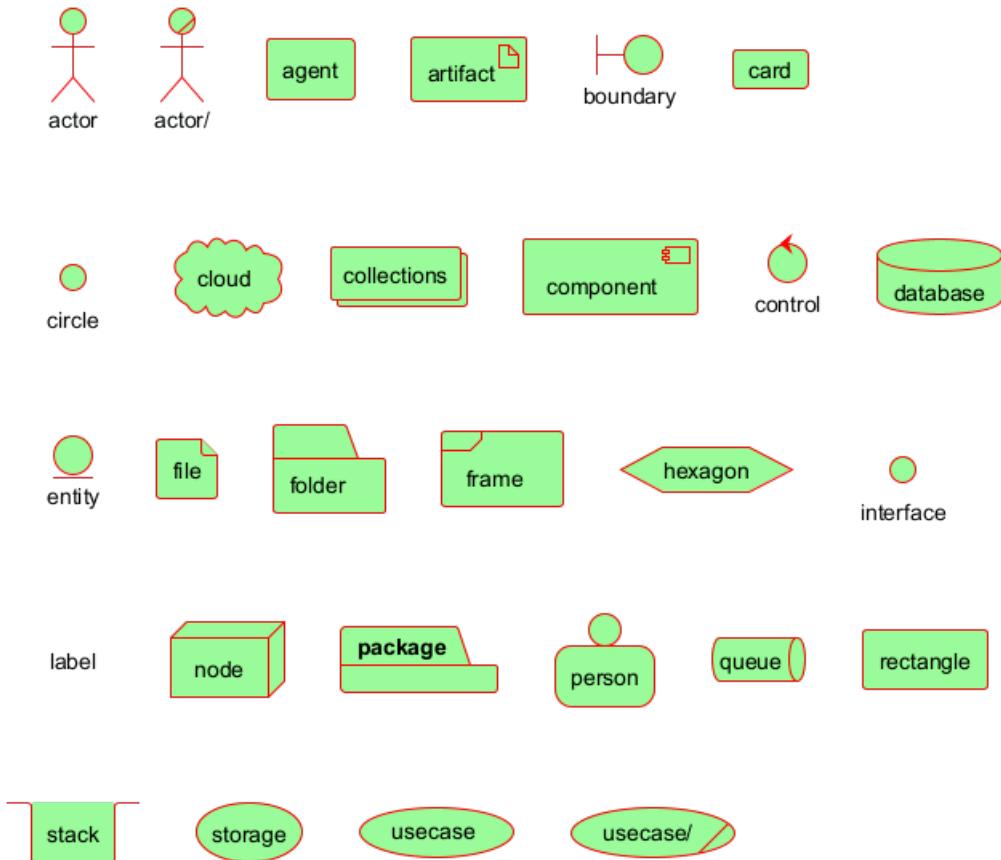
```



```

component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label
node node
package package
person person
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
usecase/ "usecase/"
@enduml

```



8.15.3 Style for each element

```

@startuml
<style>
actor {
    BackGroundColor #f80c12
    LineThickness 1
    LineColor black
}
agent {

```



```
BackGroundColor #f80c12
LineThickness 1
LineColor black
}
artifact {
    BackGroundColor #ee1100
    LineThickness 1
    LineColor black
}
boundary {
    BackGroundColor #ee1100
    LineThickness 1
    LineColor black
}
card {
    BackGroundColor #ff3311
    LineThickness 1
    LineColor black
}
circle {
    BackGroundColor #ff3311
    LineThickness 1
    LineColor black
}
cloud {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
collections {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
component {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
control {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
database {
    BackGroundColor #ff9933
    LineThickness 1
    LineColor black
}
entity {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
file {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
```



```
}

folder {
    BackGroundColor #ccbb33
    LineThickness 1
    LineColor black
}

frame {
    BackGroundColor #d0c310
    LineThickness 1
    LineColor black
}

hexagon {
    BackGroundColor #aacc22
    LineThickness 1
    LineColor black
}

interface {
    BackGroundColor #69d025
    LineThickness 1
    LineColor black
}

label {
    BackGroundColor black
    LineThickness 1
    LineColor black
}

node {
    BackGroundColor #22ccaa
    LineThickness 1
    LineColor black
}

package {
    BackGroundColor #12bdb9
    LineThickness 1
    LineColor black
}

person {
    BackGroundColor #11aabb
    LineThickness 1
    LineColor black
}

queue {
    BackGroundColor #11aabb
    LineThickness 1
    LineColor black
}

rectangle {
    BackGroundColor #4444dd
    LineThickness 1
    LineColor black
}

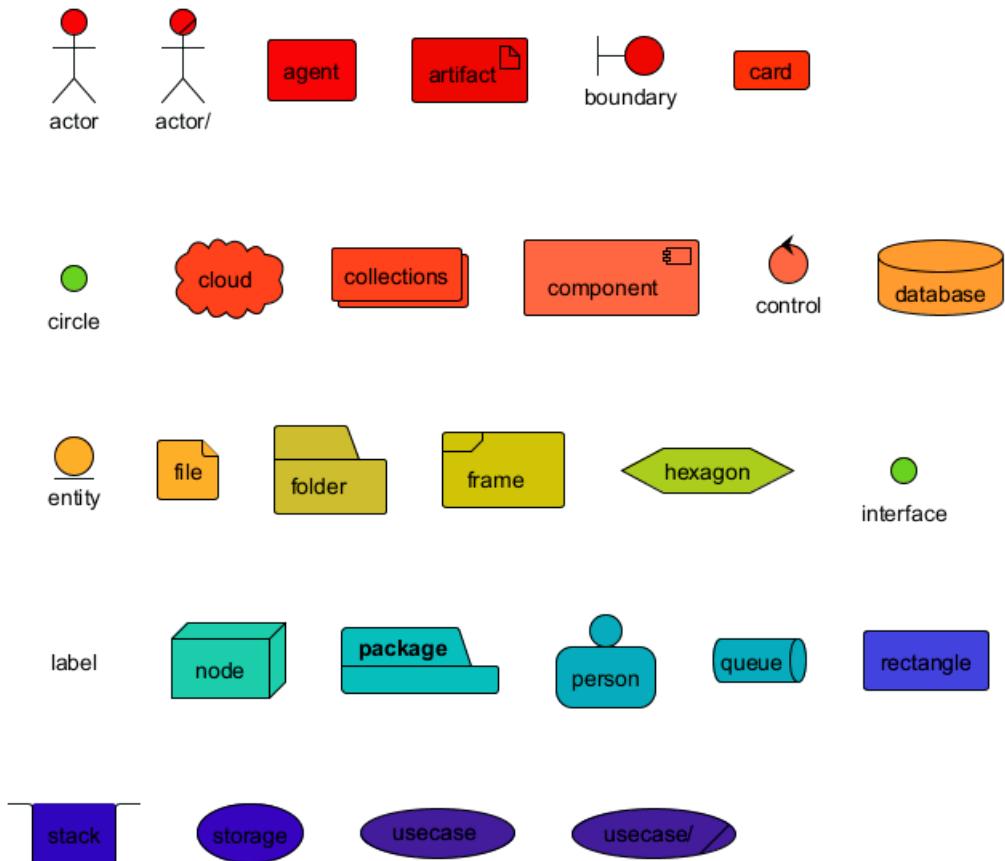
stack {
    BackGroundColor #3311bb
    LineThickness 1
    LineColor black
}

storage {
    BackGroundColor #3b0cbd
```



```
LineThickness 1
LineColor black
}
usecase {
    BackGroundColor #442299
    LineThickness 1
    LineColor black
}
</style>
actor actor
actor/ "actor/"
agent agent
artifact artifact
boundary boundary
card card
circle circle
cloud cloud
collections collections
component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label
node node
package package
person person
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
usecase/ "usecase/"
@enduml
```





[Ref. QA-13261]

8.15.4 Nested element (without level)

8.15.5 Global style (on componentDiagram)

```
@startuml
<style>
componentDiagram {
    BackGroundColor palegreen
    LineThickness 2
    LineColor red
}
</style>
artifact artifact {
}
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
```



```

}
node node {
}
package package {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml

```



8.15.6 Style for each nested element

```

@startuml
<style>
artifact {
    BackGroundColor #ee1100
    LineThickness 1
    LineColor black
}
card {
    BackGroundColor #ff3311
    LineThickness 1
    LineColor black
}
cloud {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
component {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
database {
    BackGroundColor #ff9933
    LineThickness 1
    LineColor black
}
file {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
folder {
    BackGroundColor #ccbb33
    LineThickness 1
    LineColor black
}
frame {

```



```
BackGroundColor #d0c310
LineThickness 1
LineColor black
}
hexagon {
    BackGroundColor #aacc22
    LineThickness 1
    LineColor black
}
node {
    BackGroundColor #22ccaa
    LineThickness 1
    LineColor black
}
package {
    BackGroundColor #12bdb9
    LineThickness 1
    LineColor black
}
queue {
    BackGroundColor #11aabb
    LineThickness 1
    LineColor black
}
rectangle {
    BackGroundColor #4444dd
    LineThickness 1
    LineColor black
}
stack {
    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
}
document {
    BackGroundColor white
}
</style>
artifact e1 as "artifact" {
file f1
}
card e2 as "card" {
file f2
}
cloud e3 as "cloud" {
file f3
}
component e4 as "component" {
file f4
}
database e5 as "database" {
file f5
}
file e6 as "file" {
file f6
}
folder e7 as "folder" {
file f7
}
frame e8 as "frame" {
file f8
}
hexagon e9 as "hexagon" {

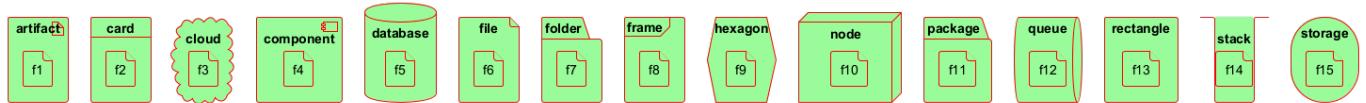
```



```

file f9
}
node e10 as "node" {
file f10
}
package e11 as "package" {
file f11
}
queue e12 as "queue" {
file f12
}
rectangle e13 as "rectangle" {
file f13
}
stack e14 as "stack" {
file f14
}
storage e15 as "storage" {
file f15
}
}
@enduml

```



8.15.9 Style for each nested element

```

@startuml
<style>
artifact {
    BackGroundColor #ee1100
    LineThickness 1
    LineColor black
}
card {
    BackGroundColor #ff3311
    LineThickness 1
    LineColor black
}
cloud {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
component {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
database {
    BackGroundColor #ff9933
    LineThickness 1
    LineColor black
}
file {
    BackGroundColor #feae2d
}

```



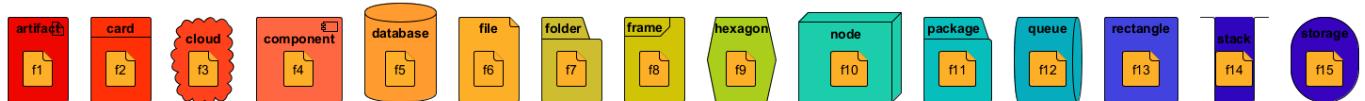
```
LineThickness 1
LineColor black
}
folder {
    BackGroundColor #ccbb33
    LineThickness 1
    LineColor black
}
frame {
    BackGroundColor #d0c310
    LineThickness 1
    LineColor black
}
hexagon {
    BackGroundColor #aacc22
    LineThickness 1
    LineColor black
}
node {
    BackGroundColor #22ccaa
    LineThickness 1
    LineColor black
}
package {
    BackGroundColor #12bdb9
    LineThickness 1
    LineColor black
}
queue {
    BackGroundColor #11aabb
    LineThickness 1
    LineColor black
}
rectangle {
    BackGroundColor #4444dd
    LineThickness 1
    LineColor black
}
stack {
    BackGroundColor #3311bb
    LineThickness 1
    LineColor black
}
storage {
    BackGroundColor #3b0cbd
    LineThickness 1
    LineColor black
}
}
</style>
artifact e1 as "artifact" {
file f1
}
card e2 as "card" {
file f2
}
cloud e3 as "cloud" {
file f3
}
```



```

component e4 as "component" {
file f4
}
database e5 as "database" {
file f5
}
file e6 as "file" {
file f6
}
folder e7 as "folder" {
file f7
}
frame e8 as "frame" {
file f8
}
hexagon e9 as "hexagon" {
file f9
}
node e10 as "node" {
file f10
}
package e11 as "package" {
file f11
}
queue e12 as "queue" {
file f12
}
rectangle e13 as "rectangle" {
file f13
}
stack e14 as "stack" {
file f14
}
storage e15 as "storage" {
file f15
}
}
@enduml

```



8.16 Appendix: Test of stereotype with style on all element

8.16.1 Simple element

```

@startuml
<style>
.stereo {
    BackgroundColor palegreen
}
</style>
actor actor << stereo >>
actor/ "actor/" << stereo >>
agent agent << stereo >>
artifact artifact << stereo >>
boundary boundary << stereo >>

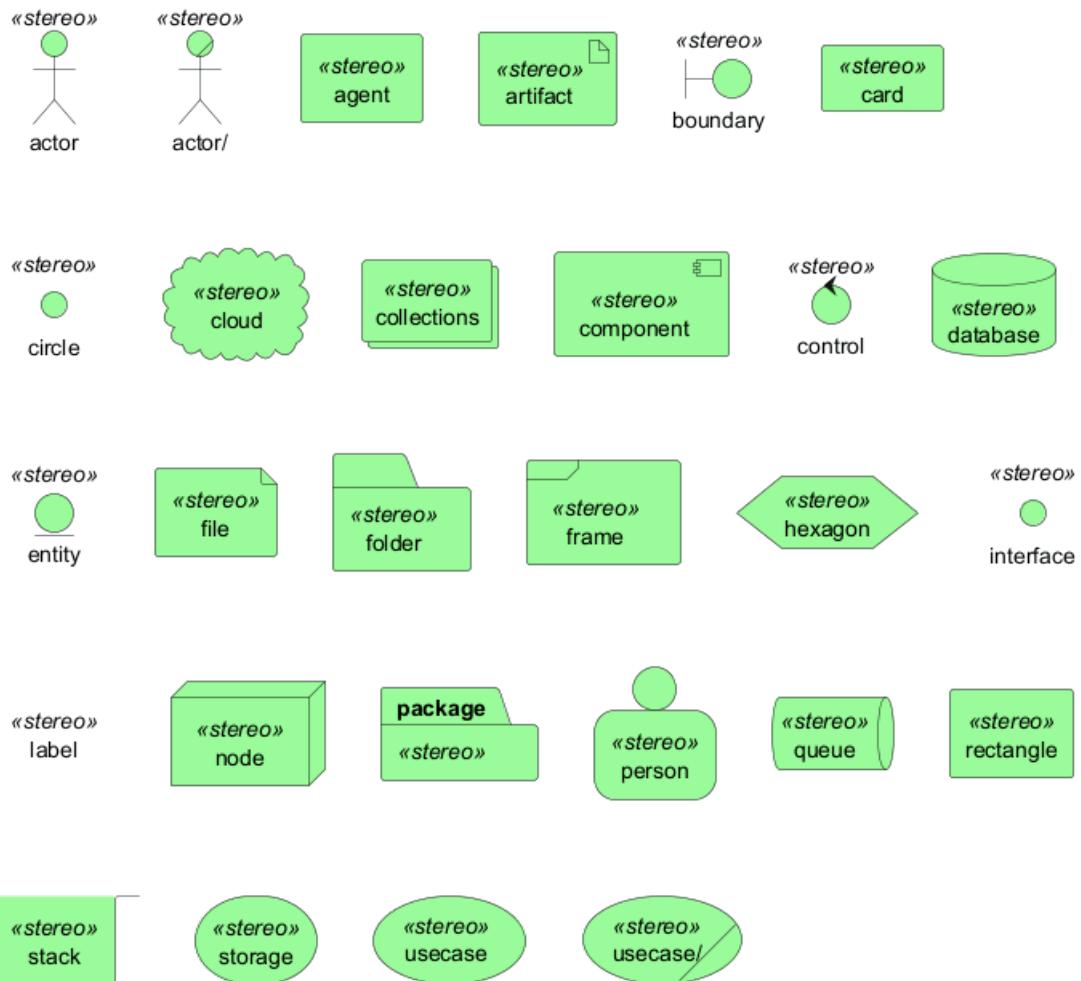
```



```

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

```



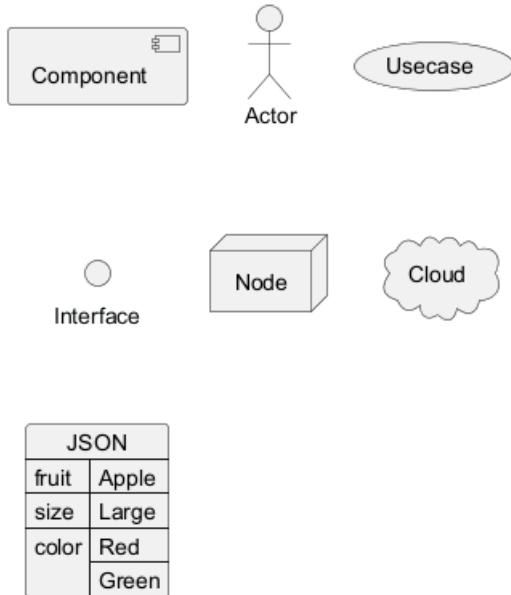
8.17 Display JSON Data on Deployment diagram

8.17.1 Simple example

```
@startuml
allowmixing

component Component
actor Actor
usecase Usecase
() Interface
node Node
cloud Cloud

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



[Ref. QA-15481]

For another example, see on JSON page.

8.18 Mixing Deployment (Usecase, Component, Deployment) element within a Class or Object diagram

In order to add a Deployment element or a State element within a Class or Object diagram, you can use the `allowmixing` or `allow_mixing` directive.

8.18.1 Mixing all elements

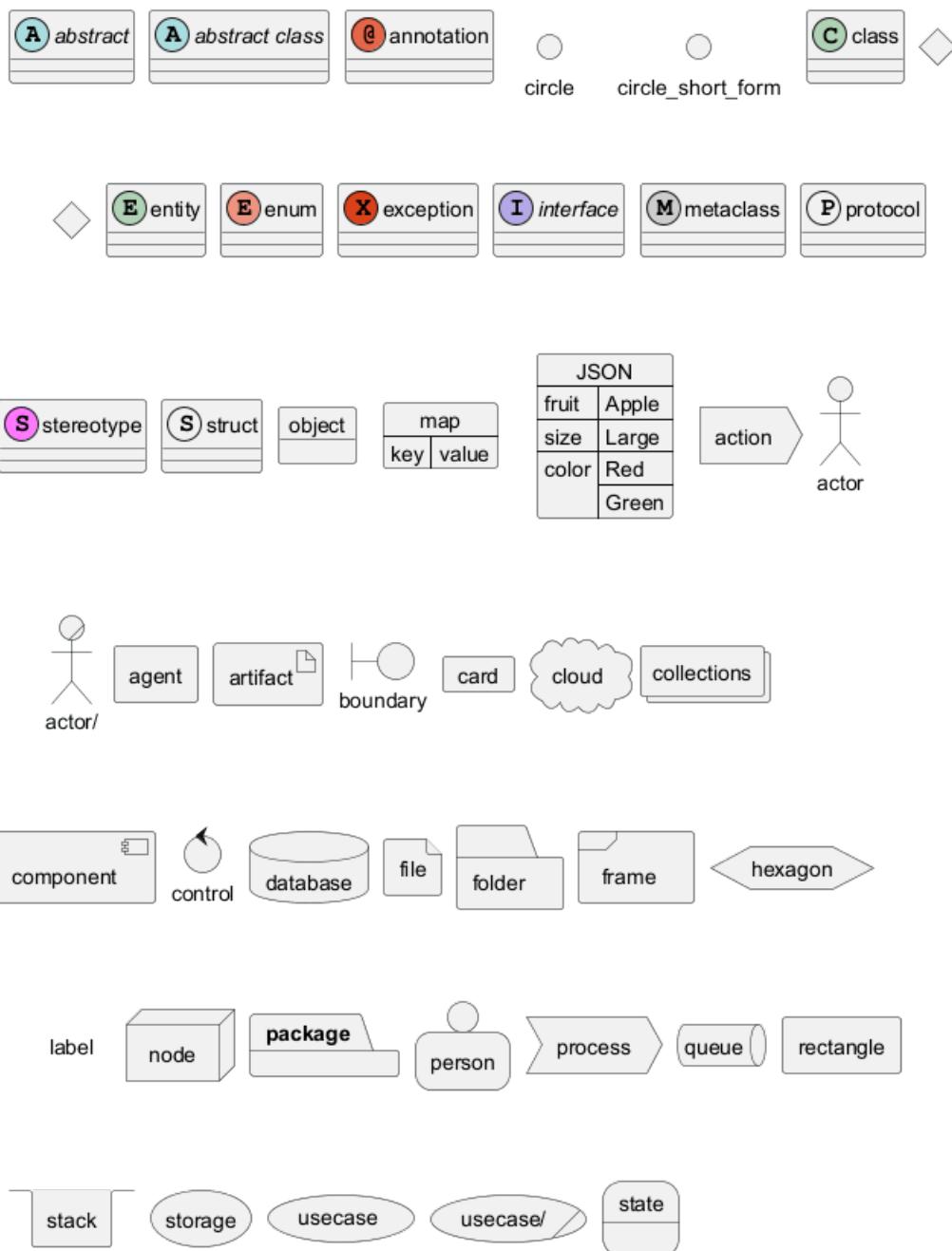
```
@startuml
allowmixing

skinparam nodesep 10
abstract      abstract
abstract class "abstract class"
annotation    annotation
```



```
circle      circle
()          circle_short_form
class       class
diamond     diamond
<>         diamond_short_form
entity      entity
enum        enum
exception   exception
interface   interface
metaclass  metaclass
protocol   protocol
stereotype stereotype
struct      struct
object      object
map map {
    key => value
}
json JSON {
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}
action action
actor actor
actor/ "actor/"
agent agent
artifact artifact
boundary boundary
card card
circle circle
cloud cloud
collections collections
component component
control control
database database
entity entity
file file
folder folder
frame frame
hexagon hexagon
interface interface
label label
node node
package package
person person
process process
queue queue
rectangle rectangle
stack stack
storage storage
usecase usecase
usecase/ "usecase/"
state state
@enduml
```





[Ref. QA-2335 and QA-5329]

8.19 Port [port, portIn, portOut]

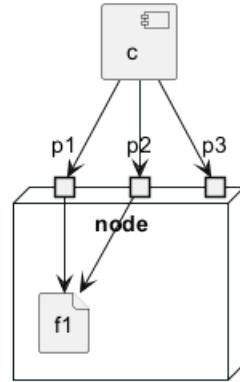
You can add port with port, portin and portout keywords.

8.19.1 Port

```
@startuml
[c]
node node {
    port p1
    port p2
    port p3
    file f1
}
```



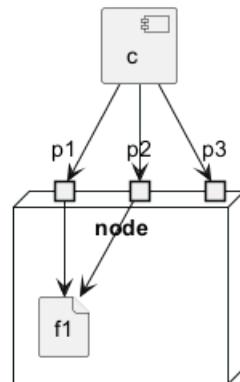
```
c --> p1
c --> p2
c --> p3
p1 --> f1
p2 --> f1
@enduml
```



8.19.2 PortIn

```
@startuml
[c]
node node {
    portin p1
    portin p2
    portin p3
    file f1
}

c --> p1
c --> p2
c --> p3
p1 --> f1
p2 --> f1
@enduml
```



8.19.3 PortOut

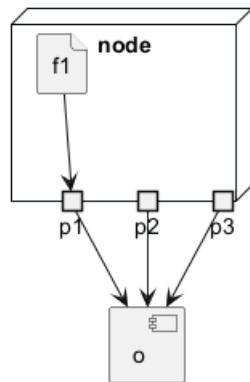
```
@startuml
node node {
    portout p1
    portout p2
```



```

portout p3
file f1
}
[o]
p1 --> o
p2 --> o
p3 --> o
f1 --> p1
@enduml

```



8.19.4 Mixing PortIn & PortOut

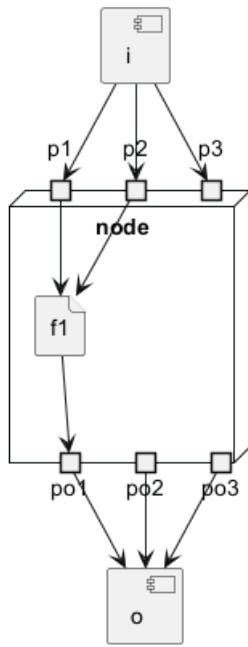
```

@startuml
[i]
node node {
    portin p1
    portin p2
    portin p3
    portout po1
    portout po2
    portout po3
    file f1
}
[o]

i --> p1
i --> p2
i --> p3
p1 --> f1
p2 --> f1
po1 --> o
po2 --> o
po3 --> o
f1 --> po1
@enduml

```





8.20 Change diagram orientation

You can change (whole) diagram orientation with:

- top to bottom direction (*by default*)
- left to right direction

8.20.1 Top to bottom (*by default*)

8.20.2 With Graphviz (layout engine *by default*)

The main rule is: Nested element first, then simple element.

```

@startuml
card a
card b
package A {
    card a1
    card a2
    card a3
    card a4
    card a5
    package sub_a {
        card sa1
        card sa2
        card sa3
    }
}

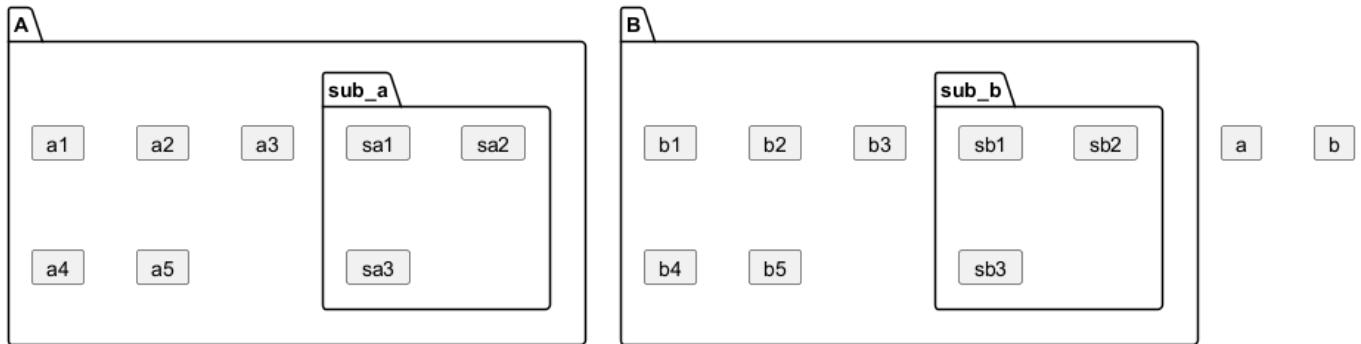
package B {
    card b1
    card b2
    card b3
    card b4
    card b5
    package sub_b {
        card sb1
        card sb2
    }
}
  
```



```

card sb3
}
}
@enduml

```



8.20.3 With Smetana (*internal layout engine*)

The main rule is the opposite: **Simple element first, then nested element.**

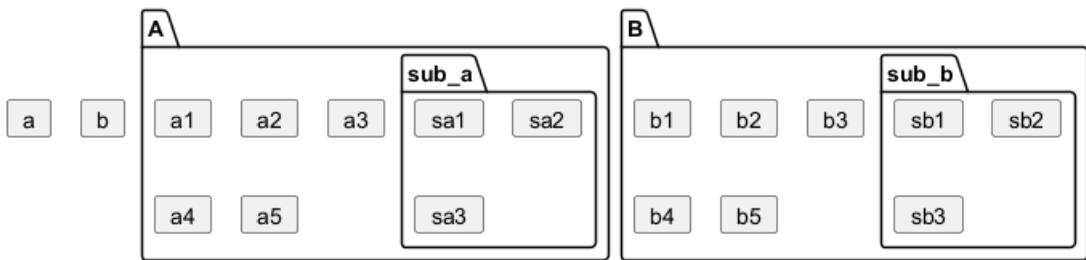
```

@startuml
!pragma layout smetana
card a
card b
package A {
    card a1
    card a2
    card a3
    card a4
    card a5
    package sub_a {
        card sa1
        card sa2
        card sa3
    }
}

package B {
    card b1
    card b2
    card b3
    card b4
    card b5
    package sub_b {
        card sb1
        card sb2
        card sb3
    }
}
@enduml

```



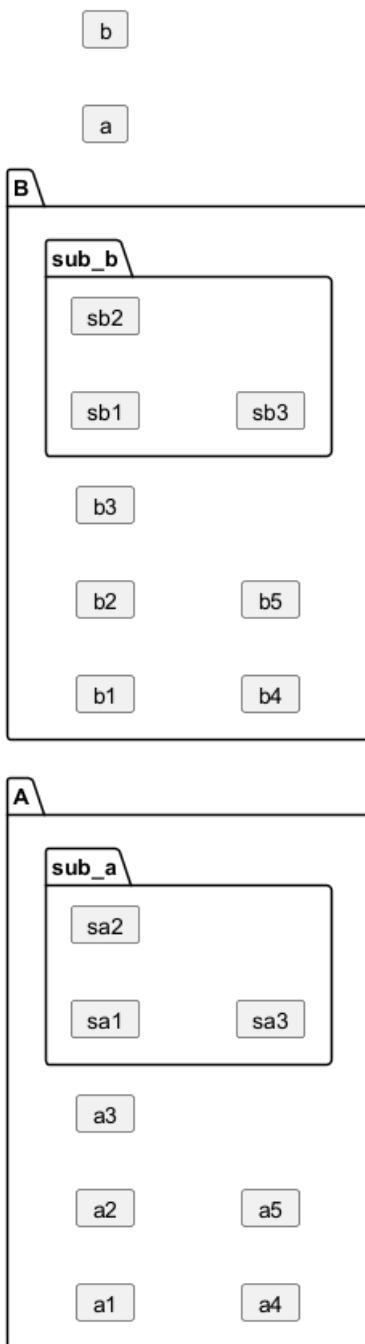


8.20.4 Left to right

8.20.5 With Graphviz (layout engine by default)

```
@startuml
left to right direction
card a
card b
package A {
    card a1
    card a2
    card a3
    card a4
    card a5
    package sub_a {
        card sa1
        card sa2
        card sa3
    }
}
package B {
    card b1
    card b2
    card b3
    card b4
    card b5
    package sub_b {
        card sb1
        card sb2
        card sb3
    }
}
@enduml
```



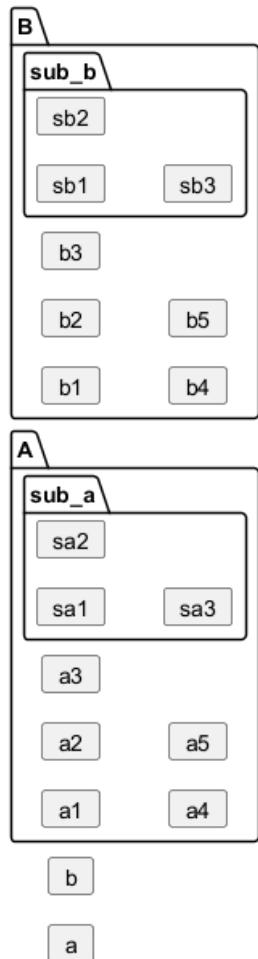


8.20.6 With Smetana (internal layout engine)

```
@startuml  
!pragma layout smetana  
left to right direction  
card a  
card b  
package A {  
    card a1  
    card a2  
    card a3  
    card a4  
    card a5  
    package sub_a {  
        card sa1
```



```
card sa2  
card sa3  
}  
}  
  
package B {  
    card b1  
    card b2  
    card b3  
    card b4  
    card b5  
    package sub_b {  
        card sb1  
        card sb2  
        card sb3  
    }  
}  
}@enduml
```



9 Диаграмма состояний

Диаграммы состояний позволяют наглядно представить различные состояния, в которых может находиться система или объект, а также переходы между этими состояниями. Они необходимы для моделирования динамического поведения систем, отражая их реакцию на различные события во времени. Диаграммы состояний отображают жизненный цикл системы, облегчая понимание, проектирование и оптимизацию ее поведения.

Использование **PlantUML** для создания диаграмм состояний имеет ряд преимуществ:

- **Текстовый язык:** Быстрое определение и визуализация состояний и переходов без необходимости рисования вручную.
- **Эффективность и согласованность:** Обеспечивают упорядоченное создание диаграмм и легкий контроль версий.
- **Универсальность:** Интеграция с различными платформами документирования и поддержка различных форматов вывода.
- **Open-Source & Community Support:** Опирается на **сильное сообщество**, которое постоянно вносит свой вклад в его совершенствование и предлагает бесценные ресурсы.

9.1 Простое состояние

Для изображения начального и конечного псевдосостояний используется [*].

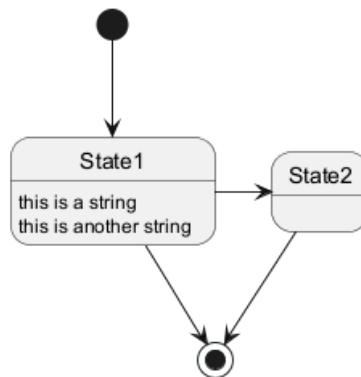
Используйте --> для изображения переходов.

```
@startuml
```

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

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

```
@enduml
```



9.2 Change state rendering

You can use hide empty description to render state as simple box.

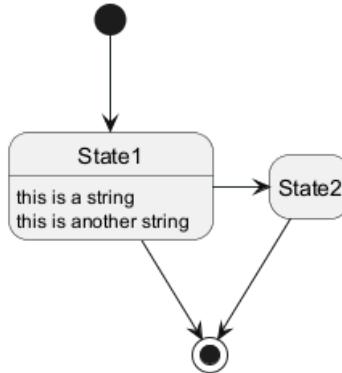
```
@startuml
hide empty description
[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string
```



```

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

```



9.3 Составное состояние

Также можно изображать составные состояния. Для этого его следует объявить, используя конструкцию `state`.

9.3.1 Внутренние состояния

```

@startuml
scale 350 width
[*] --> NotShooting

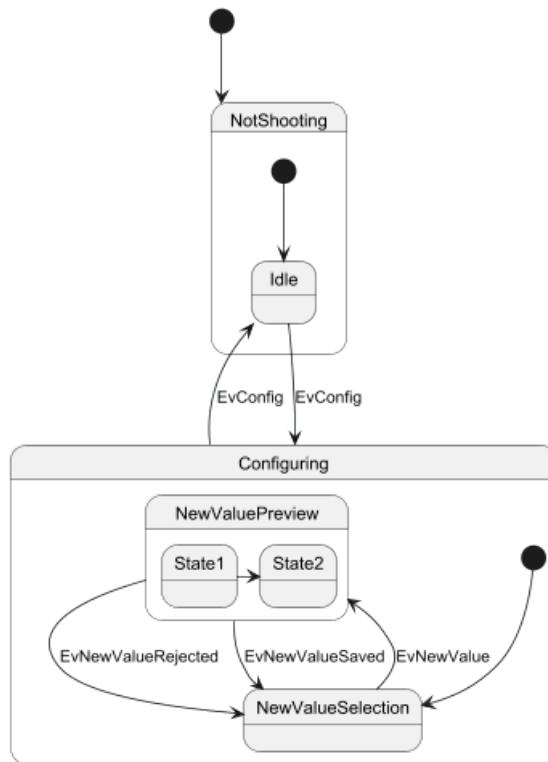
state NotShooting {
    [*] --> Idle
    Idle --> Configuring : EvConfig
    Configuring --> Idle : EvConfig
}

state Configuring {
    [*] --> NewValueSelection
    NewValueSelection --> NewValuePreview : EvnewValue
    NewValuePreview --> NewValueSelection : EvnewValueRejected
    NewValuePreview --> NewValueSelection : EvnewValueSaved

    state NewValuePreview {
        State1 -> State2
    }
}
@enduml

```





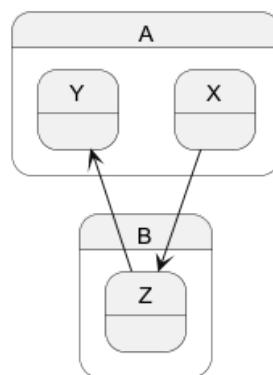
9.3.2 Внутреннее состояние к внутреннему состоянию

```

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

state B {
    state Z {
    }
}

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



[Ref. QA-3300]



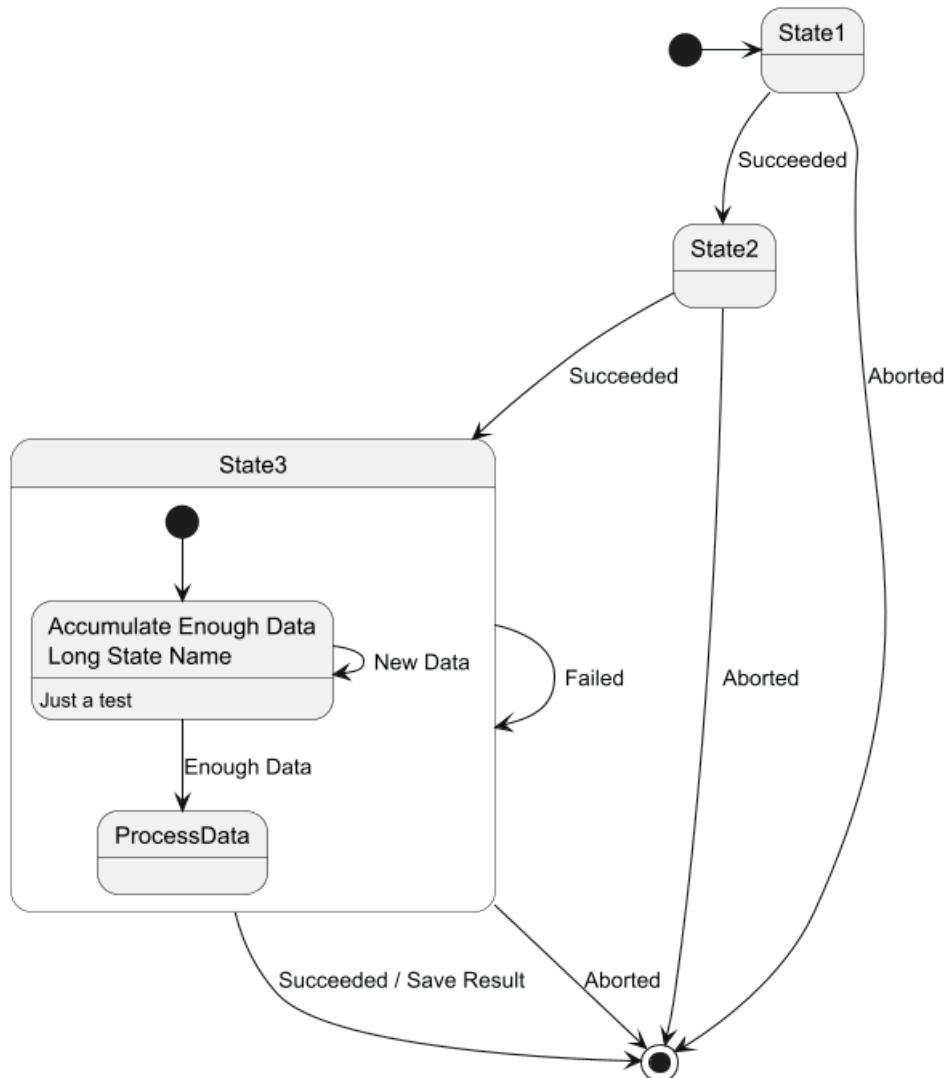
9.4 Длинные имена

Вы также можете использовать ключевое слово `state` для сокращения длинного имени состояния.

```
@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
```

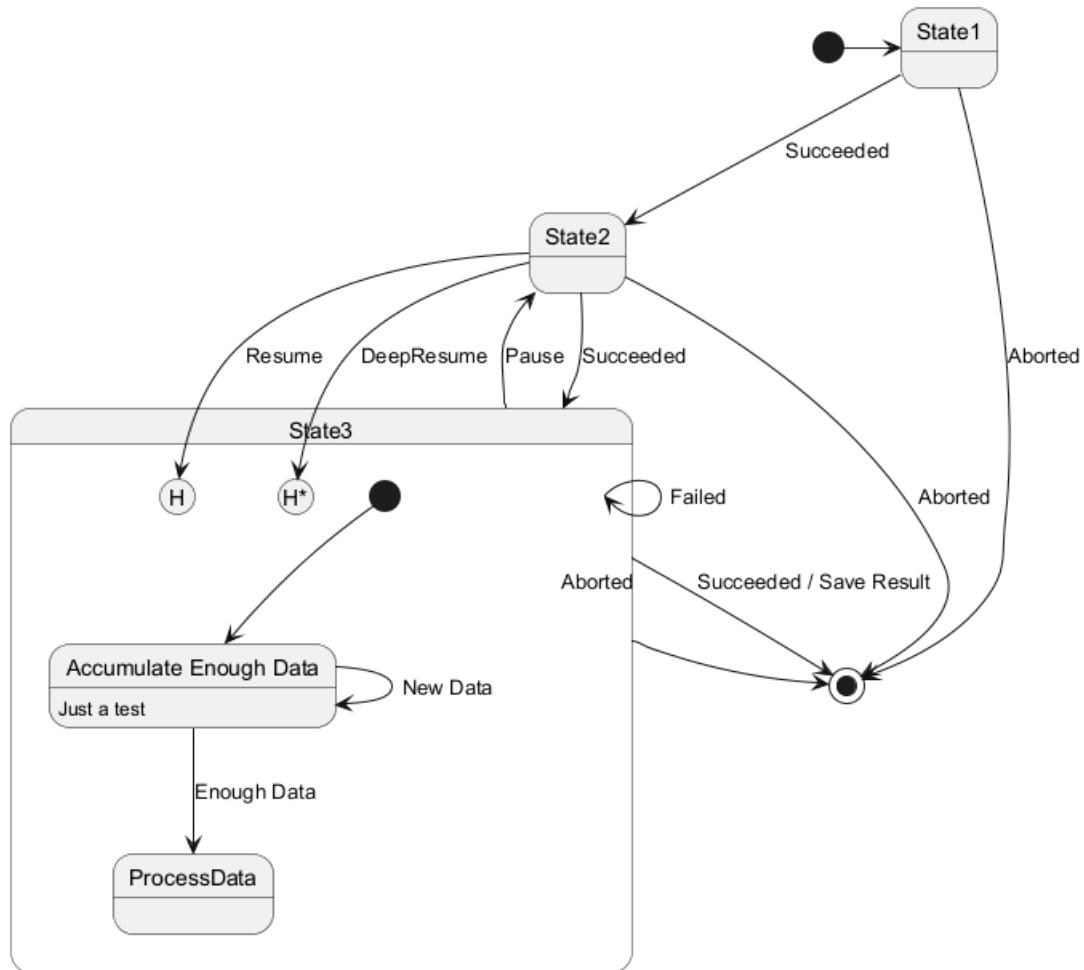
```
@enduml
```



9.5 History [[H], [H*]]

You can use [H] for the history and [H*] for the deep history of a substate.

```
@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 Ветвление [fork, join]

Также можно использовать стереотипы <<fork>> и <<join>> для создания ветвления и слияния обратно.



```

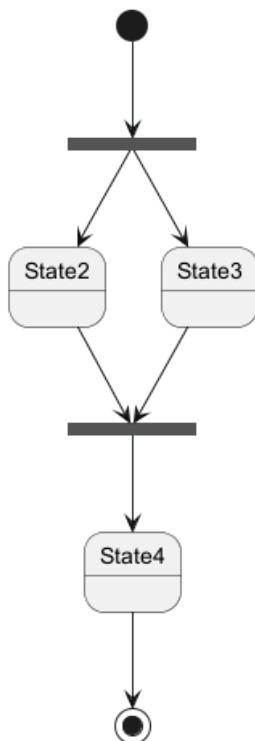
@startuml

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 Параллельные состояния

Используя -- или ||, вы можете объявлять параллельные подсостояния внутри составного состояния.

9.7.1 Горизонтальное разделение --

```

@startuml
[*] --> Active

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

```

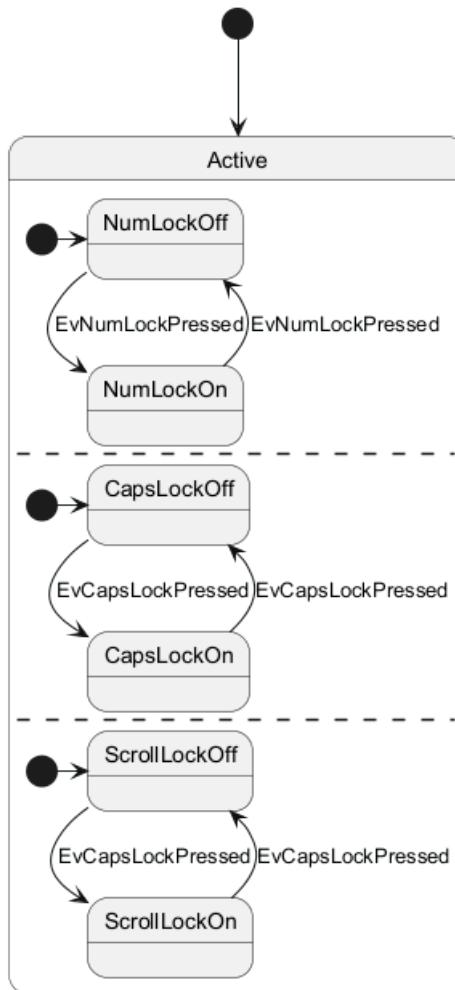


```

    ScrollLockOn --> ScrollLockOff : EvCapsLockPressed
}

@enduml

```



9.7.2 Вертикальное разделение ||

```

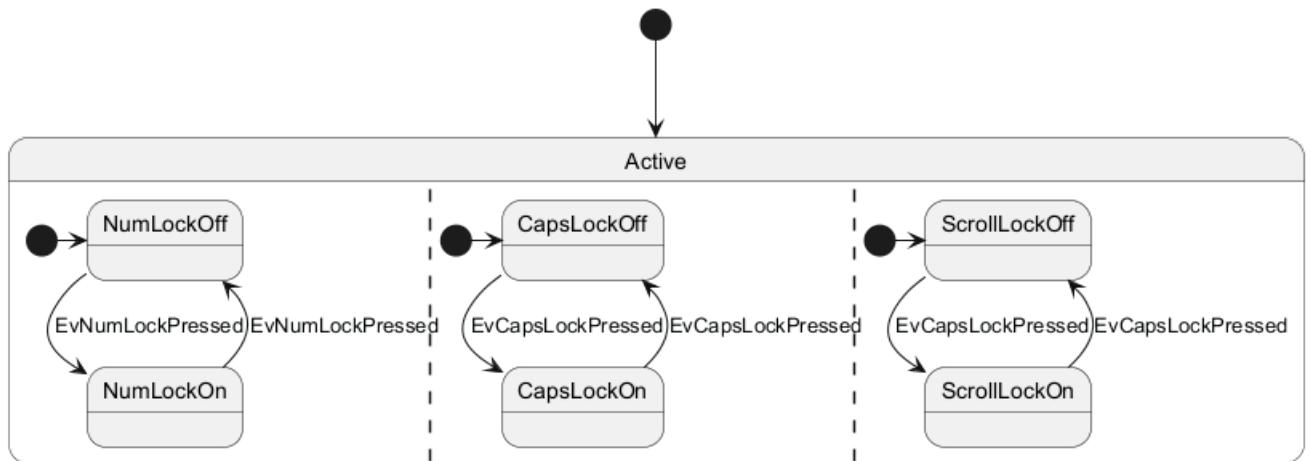
@startuml
[*] --> Active

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

```

```
@enduml
```





[Ref. [QA-3086](<https://forum.plantuml.net/3086/state-diagram-concurrent-state-horizontal-line>)]

9.8 Условия [choice]

Стереотип <<choice>> может быть использован для создания состояний-условий.

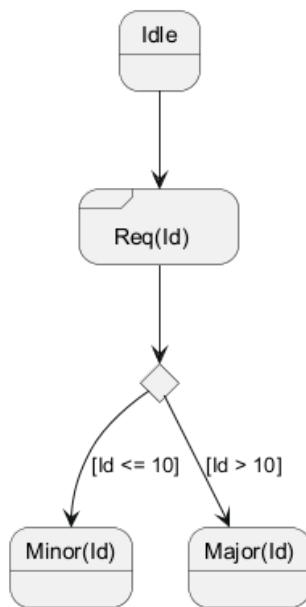
```

@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 Stereotypes full example [start, choice, fork, join, end, history, history*]

9.9.1 Start, choice, fork, join, end

```
@startuml
```



```

state start1 <<start>>
state choice1 <<choice>>
state fork1 <<fork>>
state join2 <<join>>
state end3 <<end>>

[*] --> choice1 : from start\nto choice
start1 --> choice1 : from start stereo\nto choice

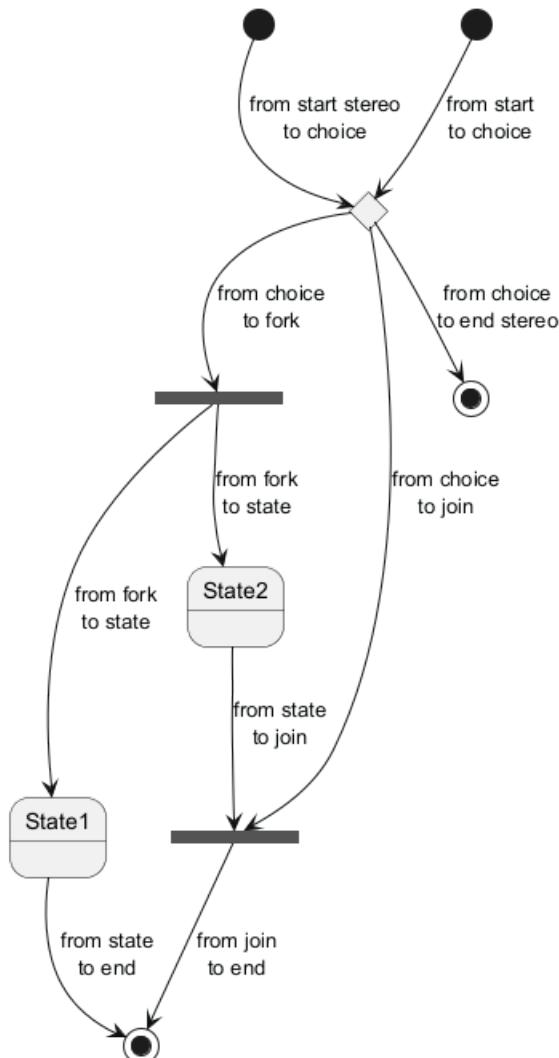
choice1 --> fork1 : from choice\nto fork
choice1 --> join2 : from choice\nto join
choice1 --> end3 : from choice\nto end stereo

fork1 ---> State1 : from fork\nto state
fork1 ---> State2 : from fork\nto state

State2 --> join2 : from state\nto join
State1 --> [*] : from state\nto end

join2 --> [*] : from join\nto end
@enduml

```

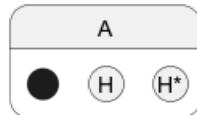


[Ref. QA-404, QA-1159 and GH-887]



9.9.2 History, history*

```
@startuml
state A {
    state s1 as "Start 1" <<start>>
    state s2 as "H 2" <<history>>
    state s3 as "H 3" <<history*>>
}
@enduml
```



[Ref. QA-16824]

9.9.3 Minimal example with all stereotypes

```
@startuml
state start1 <<start>>
state choice1 <<choice>>
state fork1 <<fork>>
state join2 <<join>>
state end3 <<end>>
state sdlreceive <<sdlreceive>>
state history <<history>>
state history2 <<history*>>
@enduml
```



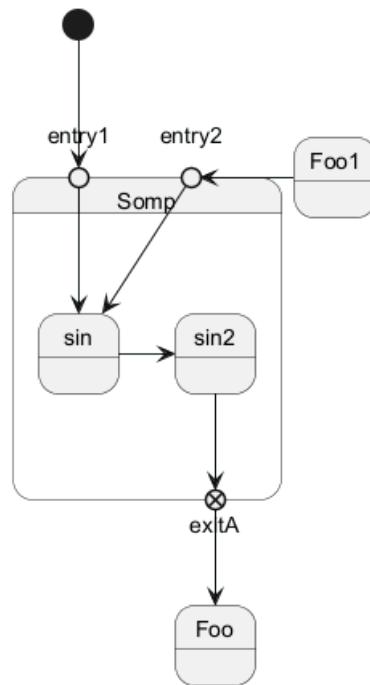
[Ref. QA-19174]

9.10 Point [entryPoint, exitPoint]

You can add **point** with <<entryPoint>> and <<exitPoint>> stereotypes:

```
@startuml
state Somp {
    state entry1 <<entryPoint>>
    state entry2 <<entryPoint>>
    state sin
    entry1 --> sin
    entry2 -> sin
    sin -> sin2
    sin2 --> exitA <<exitPoint>>
}
[*] --> entry1
exitA --> Foo
Foo1 -> entry2
@enduml
```





9.11 Pin [inputPin, outputPin]

You can add **pin** with `<<inputPin>>` and `<<outputPin>>` stereotypes:

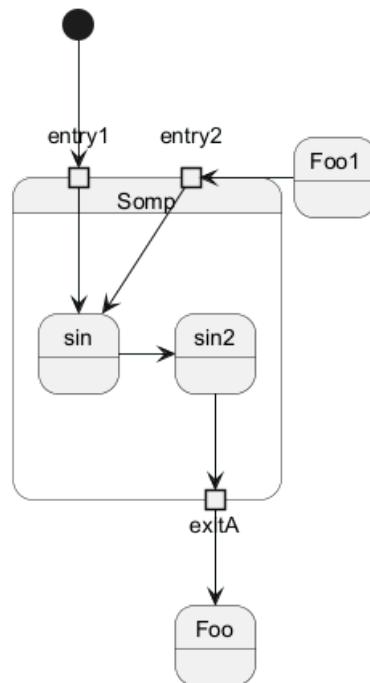
```

@startuml
state Somp {
    state entry1 <<inputPin>>
    state entry2 <<inputPin>>
    state sin
    entry1 --> sin
    entry2 -> sin
    sin -> sin2
    sin2 --> exitA <<outputPin>>
}

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

```





[Ref. QA-4309]

9.12 Expansion [expansionInput, expansionOutput]

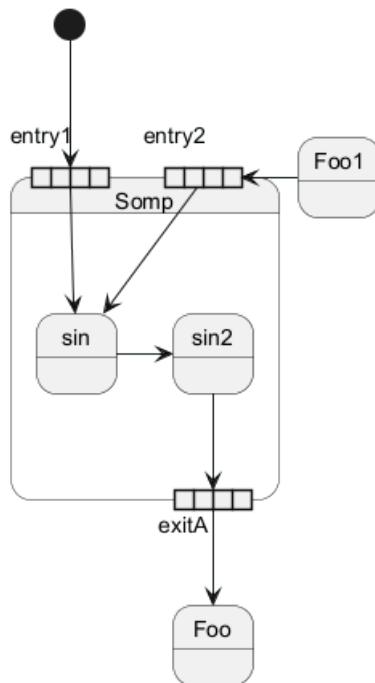
You can add **expansion** with <<expansionInput>> and <<expansionOutput>> stereotypes:

```

@startuml
state Somp {
    state entry1 <<expansionInput>>
    state entry2 <<expansionInput>>
    state sin
    entry1 --> sin
    entry2 -> sin
    sin -> sin2
    sin2 --> exitA <<expansionOutput>>
}

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





[Ref. QA-4309]

9.13 Направления стрелок

Для изображения стрелок перехода горизонтально используется оператор `->`. Следующий синтаксис позволяет задать другое направление.

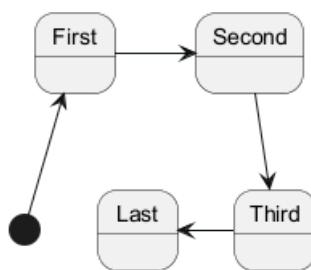
- `-down->` или `-->`
- `-right->` или `->` (по умолчанию)
- `-left->`
- `-up->`

`@startuml`

```

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

`@enduml`



Вы также можете сокращать слова в описании стрелок (например, `-d->` или `-do->` вместо `-down->`).

Не следует злоупотреблять этой функциональностью: *GraphViz* в большинстве случаев дает хороший результат без лишних манипуляций.

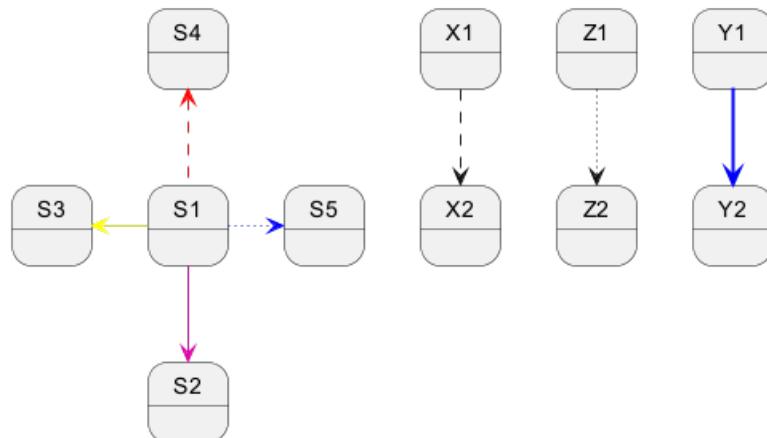


9.14 Изменение цвета и отображения

Можно использовать параметр цвета вместе со стилем.

```
@startuml
State S1
State S2
State S3
S1 -[#DD00AA]-> S2
S1 -left[#yellow]-> S3
S1 -up[#red,dashed]-> S4
S1 -right[dotted,#blue]-> S5
```

```
X1 -[dashed]-> X2
Z1 -[dotted]-> Z2
Y1 -[#blue,bold]-> Y2
@enduml
```



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

9.15 Заметки

К состоянию можно добавлять заметки, используя специальные ключевые слова: `note left of`, `note right of`, `note top of`, `note bottom of`.

Заметки можно определять в несколько строк.

```
@startuml
```

```

[*] --> Active
Active --> Inactive

note left of Active : this is a short\nnote

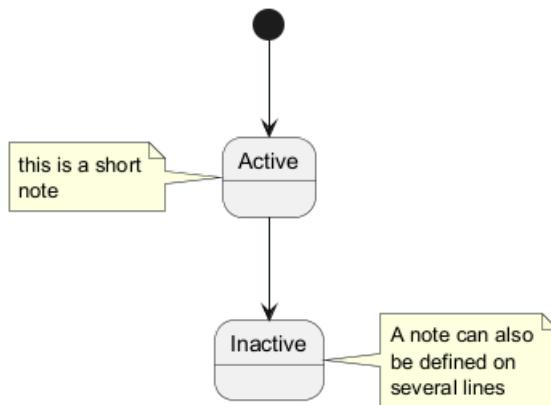
note right of Inactive
    A note can also
    be defined on
    several lines
end note

[*] --> Active
Active --> Inactive

```

```
@enduml
```





Можно создавать заметки, не привязанные ни к какому объекту.

@startuml

```

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

@enduml



9.16 Заметки о переходах

Можно оставлять заметки о переходах состояний или связей с помощью note on link.

```

@startuml
[*] --> State1
State1 --> State2
note on link
  
```

```

end note
@enduml
  
```



9.17 Еще о заметках

Также заметки можно прикреплять к составным состояниям.

@startuml

```

[*] --> NotShooting

state "Not Shooting State" as NotShooting {
    state "Idle mode" as Idle
  
```



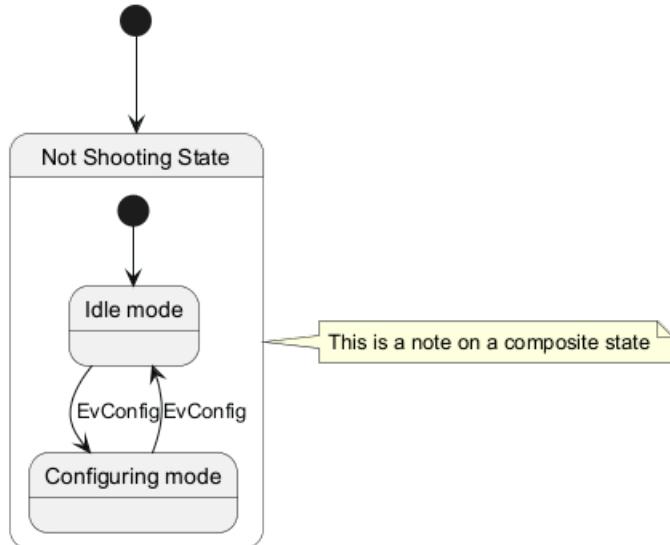
```

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

```



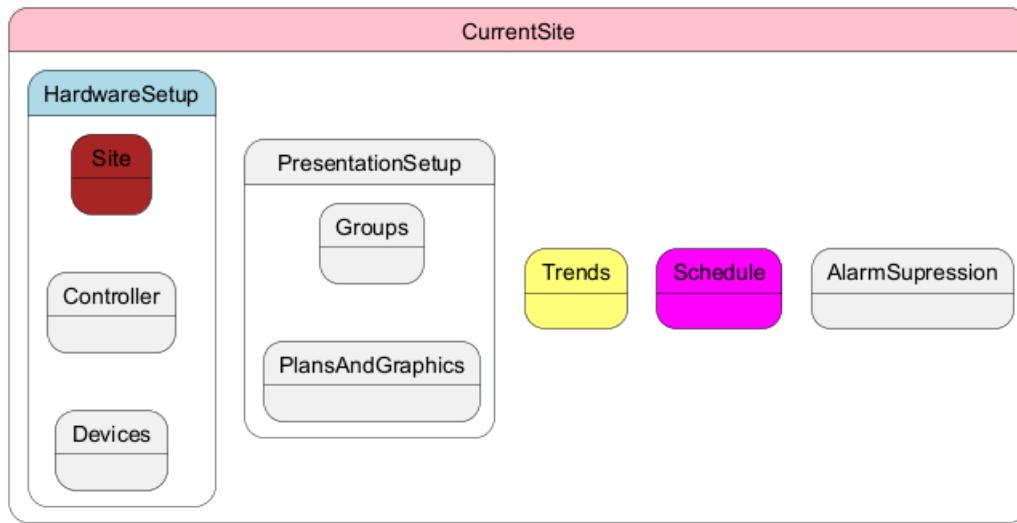
9.18 *Inline color*

```

@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

```





[Ref. QA-1812]

9.19 Skinparam

Вы можете использовать команду `skinparam` для изменения шрифтов и цветов диаграммы

Вы можете использовать данную команду :

- В определении диаграммы, как любую другую команду,
- В подключенном файле,
- В конфигурационном файле, указанном в командной строке в задании ANT.

Вы можете задавать цвета и шрифты для именованных шаблонов состояний.

```

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

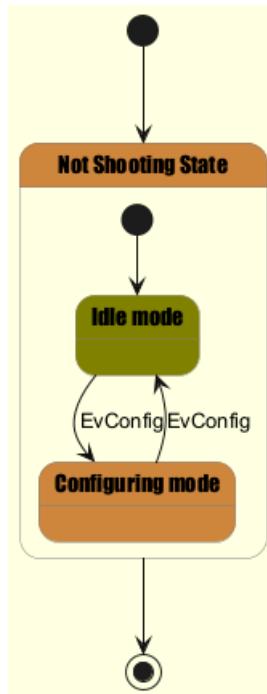
[*] --> NotShooting

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>

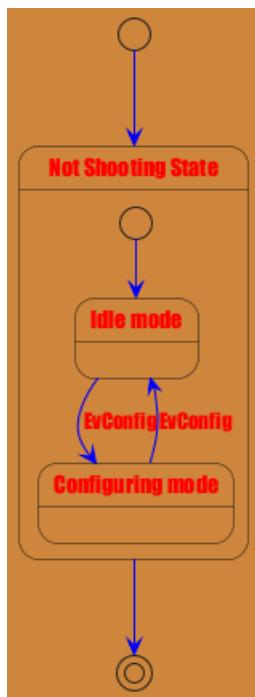
[*] --> NotShooting

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

```

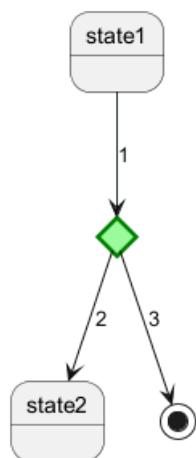




```

@startuml
<style>
  diamond {
    BackgroundColor #palegreen
    LineColor #green
    LineThickness 2.5
  }
</style>
state state1
state state2
state choice1 <<choice>>
state end3      <<end>>

state1 --> choice1 : 1
choice1 --> state2 : 2
choice1 --> end3   : 3
@enduml
  
```



[Ref. GH-880]



9.21 Change state color and style (inline style)

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

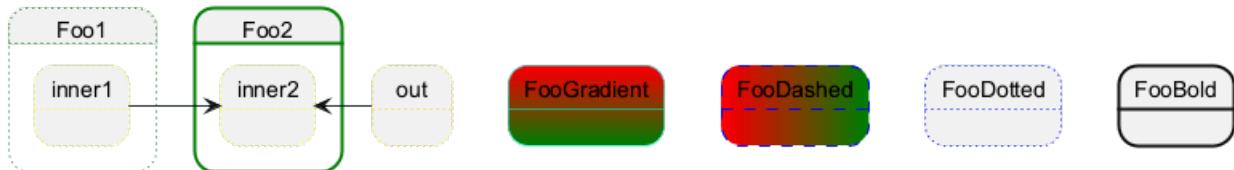
- `#color ##[style]color`

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

```
@startuml
state FooGradient #red-green ##00FFFF
state FooDashed #red|green ##[dashed]blue {
}
state FooDotted ##[dotted]blue {
}
state FooBold ##[bold] {
}
state Foo1 ##[dotted]green {
state inner1 ##[dotted]yellow
}

state out ##[dotted]gold

state Foo2 ##[bold]green {
state inner2 ##[dotted]yellow
}
inner1 -> inner2
out -> inner2
@enduml
```



[Ref. QA-1487]

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

TODO: FIXME text:color seems not to be taken into account **TODO:** FIXME

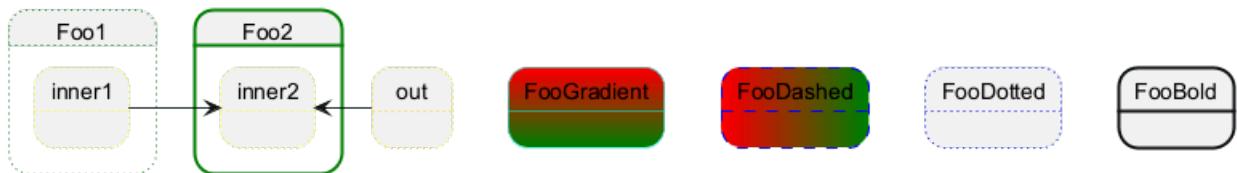
```
@startuml
@startuml
state FooGradient #red-green;line:00FFFF
state FooDashed #red|green;line.dashed;line:blue {
}
state FooDotted #line.dotted;line:blue {
}
state FooBold #line.bold {
}
state Foo1 #line.dotted;line:green {
state inner1 #line.dotted;line:yellow
}

state out #line.dotted;line:gold

state Foo2 #line.bold;line:green {
state inner2 #line.dotted;line:yellow
}
inner1 -> inner2
out -> inner2
```



```
@enduml
@enduml
```



```
@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]

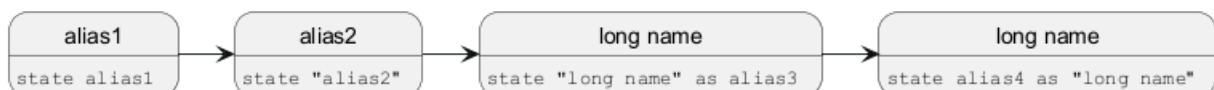
9.22 Alias

With State you can use alias, like:

```
@startuml
state alias1
state "alias2"
state "long name" as alias3
state alias4 as "long name"

alias1 : ""state alias1"""
alias2 : ""state "alias2"""
alias3 : ""state "long name" as alias3"""
alias4 : ""state alias4 as "long name"""

alias1 -> alias2
alias2 -> alias3
alias3 -> alias4
@enduml
```



or:

```
@startuml
state alias1 : ""state alias1"""
state "alias2" : ""state "alias2"""
state "long name" as alias3 : ""state "long name" as alias3"""
state alias4 as "long name" : ""state alias4 as "long name"""

alias1 -> alias2
alias2 -> alias3
alias3 -> alias4
@enduml
```





[Ref. QA-1748, QA-14560]

9.23 Display JSON Data on State diagram

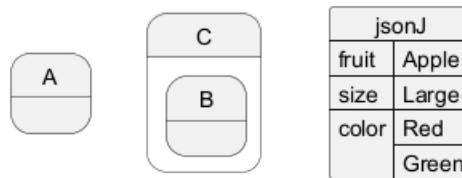
9.23.1 Simple example

```

@startuml
state "A" as stateA
state "C" as stateC {
    state B
}

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

```



[Ref. QA-17275]

For another example, see on JSON page.

9.24 State description

You can add description to a state or to a composite state.

```

@startuml
hide empty description

state s0

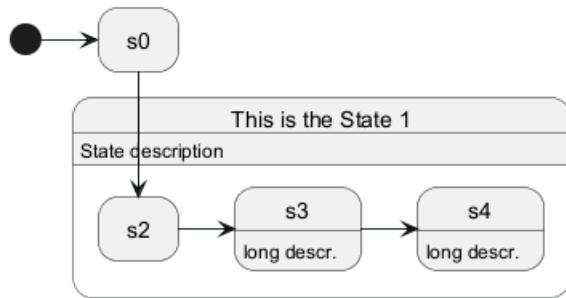
state "This is the State 1" as s1 {
    s1: State description
    state s2
    state s3: long descr.
    state s4
    s4: long descr.
}

[*] -> s0
s0 --> s2

s2 -> s3
s3 -> s4
@enduml

```



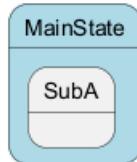


[Ref. QA-16719]

9.25 Style for Nested State Body

```
@startuml
<style>
.foo {
    state,stateBody {
        BackGroundColor lightblue;
    }
}
</style>

state MainState <><> {
    state SubA
}
@enduml
```



[Ref. QA-16774]



10 Timing Diagram

A Timing Diagram in UML is a specific type of **interaction diagram** that visualizes the **timing constraints** of a system. It focuses on the **chronological order of events**, showcasing how different objects interact with each other over time. **Timing diagrams** are especially useful in **real-time systems** and **embedded systems** to understand the behavior of objects throughout a given period.

10.1 Declaring element or participant

You declare participant using the following keywords, depending on how you want them to be drawn.

Keyword	Description
analog	An analog signal is continuous, and the values are linearly interpolated between the given setpoints
binary	A binary signal restricted to only 2 states
clock	A clocked signal that repeatedly transitions from high to low, with a period, and an optional pulse and offset
concise	A simplified concise signal designed to show the movement of data (great for messages)
robust	A robust complex line signal designed to show the transition from one state to another (can have many states)

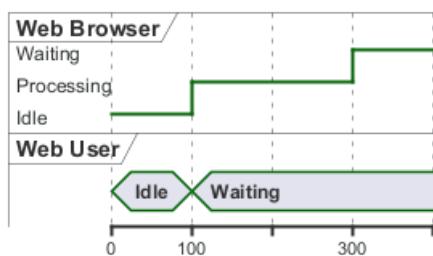
You define state change using the @ notation, and the is verb.

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

```
@0
WU is Idle
WB is Idle
```

```
@100
WU is Waiting
WB is Processing
```

```
@300
WB is Waiting
@enduml
```



```
@startuml
clock "Clock_0" as C0 with period 50
clock "Clock_1" as C1 with period 50 pulse 15 offset 10
binary "Binary" as B
concise "Concise" as C
robust "Robust" as R
analog "Analog" as A
```

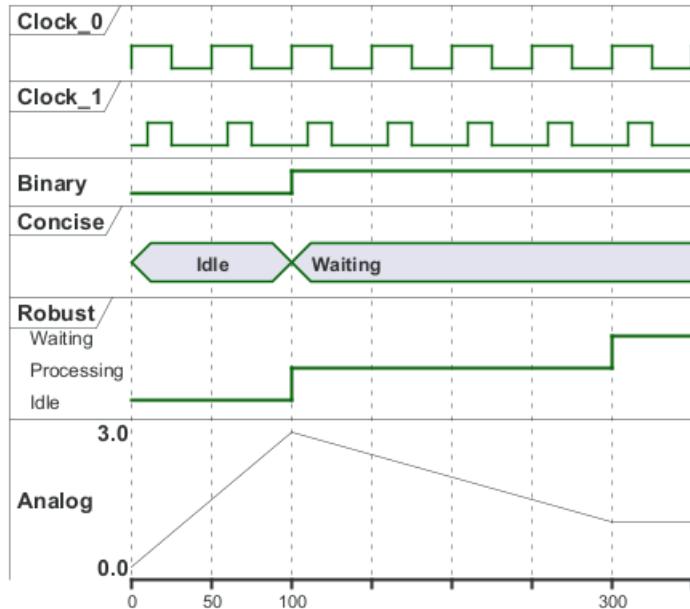
```
@0
C is Idle
R is Idle
A is 0
```

```
@100
```



```
B is high
C is Waiting
R is Processing
A is 3
```

```
@300
R is Waiting
A is 1
@enduml
```



[Ref. QA-14631, QA-14647 and QA-11288]

10.2 Binary and Clock

It's also possible to have binary and clock signal, using the following keywords:

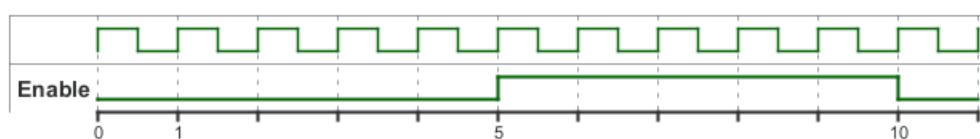
- `binary`
- `clock`

```
@startuml
clock clk with period 1
binary "Enable" as EN
```

```
@0
EN is low
```

```
@5
EN is high
```

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



10.3 Adding message

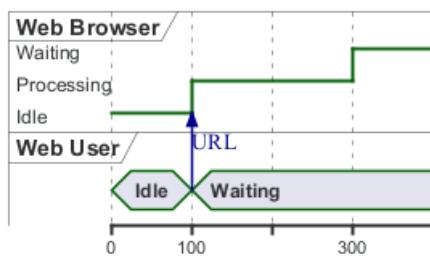
You can add message using the following syntax.

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

```
@0
WU is Idle
WB is Idle
```

```
@100
WU -> WB : URL
WU is Waiting
WB is Processing
```

```
@300
WB is Waiting
@enduml
```



10.4 Relative time

It is possible to use relative time with @.

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

```
@0
WU is Idle
WB is Idle
DNS is Idle
```

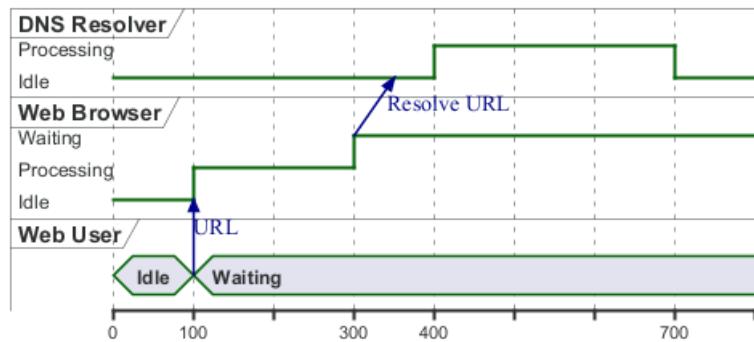
```
@+100
WU -> WB : URL
WU is Waiting
WB is Processing
```

```
@+200
WB is Waiting
WB -> DNS@+50 : Resolve URL
```

```
@+100
DNS is Processing
```

```
@+300
DNS is Idle
@enduml
```





10.5 Anchor Points

Instead of using absolute or relative time on an absolute time you can define a time as an anchor point by using the `@XX as :<anchor point name>`

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

```
@0 as :start
@5 as :en_high
@10 as :en_low
@:en_high-2 as :en_highMinus2
```

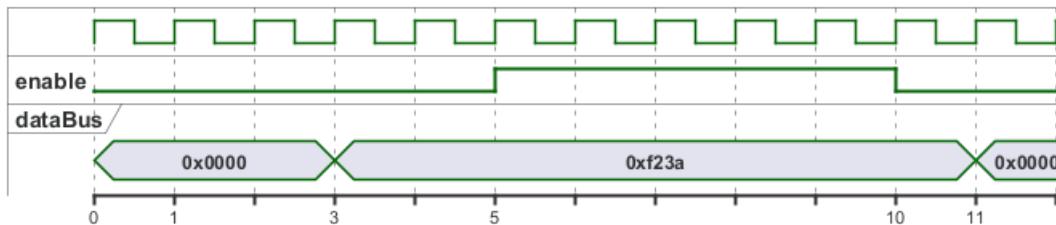
```
@:start
EN is low
db is "0x0000"
```

```
@:en_high
EN is high
```

```
@:en_low
EN is low
```

```
@:en_highMinus2
db is "0xf23a"
```

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



10.6 Participant oriented

Rather than declare the diagram in chronological order, you can define it by participant.

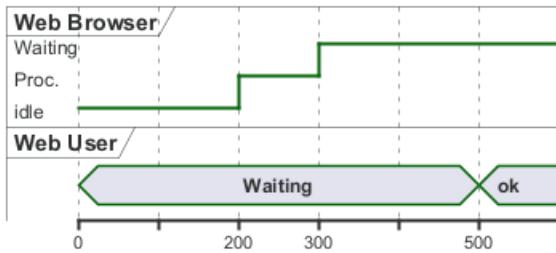
```
@startuml
```



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

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

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

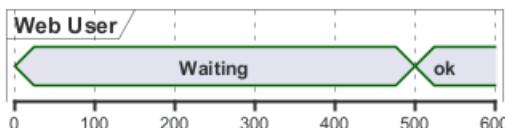


10.7 Setting scale

You can also set a specific scale.

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

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



When using absolute Times/Dates, 1 "tick" is equivalent to 1 second.

```
@startuml
concise "Season" as S
'30 days is scaled to 50 pixels
scale 2592000 as 50 pixels
```

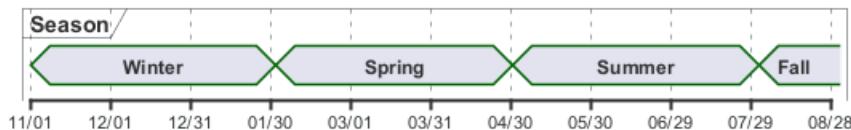
```
@2000/11/01
S is "Winter"
```

```
@2001/02/01
S is "Spring"
```

```
@2001/05/01
S is "Summer"
```

```
@2001/08/01
S is "Fall"
@enduml
```





10.8 Initial state

You can also define an initial state.

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

WB is Initializing
WU is Absent

```
@WB
0 is idle
+200 is Processing
+100 is Waiting
```

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



10.9 Intricated state

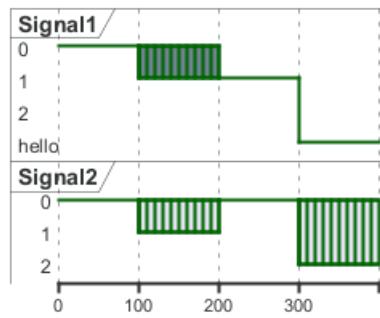
A signal could be in some undefined state.

10.9.1 Intricated or undefined robust state

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



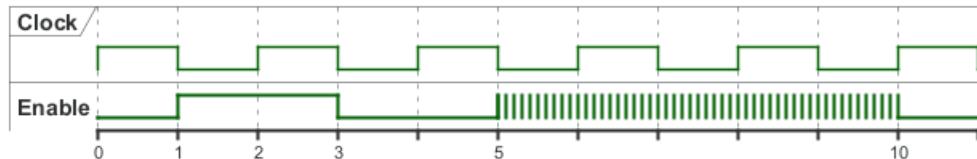
```
S2 is {0,2}
@enduml
```



10.9.2 Intricated or undefined binary state

```
@startuml
clock "Clock" as C with period 2
binary "Enable" as EN
```

```
@0
EN is low
@1
EN is high
@3
EN is low
@5
EN is {low,high}
@10
EN is low
@enduml
```



[Ref. QA-11936 and QA-15933]

10.10 Hidden state

It is also possible to hide some state.

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

```
@0
WU is {-}
```

```
@100
WU is A1
```

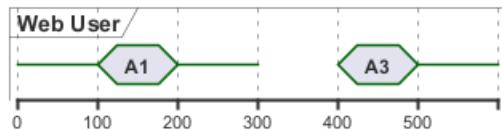
```
@200
WU is {-}
```

```
@300
WU is {hidden}
```



```
@400
WU is A3
```

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



```
@startuml
scale 1 as 50 pixels
```

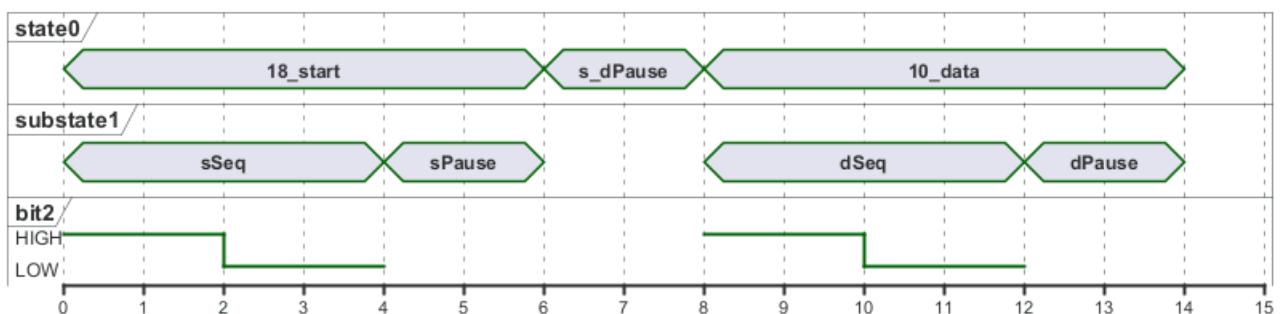
```
concise state0
concise substate1
robust bit2
```

```
bit2 has HIGH,LOW
```

```
@state0
0 is 18_start
6 is s_dPause
8 is 10_data
14 is {hidden}
```

```
@substate1
0 is sSeq
4 is sPause
6 is {hidden}
8 is dSeq
12 is dPause
14 is {hidden}
```

```
@bit2
0 is HIGH
2 is LOW
4 is {hidden}
8 is HIGH
10 is LOW
12 is {hidden}
@enduml
```



[Ref. QA-12222]



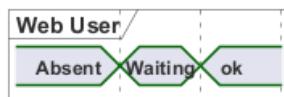
10.11 Hide time axis

It is possible to hide time axis.

```
@startuml
hide time-axis
concise "Web User" as WU
```

WU is Absent

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



10.12 Using Time and Date

It is possible to use time or date.

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

@2019/07/02

WU is Idle

WB is Idle

@2019/07/04

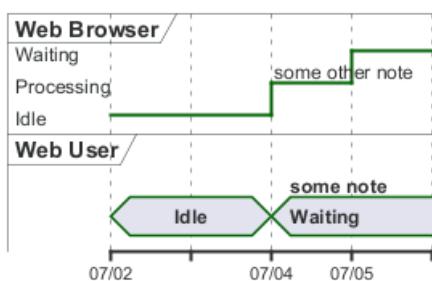
WU is Waiting : some note

WB is Processing : some other note

@2019/07/05

WB is Waiting

@enduml



@startuml

robust "Web Browser" as WB

concise "Web User" as WU

@1:15:00

WU is Idle

WB is Idle

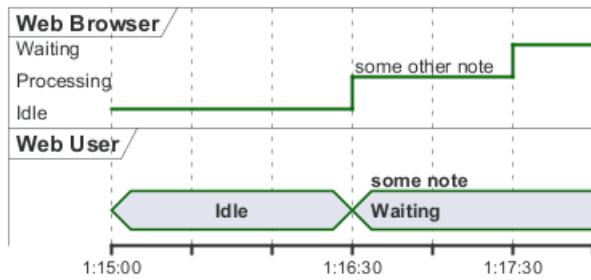
@1:16:30

WU is Waiting : some note

WB is Processing : some other note



@1:17:30
WB is Waiting
@enduml



[Ref. QA-7019]

10.13 Change Date Format

It is also possible to change date format.

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

use date format "YY-MM-dd"
```

@2019/07/02

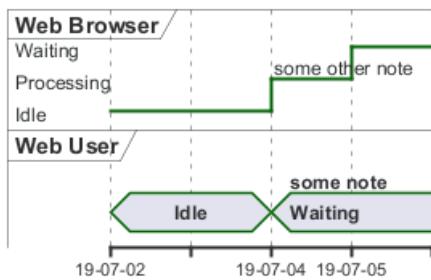
WU is Idle
WB is Idle

@2019/07/04

WU is Waiting : some note
WB is Processing : some other note

@2019/07/05

WB is Waiting
@enduml



10.14 Manage time axis labels

You can manage the time-axis labels.

10.14.1 Label on each tick (by default)

```
@startuml
scale 31536000 as 40 pixels
use date format "yy-MM"
concise "OpenGL Desktop" as OD
```



```
@1992/01/01
OD is {hidden}
```

```
@1992/06/30
OD is 1.0
```

```
@1997/03/04
OD is 1.1
```

```
@1998/03/16
OD is 1.2
```

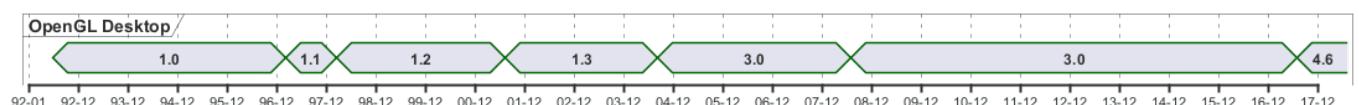
```
@2001/08/14
OD is 1.3
```

```
@2004/09/07
OD is 3.0
```

```
@2008/08/01
OD is 3.0
```

```
@2017/07/31
OD is 4.6
```

```
@enduml
```



10.14.2 Manual label (only when the state changes)

```
@startuml
scale 31536000 as 40 pixels
```

```
manual time-axis
use date format "yy-MM"
concise "OpenGL Desktop" as OD
```

```
@1992/01/01
OD is {hidden}
```

```
@1992/06/30
OD is 1.0
```

```
@1997/03/04
OD is 1.1
```

```
@1998/03/16
OD is 1.2
```

```
@2001/08/14
OD is 1.3
```

```
@2004/09/07
```



OD is 3.0

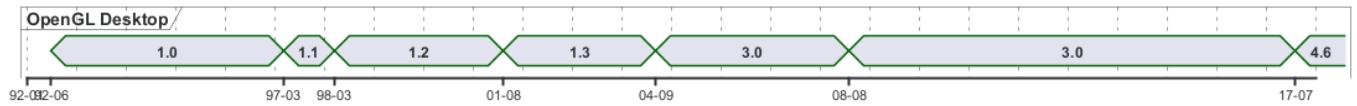
@2008/08/01

OD is 3.0

@2017/07/31

OD is 4.6

@enduml



[Ref. GH-1020]

10.15 Adding constraint

It is possible to display time constraints on the diagrams.

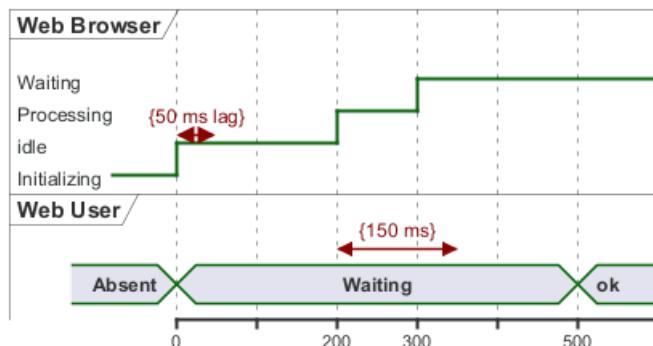
```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU
```

WB is Initializing

WU is Absent

```
@WB
0 is idle
+200 is Processing
+100 is Waiting
WB@0 <-> @50 : {50 ms lag}
```

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



10.16 Highlighted period

You can highlight a part of diagram.

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



```

@0
WU is Idle
WB is Idle

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

@200
WB is Proc.

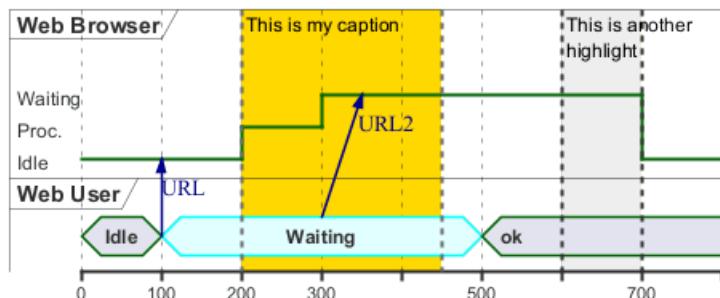
@300
WU -> WB@350 : URL2
WB is Waiting

@+200
WU is ok

@+200
WB is Idle

highlight 200 to 450 #Gold;line:DimGrey : This is my caption
highlight 600 to 700 : This is another\nhighlight
@enduml

```



[Ref. QA-10868]

10.17 Using notes

You can use the `note top of` and `note bottom of` keywords to define notes related to a single object or participant (*available only for concise or binary object*).

```

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

@0
WU is Idle
WB is Idle

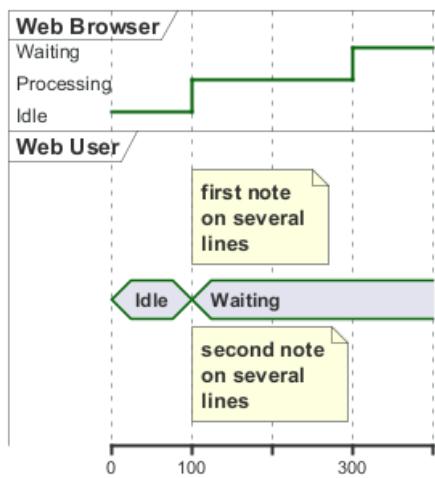
@100
WU is Waiting
WB is Processing
note top of WU : first note\nnon several\nlines
note bottom of WU : second note\nnon several\nlines

@300
WB is Waiting

```



```
@enduml
```



[Ref. QA-6877, GH-1465]

10.18 Adding texts

You can optionally add a title, a header, a footer, a legend and a caption:

```
@startuml
Title This is my title
header: some header
footer: some footer
legend
Some legend
end legend
caption some caption

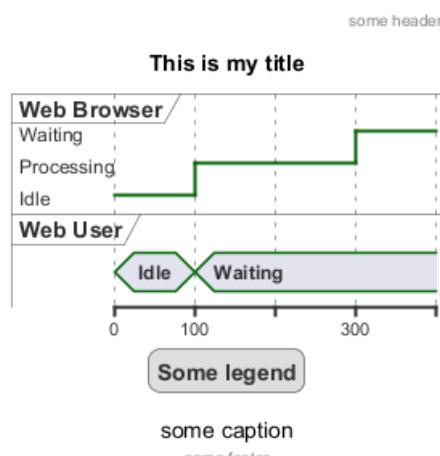
robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting
@enduml
```





10.19 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

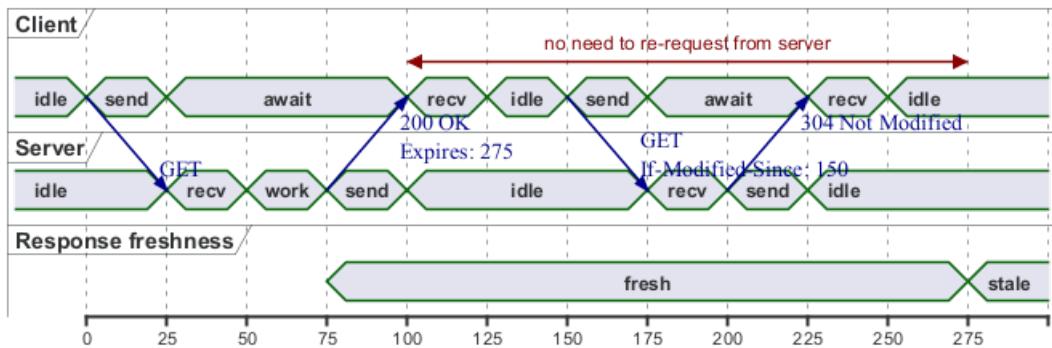
@Client
0 is send
Client -> Server@+25 : GET
+25 is await
+75 is recv
+25 is idle
+25 is send
Client -> Server@+25 : GET\nIf-Modified-Since: 150
+25 is await
+50 is recv
+25 is idle
@100 <-> @275 : no need to re-request from server

@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

```





10.20 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

```



```

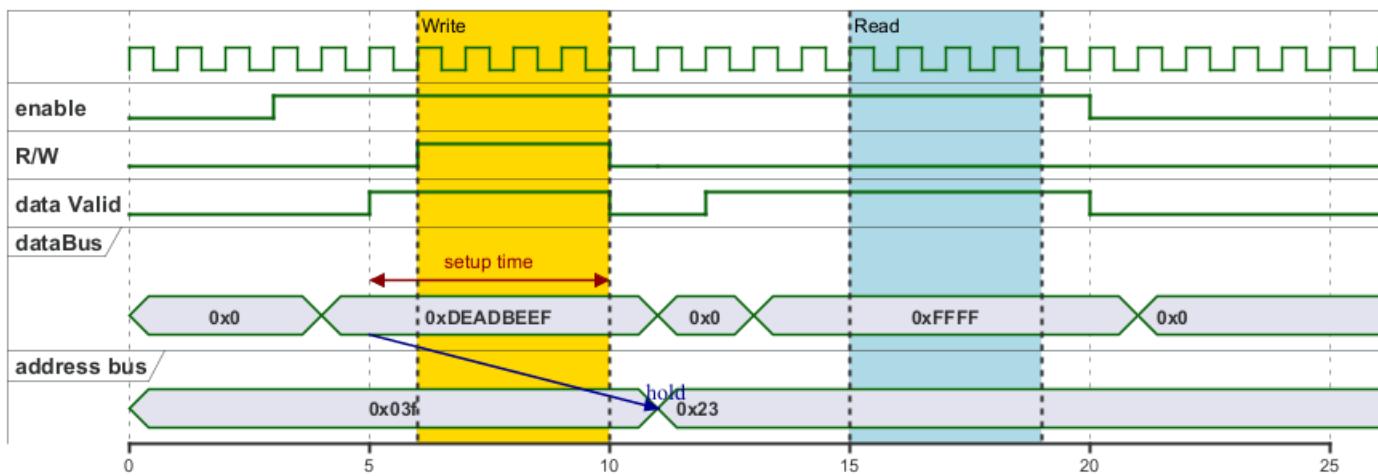
@13
db is "0xFFFF"

@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

```



10.21 Adding color

You can add color.

```

@startuml
concise "LR" as LR
concise "ST" as ST

```

```

LR is AtPlace #palegreen
ST is AtLoad #gray

```

```

@LR
0 is Lowering
100 is Lowered #pink
350 is Releasing

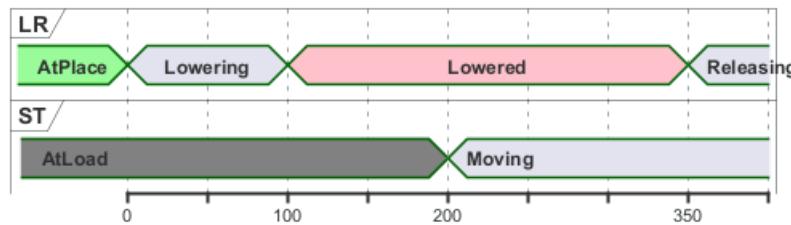
```

```

@ST
200 is Moving
@enduml

```





[Ref. QA-5776]

10.22 Using (global) style

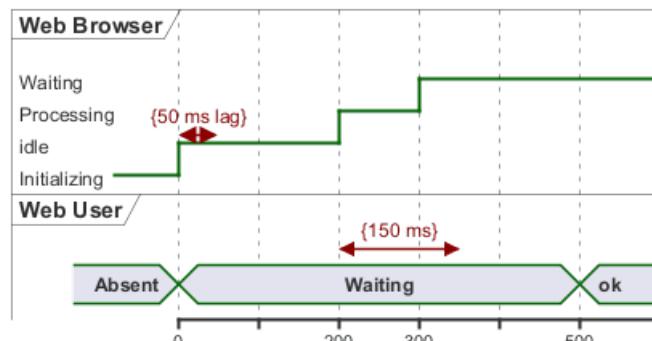
10.22.1 Without style (by default)

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

WB is Initializing
WU is Absent

```
@WB
0 is idle
+200 is Processing
+100 is Waiting
WB@0 <-> @50 : {50 ms lag}
```

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



10.22.2 With style

You can use style to change rendering of elements.

```
@startuml
<style>
timingDiagram {
    document {
        BackGroundColor SandyBrown
    }
    constraintArrow {
        LineStyle 2-1
        LineThickness 3
        LineColor Blue
    }
}
```



```

}
}
</style>
robust "Web Browser" as WB
concise "Web User" as WU

```

WB is Initializing
WU is Absent

```

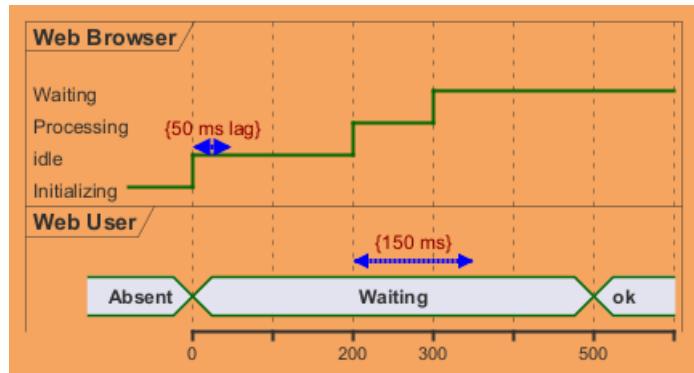
@WB
0 is idle
+200 is Processing
+100 is Waiting
WB00 <-> @50 : {50 ms lag}

```

```

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

```



[Ref. QA-14340]

10.23 Applying Colors to specific lines

You can use the `<style>` tags and stereotyping to give a name to line attributes.

```

@startuml
<style>
timingDiagram {
    .red {
        LineColor red
    }
    .blue {
        LineColor blue
        LineThickness 5
    }
}
</style>

clock clk with period 1
binary "Input Signal 1" as IS1
binary "Input Signal 2" as IS2 <<blue>>
binary "Output Signal 1" as OS1 <<red>>

```

```

@0
IS1 is low

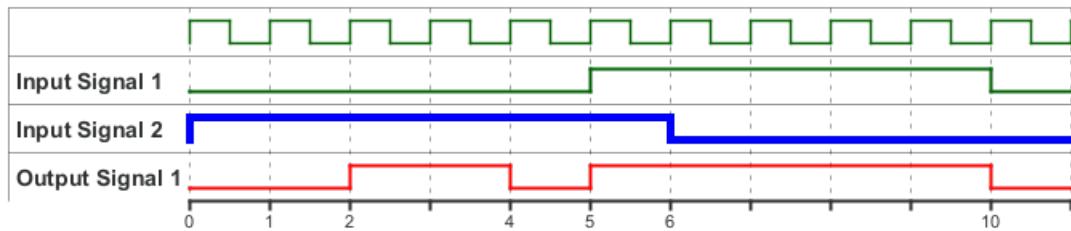
```



```

IS2 is high
OS1 is low
@2
OS1 is high
@4
OS1 is low
@5
IS1 is high
OS1 is high
@6
IS2 is low
@10
IS1 is low
OS1 is low
@enduml

```



[Ref. QA-15870]

10.24 Compact mode

You can use compact command to compact the timing layout.

10.24.1 By default

```

@startuml
robust "Web Browser" as WB
concise "Web User" as WU
robust "Web Browser2" as WB2

@0
WU is Waiting
WB is Idle
WB2 is Idle

@200
WB is Proc.

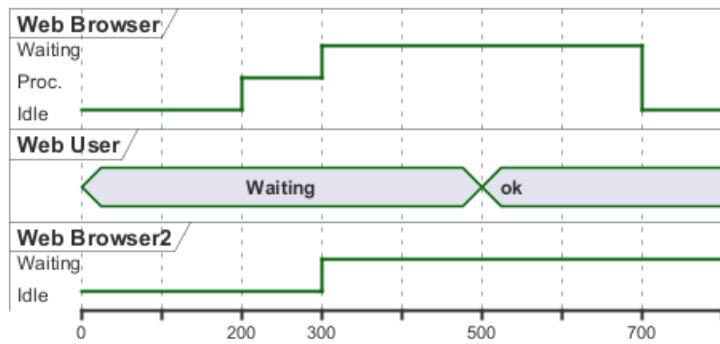
@300
WB is Waiting
WB2 is Waiting

@500
WU is ok

@700
WB is Idle
@enduml

```





10.24.2 Global mode with mode compact

```
@startuml
mode compact
robust "Web Browser" as WB
concise "Web User" as WU
robust "Web Browser2" as WB2
```

```
@0
WU is Waiting
WB is Idle
WB2 is Idle
```

```
@200
WB is Proc.
```

```
@300
WB is Waiting
WB2 is Waiting
```

```
@500
WU is ok
```

```
@700
WB is Idle
@enduml
```



10.24.3 Local mode with only compact on element

```
@startuml
compact robust "Web Browser" as WB
compact concise "Web User" as WU
robust "Web Browser2" as WB2
```

```
@0
WU is Waiting
WB is Idle
```



WB2 is Idle

@200

WB is Proc.

@300

WB is Waiting

WB2 is Waiting

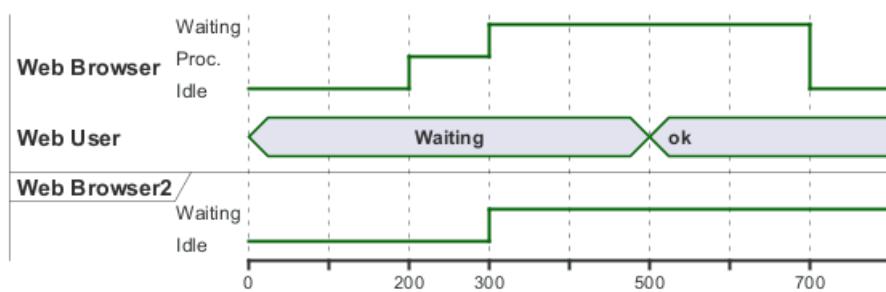
@500

WU is ok

@700

WB is Idle

@enduml



[Ref. QA-11130]

10.25 Scaling analog signal

You can scale analog signal.

10.25.1 Without scaling: 0-max (by default)

@startuml

title Between 0-max (by default)

analog "Analog" as A

@0

A is 350

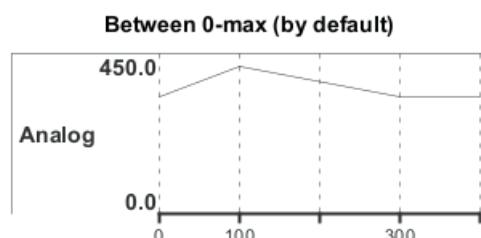
@100

A is 450

@300

A is 350

@enduml



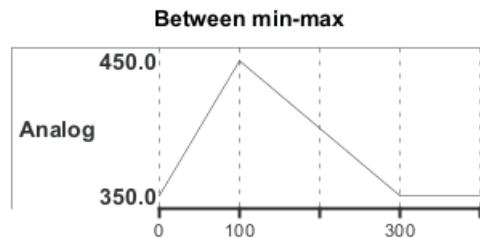
10.25.2 With scaling: min-max

```
@startuml
title Between min-max
analog "Analog" between 350 and 450 as A

@0
A is 350

@100
A is 450

@300
A is 350
@enduml
```



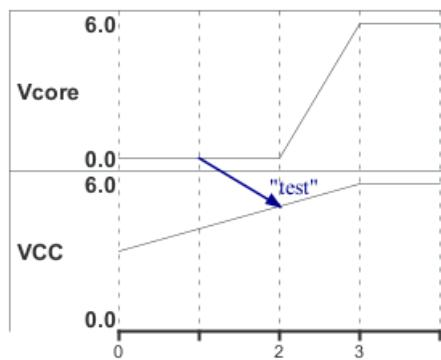
[Ref. QA-17161]

10.26 Customise analog signal

10.26.1 Without any customisation (by default)

```
@startuml
analog "Vcore" as VDD
analog "VCC" as VCC

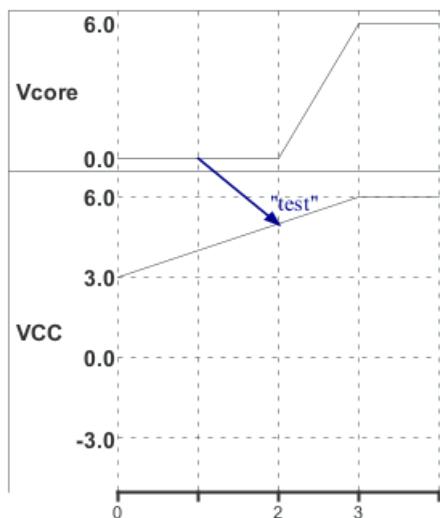
@0
VDD is 0
VCC is 3
@2
VDD is 0
@3
VDD is 6
VCC is 6
VDD@1 -> VCC@2 : "test"
@enduml
```



10.26.2 With customisation (on scale, ticks and height)

```
@startuml
analog "Vcore" as VDD
analog "VCC" between -4.5 and 6.5 as VCC
VCC ticks num on multiple 3
VCC is 200 pixels height

@0
VDD is 0
VCC is 3
@2
VDD is 0
@3
VDD is 6
VCC is 6
VDD@1 -> VCC@2 : "test"
@enduml
```



[Ref. QA-II288]

10.27 Order state of robust signal

10.27.1 Without order (by default)

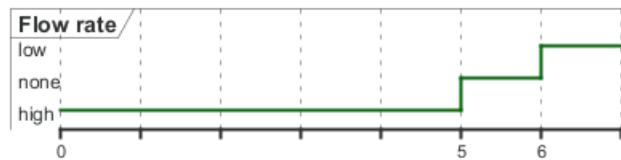
```
@startuml
robust "Flow rate" as rate

@0
rate is high

@5
rate is none

@6
rate is low
@enduml
```





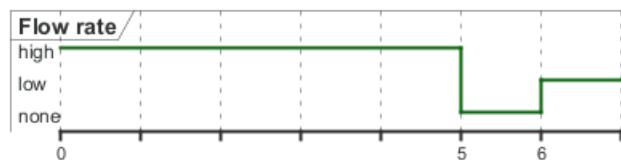
10.27.2 With order

```
@startuml
robust "Flow rate" as rate
rate has high,low,none
```

```
@0
rate is high
```

```
@5
rate is none
```

```
@6
rate is low
@enduml
```



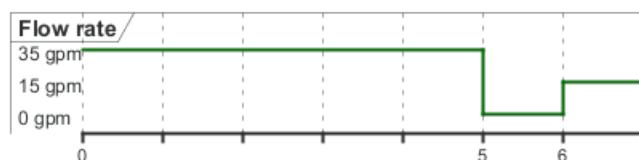
10.27.3 With order and label

```
@startuml
robust "Flow rate" as rate
rate has "35 gpm" as high
rate has "15 gpm" as low
rate has "0 gpm" as none
```

```
@0
rate is high
```

```
@5
rate is none
```

```
@6
rate is low
@enduml
```



[Ref. QA-6651]



10.28 Defining a timing diagram

10.28.1 By Clock (@clk)

```
@startuml
clock "clk" as clk with period 50
concise "Signal1" as S1
robust "Signal2" as S2
binary "Signal3" as S3

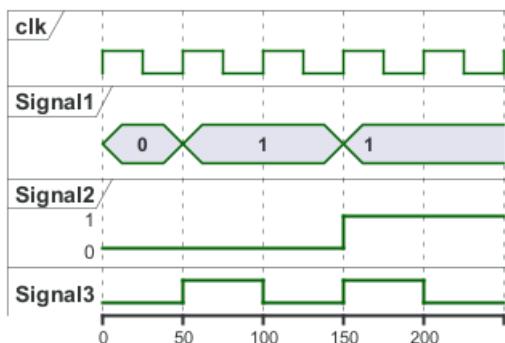
@clk*0
S1 is 0
S2 is 0

@clk*1
S1 is 1
S3 is high

@clk*2
S3 is down

@clk*3
S1 is 1
S2 is 1
S3 is 1

@clk*4
S3 is down
@enduml
```



10.28.2 By Signal (@S)

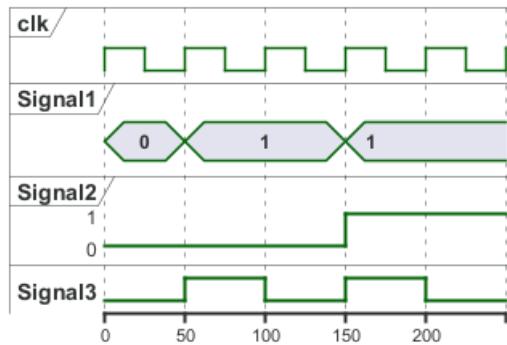
```
@startuml
clock "clk" as clk with period 50
concise "Signal1" as S1
robust "Signal2" as S2
binary "Signal3" as S3

@S1
0 is 0
50 is 1
150 is 1

@S2
0 is 0
150 is 1
```



```
@S3
50 is 1
100 is low
150 is high
200 is 0
@enduml
```



10.28.3 By Time (@time)

```
@startuml
clock "clk" as clk with period 50
concise "Signal1" as S1
robust "Signal2" as S2
binary "Signal3" as S3
```

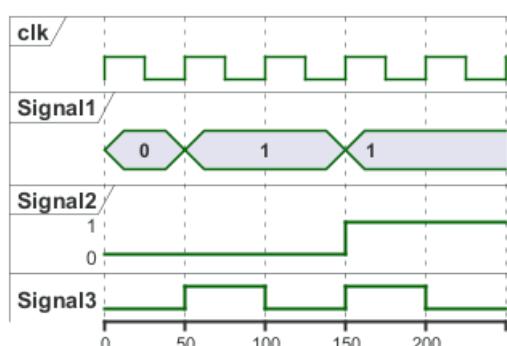
```
@0
S1 is 0
S2 is 0
```

```
@50
S1 is 1
S3 is 1
```

```
@100
S3 is low
```

```
@150
S1 is 1
S2 is 1
S3 is high
```

```
@200
S3 is 0
@enduml
```



[Ref. QA-9053]

10.29 Annotate signal with comment

```
@startuml
binary "Binary Serial Data" as D
robust "Robust" as R
concise "Concise" as C
```

```
@-3
D is low: idle
R is lo: idle
C is 1: idle
@-1
D is high: start
R is hi: start
C is 0: start
```

```
@0
D is low: 1 lsb
R is lo: 1 lsb
C is 1: lsb
```

```
@1
D is high: 0
R is hi: 0
C is 0
```

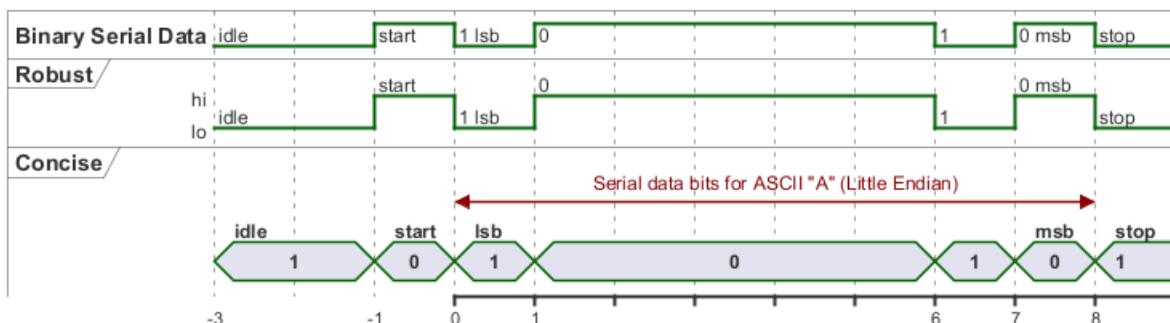
```
@6
D is low: 1
R is lo: 1
C is 1
```

```
@7
D is high: 0 msb
R is hi: 0 msb
C is 0: msb
```

```
@8
D is low: stop
R is lo: stop
C is 1: stop
```

00 <-> 08 : Serial data bits for ASCII "A" (LittleEndian)

@enduml



[Ref. QA-15762, and QH-888]



11 Display JSON Data

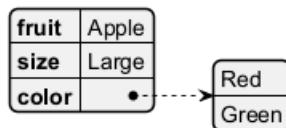
JSON широко применяется в программном обеспечении.

Вы можете использовать PlantUML для визуализации ваших данных.

Чтобы активировать эту возможность, диаграмма должна:

- начинаться с ключевого слова `@startjson`
- заканчиваться ключевым словом `@endjson`.

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



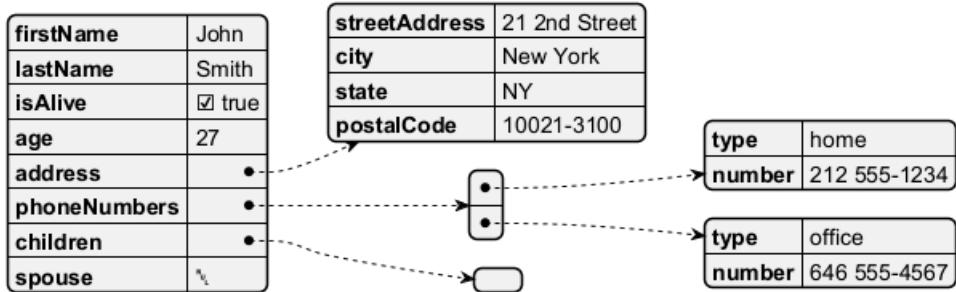
□ *If you are looking for how to manipulate and manage JSON data on PlantUML: see rather [Preprocessing JSON](preprocessing-json).*

11.1 Complex example

You can use complex JSON structure.

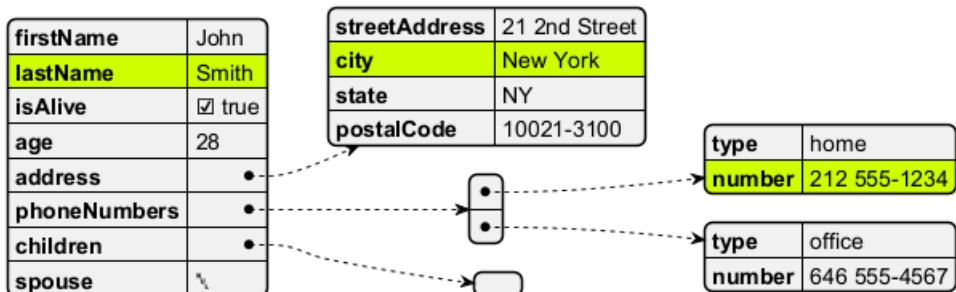
```
@startjson
{
    "firstName": "John",
    "lastName": "Smith",
    "isAlive": true,
    "age": 27,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021-3100"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "office",
            "number": "646 555-4567"
        }
    ],
    "children": [],
    "spouse": null
}
@endjson
```





11.2 Highlight parts

```
@startjson
#highlight "lastName"
#highlight "address" / "city"
#highlight "phoneNumbers" / "0" / "number"
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 28,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
@endjson
```



11.3 Using different styles for highlight

It is possible to have different styles for different highlights.

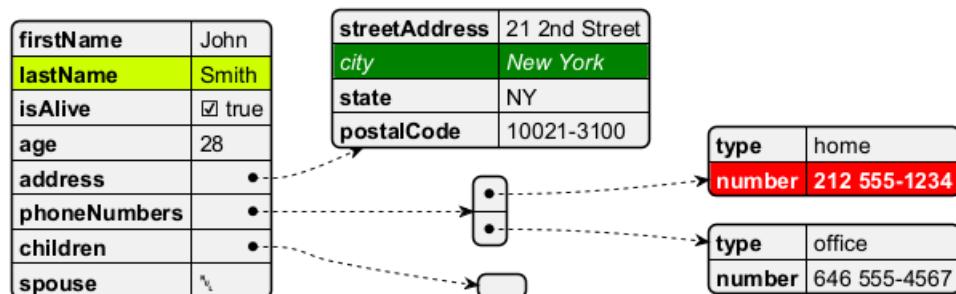
```
@startjson
```



```

<style>
.h1 {
    BackGroundColor green
    FontColor white
    FontStyle italic
}
.h2 {
    BackGroundColor red
    FontColor white
    FontStyle bold
}
</style>
#highlight "lastName"
#highlight "address" / "city" <><h1>>
#highlight "phoneNumbers" / "0" / "number" <><h2>>
{
    "firstName": "John",
    "lastName": "Smith",
    "isAlive": true,
    "age": 28,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021-3100"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "office",
            "number": "646 555-4567"
        }
    ],
    "children": [],
    "spouse": null
}
@endjson

```



[Ref. QA-15756, GH-1393]

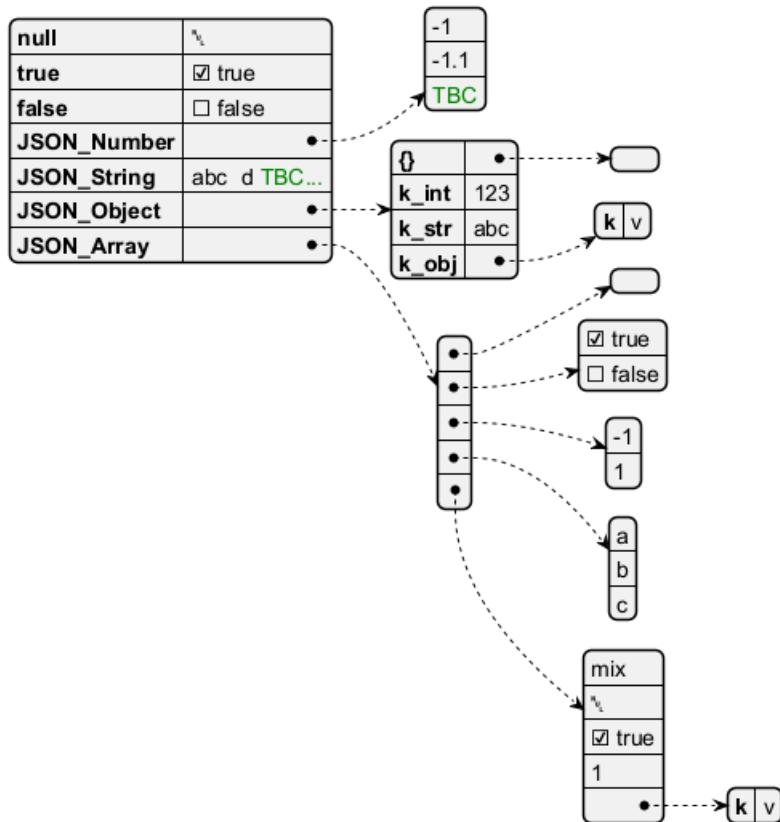
11.4 JSON basic element

11.4.1 Synthesis of all JSON basic element

@startjson



```
{
  "null": null,
  "true": true,
  "false": false,
  "JSON_Number": [-1, -1.1, "<color:green>TBC"],
  "JSON_String": "a\nb\rc\td <color:green>TBC...",
  "JSON_Object": {
    "{}": {},
    "k_int": 123,
    "k_str": "abc",
    "k_obj": {"k": "v"}
  },
  "JSON_Array" : [
    [],
    [true, false],
    [-1, 1],
    ["a", "b", "c"],
    ["mix", null, true, 1, {"k": "v"}]
  ]
}
}
@endjson
```



11.5 JSON array or table

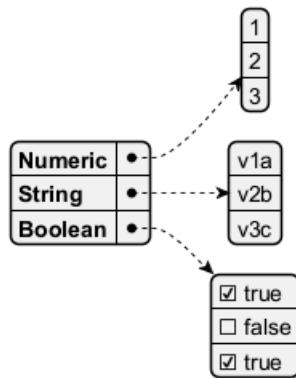
11.5.1 Array type

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



```
}
```

```
@endjson
```



11.5.2 Minimal array or table

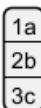
11.5.3 Number array

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



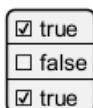
11.5.4 String array

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



11.5.5 Boolean array

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

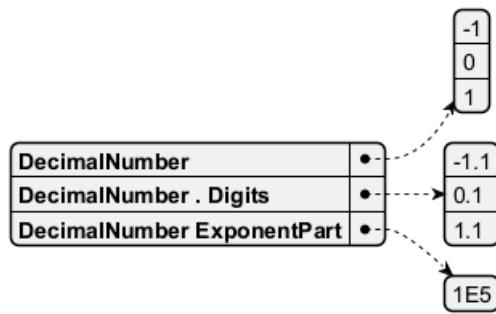


11.6 JSON numbers

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



```
@endjson
```



11.7 JSON strings

11.7.1 JSON Unicode

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

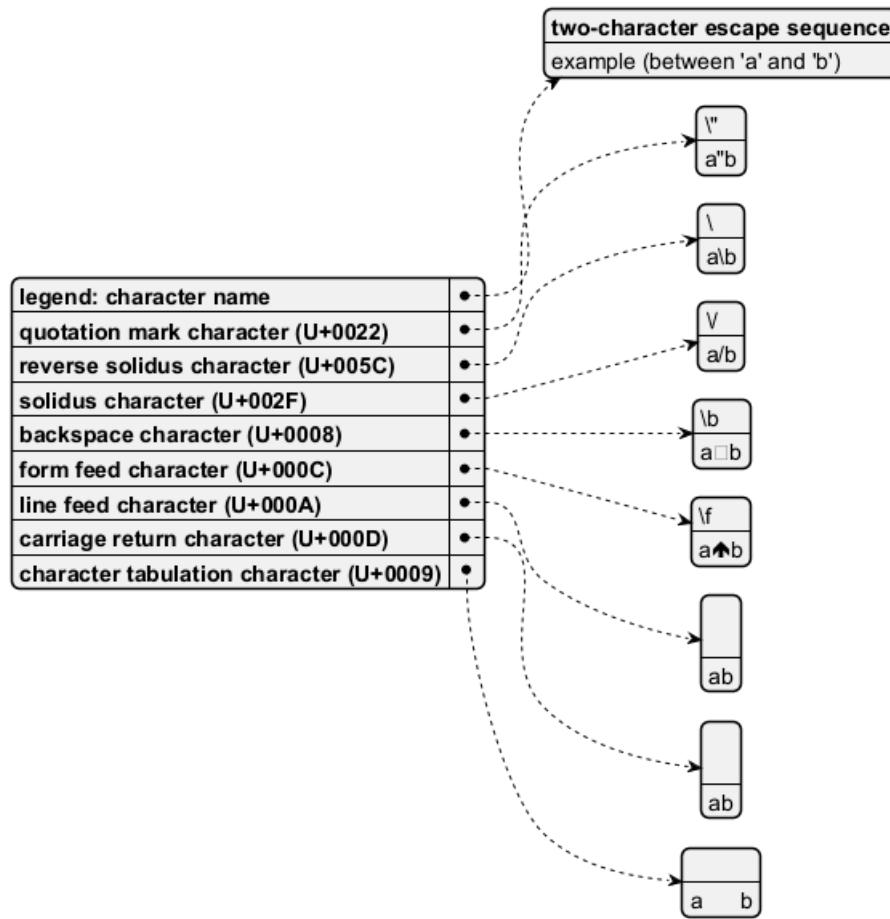
```
@startjson
{
  "<color:blue><b>code": "<color:blue><b>value",
  "a\\u005Cb": "a\u005Cb",
  "\\uD83D\\uDE10": "\uD83D\uDE10",
  " ":
}
@endjson
```

code	value
a\u005Cb	a\b
\uD83D\uDE10	(?)
(?)	(?)

11.7.2 JSON two-character escape sequence

```
@startjson
{
  "**legend**: character name": [
    "quotation mark character (U+0022)": ["\"\"", "a\"b"],
    "reverse solidus character (U+005C)": ["\\\\\\", "a\\b"],
    "solidus character (U+002F)": ["\\\\/", "a/b"],
    "backspace character (U+0008)": ["\\b", "a\bb"],
    "form feed character (U+000C)": ["\\f", "a\fb"],
    "line feed character (U+000A)": ["\\n", "a\nb"],
    "carriage return character (U+000D)": ["\\r", "a\rb"],
    "character tabulation character (U+0009)": ["\\t", "a\tb"]
  ]
}
@endjson
```





TODO: FIXME FIXME or not ☐, on the same item as \n management in PlantUML ☐ See Report Bug on QA-13066 **TODO:** FIXME

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



11.8 Minimal JSON examples

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

Hello world!



```
@startjson
42
@endjson
```

```
@startjson
true
@endjson
```

(Examples come from STD 90 - Examples)

11.9 Empty table or list

```
@startjson
{
  "empty_tab": [],
  "empty_list": []
}
@endjson
```

empty_tab	
empty_list	

[Ref. QA-14397]

11.10 Using (global) style

11.10.1 Without style (by default)

```
@startjson
#highlight "1" / "hr"
[
  {
    "name": "Mark McGwire",
    "hr": 65,
    "avg": 0.278
  },
  {
    "name": "Sammy Sosa",
    "hr": 63,
    "avg": 0.288
  }
]
@endjson
```

name	Mark McGwire
hr	65
avg	0.278

name	Sammy Sosa
hr	63
avg	0.288

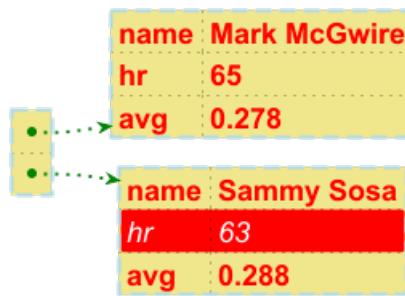


11.10.2 With style

You can use style to change rendering of elements.

```
@startjson
<style>
jsonDiagram {
    node {
        BackGroundColor Khaki
        LineColor lightblue
        FontName Helvetica
        FontColor red
        FontSize 18
        FontStyle bold
        RoundCorner 0
        LineThickness 2
        LineStyle 10-5
        separator {
            LineThickness 0.5
            LineColor black
            LineStyle 1-5
        }
    }
    arrow {
        BackGroundColor lightblue
        LineColor green
        LineThickness 2
        LineStyle 2-5
    }
    highlight {
        BackGroundColor red
        FontColor white
        FontStyle italic
    }
}
</style>
#highlight "1" / "hr"
[
{
    "name": "Mark McGwire",
    "hr": 65,
    "avg": 0.278
},
{
    "name": "Sammy Sosa",
    "hr": 63,
    "avg": 0.288
}
]
@endjson
```



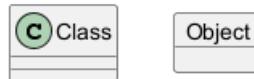


[Adapted from QA-13123 and QA-13288]

11.11 Display JSON Data on Class or Object diagram

11.11.1 Simple example

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



JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

11.11.2 Complex example: with all JSON basic element

```
@startuml
json "<b>JSON basic element</b>" as J {
"null": null,
"true": true,
"false": false,
"JSON_Number": [-1, -1.1, "<color:green>TBC"],
"JSON_String": "a\nb\rc\td <color:green>TBC...",
"JSON_Object": {
    "{}": {},
    "k_int": 123,
    "k_str": "abc",
    "k_obj": {"k": "v"}
},
"JSON_Array" : [
    [],
    [true, false],
    [-1, 1],
    ...
]
```



```

["a", "b", "c"],
["mix", null, true, 1, {"k": "v"}]
]
}
@enduml

```

JSON basic element	
null	null
true	true
false	false
JSON_Number	-1 -1.1 TBC
JSON_String	abc d TBC...
JSON_Object	{ k_int 123 k_str abc k_obj k v}
JSON_Array	true false -1 1 a b c mix null true 1 k v

11.12 Display JSON Data on Deployment (Usecase, Component, Deployment) diagram

11.12.1 Simple example

```

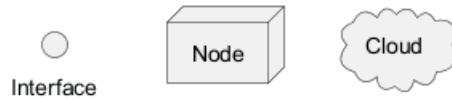
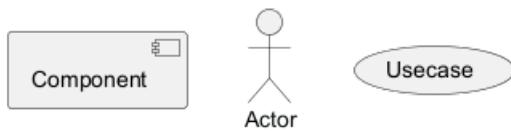
@startuml
allowmixing

component Component
actor Actor
usecase Usecase
() Interface
node Node
cloud Cloud

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

```





JSON	
fruit	Apple
size	Large
color	Red
	Green

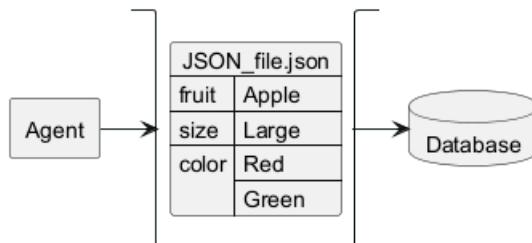
[Ref. QA-15481]

Complex example: with arrow

```
@startuml
allowmixing

agent Agent
stack {
    json "JSON_file.json" as J {
        "fruit":"Apple",
        "size":"Large",
        "color": ["Red", "Green"]
    }
}
database Database

Agent -> J
J -> Database
@enduml
```



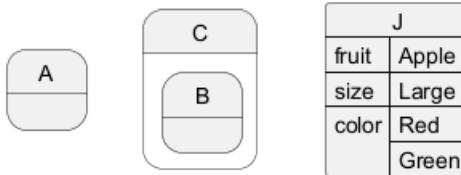
11.13 Display JSON Data on State diagram

11.13.1 Simple example

```
@startuml
state "A" as stateA
state "C" as stateC {
    state B
}
```



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



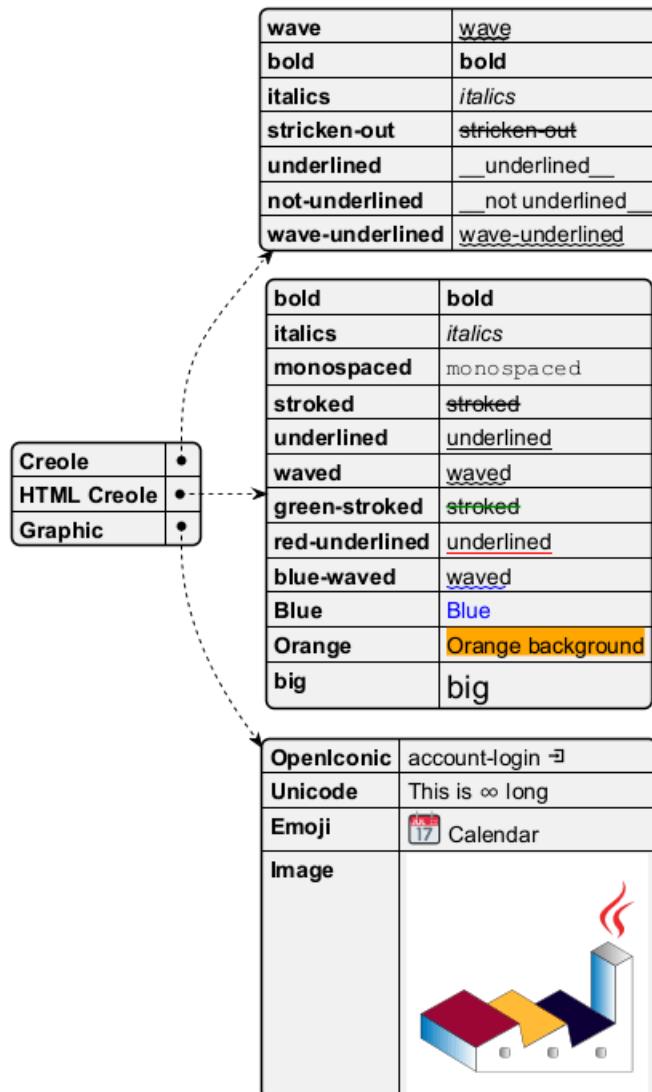
[Ref. QA-17275]

11.14 Creole on JSON

You can use Creole or HTML Creole on JSON diagram:

```
@startjson
{
  "Creole": {
    "wave": "~~wave~~",
    "bold": "**bold**",
    "italics": "//italics//",
    "stricken-out": "--stricken-out--",
    "underlined": "__underlined__",
    "not-underlined": "~__not underlined__",
    "wave-underlined": "~~wave-underlined~~"
  },
  "HTML Creole": {
    "bold": "<b>bold",
    "italics": "<i>italics",
    "monospaced": "<font:monospaced>monospaced",
    "stroked": "<s>stroked",
    "underlined": "<u>underlined",
    "waved": "<w>waved",
    "green-stroked": "<s:green>stroked",
    "red-underlined": "<u:red>underlined",
    "blue-waved": "<w:#0000FF>waved",
    "Blue": "<color:blue>Blue",
    "Orange": "<back:orange>Orange background",
    "big": "<size:20>big"
  },
  "Graphic": {
    "OpenIconic": "account-login &account-login",
    "Unicode": "This is <U+221E> long",
    "Emoji": "<:calendar:> Calendar",
    "Image": "<img:https://plantuml.com/logo3.png>"
  }
}
@endjson
```





12 Display YAML Data

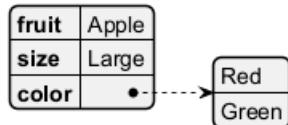
YAML format is widely used in software.

You can use PlantUML to visualize your data.

To activate this feature, the diagram must:

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

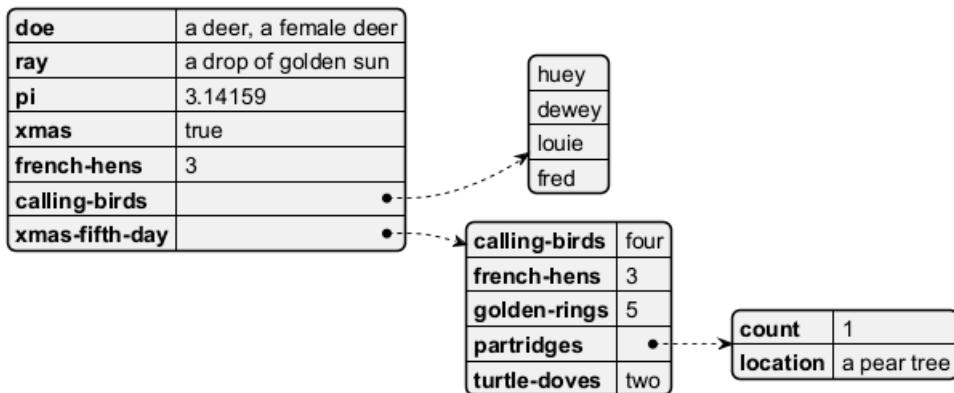
```
@startyaml
fruit: Apple
size: Large
color:
  - Red
  - Green
@endyaml
```



12.1 Complex example

```
@startyaml
doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
  - huey
  - dewey
  - louie
  - fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```





12.2 Specific key (with symbols or unicode)

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

@fruit	Apple
\$size	Large
&color	Red
♥	Heart
%	Per mille

[Ref. QA-13376]

12.3 Highlight parts

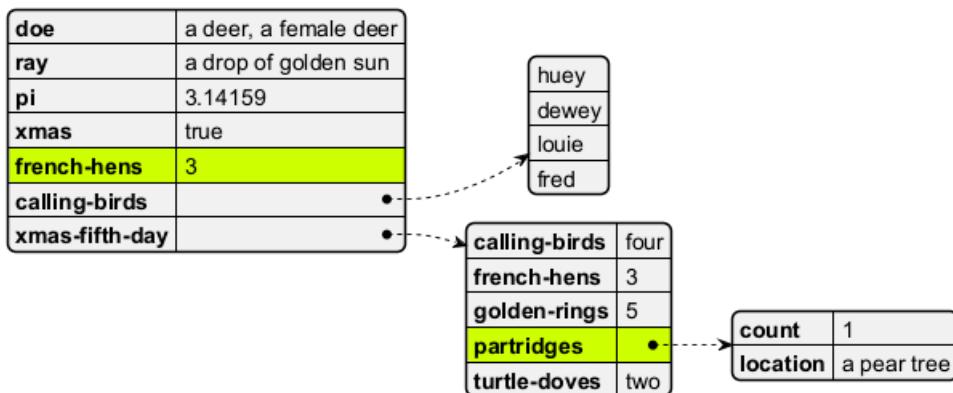
12.3.1 Normal style

```
@startyaml
#highlight "french-hens"
#highlight "xmas-fifth-day" / "partridges"
```

```
doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
```



@endyaml

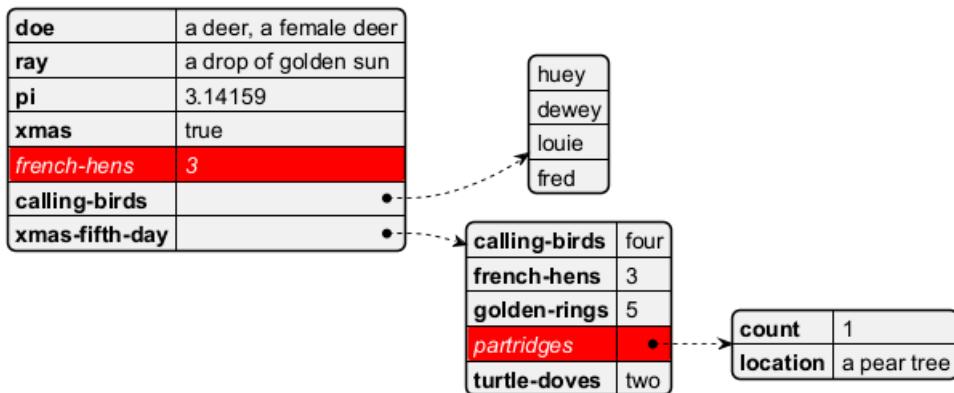


12.3.2 Customised style

```
@startyaml
<style>
yamlDiagram {
    highlight {
        BackGroundColor red
        FontColor white
        FontStyle italic
    }
}
</style>
#highlight "french-hens"
#highlight "xmas-fifth-day" / "partridges"

doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```





[Ref. QA-13288]

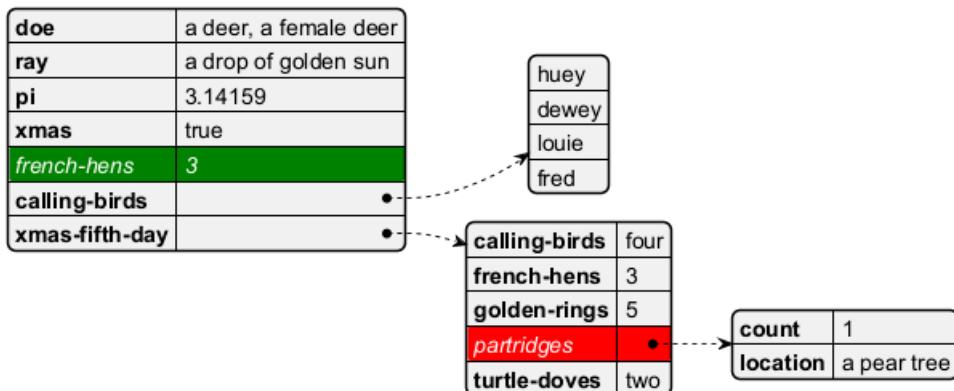
12.4 Using different styles for highlight

It is possible to have different styles for different highlights.

```
@startyaml
<style>
    .h1 {
        BackGroundColor green
        FontColor white
        FontStyle italic
    }
    .h2 {
        BackGroundColor red
        FontColor white
        FontStyle italic
    }
</style>
#highlight "french-hens" <<h1>>
#highlight "xmas-fifth-day" / "partridges" <<h2>>

doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```





[Ref. QA-15756, GH-1393]

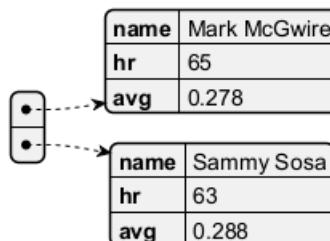
12.5 Using (global) style

12.5.1 Without style (by default)

```
@startyaml
```

```
-  
  name: Mark McGwire  
  hr: 65  
  avg: 0.278  
  
-  
  name: Sammy Sosa  
  hr: 63  
  avg: 0.288
```

```
@endyaml
```



12.5.2 With style

You can use style to change rendering of elements.

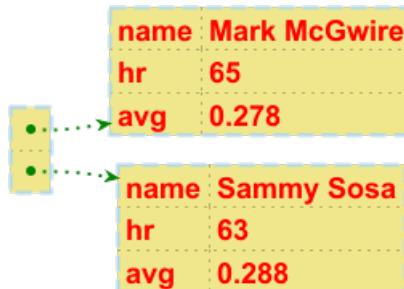
```
@startyaml
<style>
yamlDiagram {
  node {
    BackGroundColor lightblue
    LineColor lightblue
    FontName Helvetica
    FontColor red
    FontSize 18
    FontStyle bold
    BackGroundColor Khaki
    RoundCorner 0
    LineThickness 2
    LineStyle 10-5
    separator {
      LineThickness 0.5
    }
  }
}
```



```

    LineColor black
    LineStyle 1-5
}
}
arrow {
    BackGroundColor lightblue
    LineColor green
    LineThickness 2
    LineStyle 2-5
}
}
</style>
-
  name: Mark McGwire
  hr: 65
  avg: 0.278
-
  name: Sammy Sosa
  hr: 63
  avg: 0.288
@endyaml

```



[Ref. QA-13123]

12.6 Creole on YAML

You can use Creole or HTML Creole on YAML diagram:

```

@startyaml
Creole:
  wave: ~~wave~~
  bold: **bold**
  italics: //italics//"
  monospaced: ""monospaced"""
  stricken-out: --stricken-out--
  underlined: __underlined__
  not-underlined: ~__not underlined__
  wave-underlined: ~~wave-underlined~~
HTML Creole:
  bold: <b>bold
  italics: <i>italics
  monospaced: <font:monospaced>monospaced
  stroked: <s>stroked
  underlined: <u>underlined
  waved: <w>waved
  green-stroked: <s:green>stroked
  red-underlined: <u:red>underlined
  blue-waved: <w:#0000FF>waved
  Blue: <color:blue>Blue

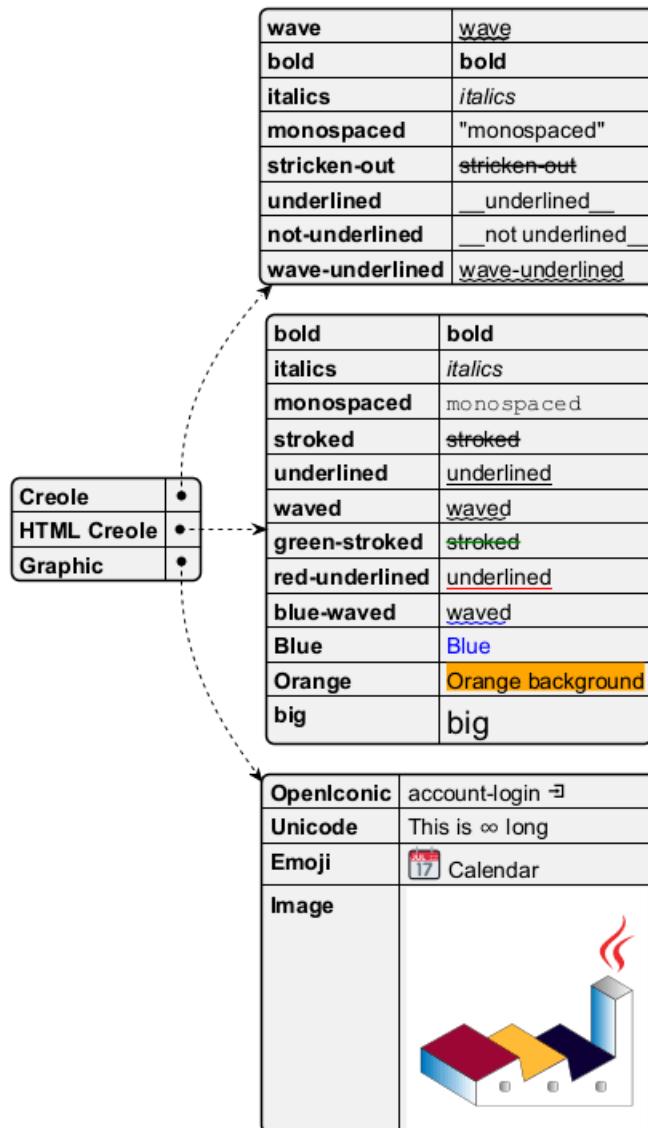
```



```

Orange: <back:orange>Orange background
big: <size:20>big
Graphic:
  OpenIconic: account-login <&account-login>
  Unicode: This is <U+221E> long
  Emoji: <:calendar:> Calendar
  Image: <img:https://plantuml.com/logo3.png>
@endyaml

```



13 Network Diagram with nwdiag

A network diagram is a visual representation of a computer or telecommunications network. It illustrates the **arrangement and interconnections** of network components, including servers, routers, switches, hubs, and devices. Network diagrams are invaluable tools for network engineers and administrators to **understand, set up, and troubleshoot networks**. They are also essential for **visualizing the structure and flow of data** in a network, ensuring optimal performance and security.

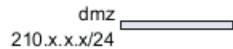
nwdiag, developed by Takeshi Komiya, provides a streamlined platform to swiftly sketch **network diagrams**. We extend our gratitude to Takeshi for this **innovative tool!**

Given its intuitive syntax, nwdiag has been seamlessly integrated into **PlantUML**. The examples showcased here are inspired by the ones documented by Takeshi.

13.1 Простая диаграмма

13.1.1 Определите сеть

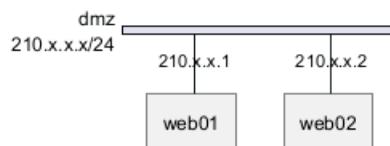
```
@startuml
nwdiag {
    network dmz {
        address = "210.x.x.x/24"
    }
}
@enduml
```



13.1.2 Определение некоторых элементов или серверов в сети

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

        web01 [address = "210.x.x.1"];
        web02 [address = "210.x.x.2"];
    }
}
@enduml
```



13.1.3 Полный пример

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

        web01 [address = "210.x.x.1"];
        web02 [address = "210.x.x.2"];
    }
}
```

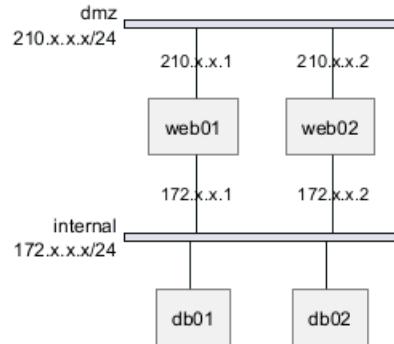


```

}
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

```

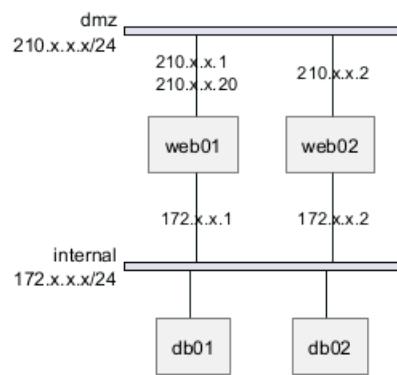
@startuml
nwdiag {
    network dmz {
        address = "210.x.x.x/24"

        // 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

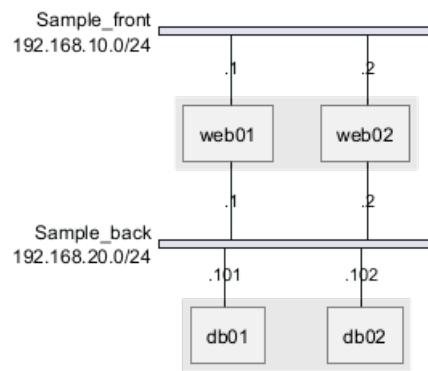
```
@startuml
nwdiag {
    network Sample_front {
        address = "192.168.10.0/24";

        // 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
```





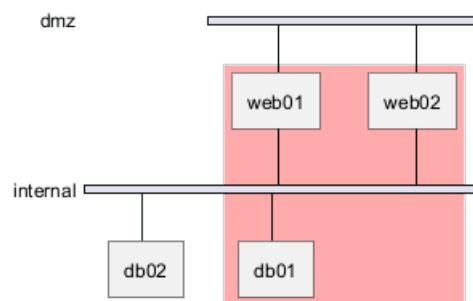
13.3.2 Define group outside of network definitions

```

@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 {
  
```



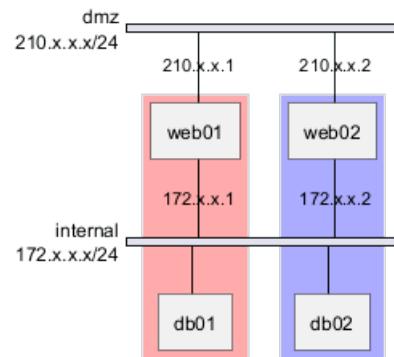
```

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";
    }
}

```

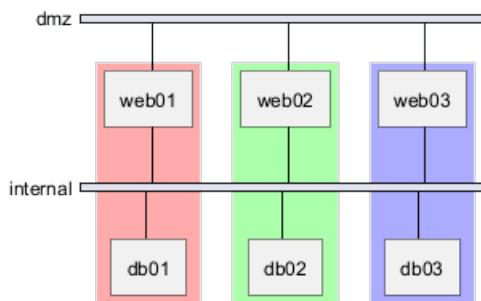


```

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"
    }
}

```



```

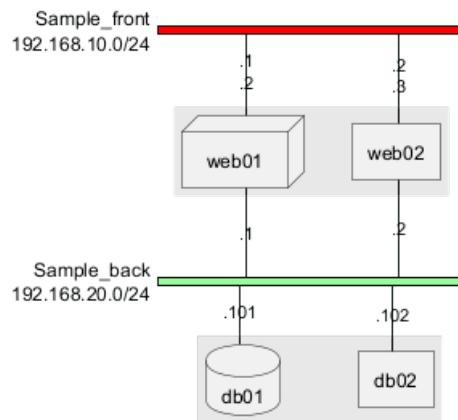
color = "palegreen"
web01 [address = ".1"]
web02 [address = ".2"]
db01 [address = ".101", shape = database ]
db02 [address = ".102"]

// 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";

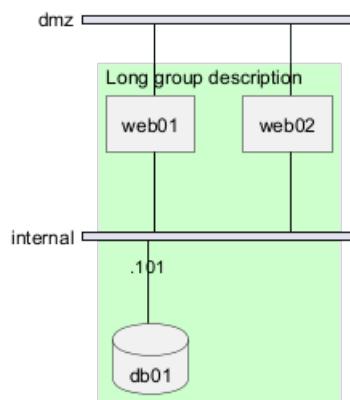
        web01;
        web02;
        db01;
    }

    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.

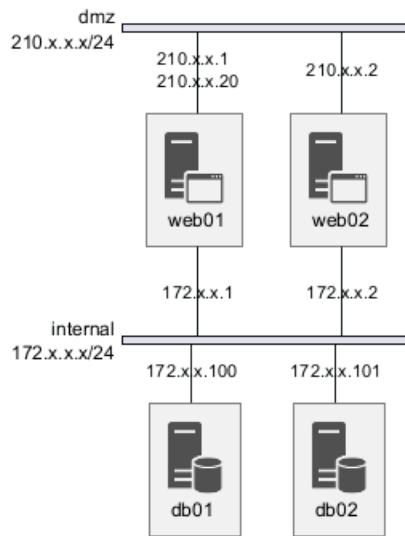
```
@startuml
!include <office/Servers/application_server>
!include <office/Servers/database_server>

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 \n to make a newline:

```
@startuml
```

```

nwdiag {
    group nightly {
        color = "#FFAAAA";
        description = "<&clock> Restarted nightly <&clock>";
        web02;
        db01;
    }
    network dmz {
        address = "210.x.x.x/24"

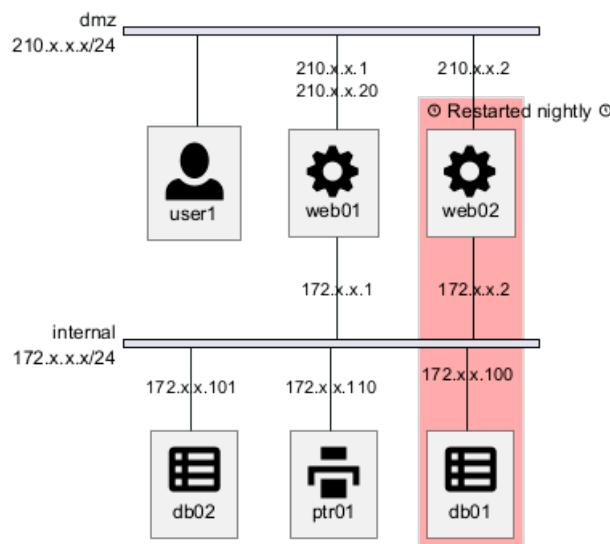
        user [description = "<&person*4.5>\n user1"];
        // set multiple addresses (using comma)
        web01 [address = "210.x.x.1, 210.x.x.20", description = "<&cog*4>\nweb01"]
        web02 [address = "210.x.x.2", description = "<&cog*4>\nweb02"];

    }
    network internal {
        address = "172.x.x.x/24";

        web01 [address = "172.x.x.1"];
        web02 [address = "172.x.x.2"];
        db01 [address = "172.x.x.100", description = "<&spreadsheet*4>\n db01"];
        db02 [address = "172.x.x.101", description = "<&spreadsheet*4>\n db02"];
        ptr [address = "172.x.x.110", description = "<&print*4>\n ptr01"];
    }
}
@enduml

```





13.7 Same nodes on more than two networks

You can use same nodes on different networks (more than two networks); *nwdiag* use in this case '*jump line*' over networks.

```

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

        web01;
        web02;
        db01;
    }

    network dmz {
        color = "pink"

        web01;
        web02;
    }

    network internal {
        web01;
        web02;
        db01 [shape = database ];
    }

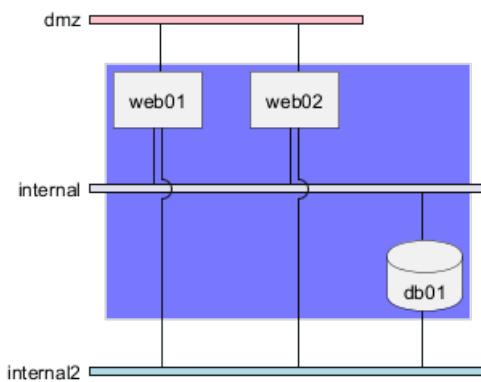
    network internal2 {
        color = "LightBlue";

        web01;
        web02;
        db01;
    }
}

```

}

```
@enduml
```

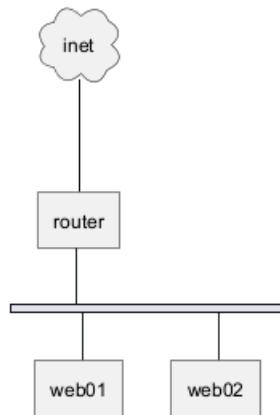


13.8 Peer networks

Peer networks are simple connections between two nodes, for which we don't use a horizontal "busbar" network

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

    network {
        router;
        web01;
        web02;
    }
}
@enduml
```



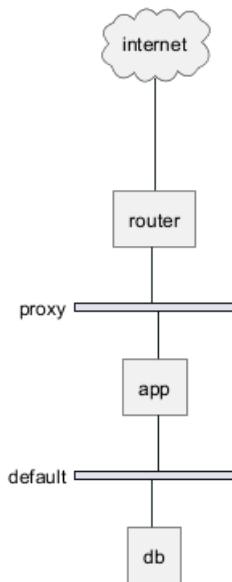
13.9 Peer networks and group

13.9.1 Without group

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



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



13.9.2 Group on first

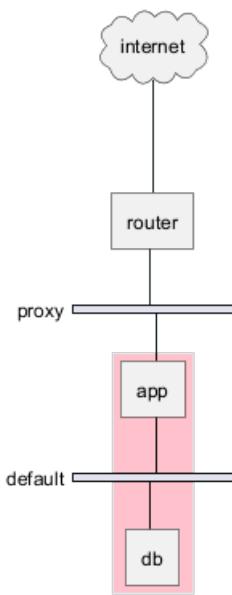
```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

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

    network proxy {
        router;
        app;
    }

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

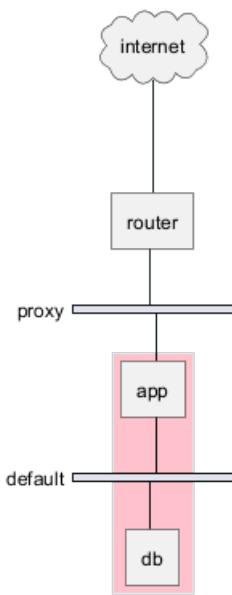




13.9.3 Group on second

```
@startuml  
nwdiag {  
    internet [ shape = cloud];  
    internet -- router;  
  
    network proxy {  
        router;  
        app;  
    }  
  
    group {  
        color = "pink";  
        app;  
        db;  
    }  
  
    network default {  
        app;  
        db;  
    }  
}  
@enduml
```





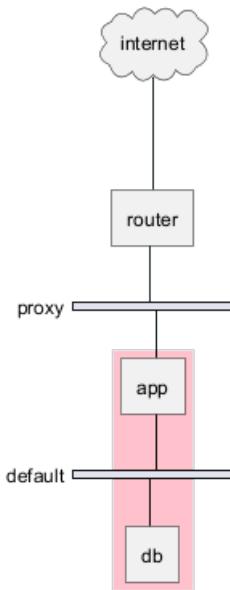
13.9.4 Group on third

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

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

The code defines a network diagram using the nwdiag notation. It starts with an "internet" node (cloud shape) connected to a "router" node. The "router" node is part of a "proxy" network, which also contains an "app" node. The "app" and "db" nodes are grouped together and assigned a pink color. The "group" block also contains the "app" and "db" nodes.





[Ref. Issue#408 and QA-12655]

13.10 Add title, caption, header, footer or legend on network diagram

```
@startuml
```

```
header some header
```

```
footer some footer
```

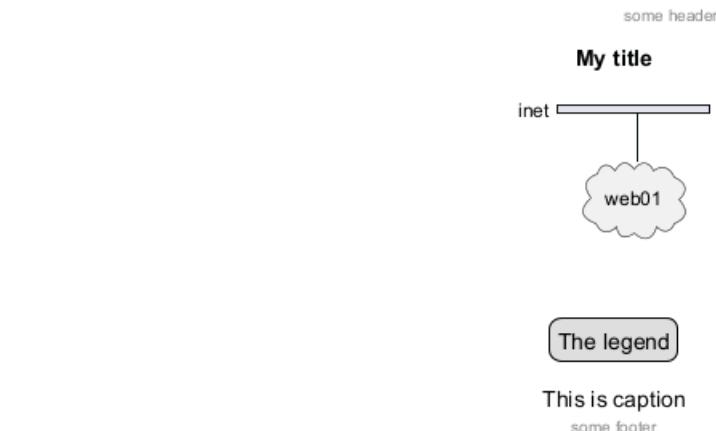
```
title My title
```

```
nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}
```

```
legend
The legend
end legend
```

```
caption This is caption
@enduml
```



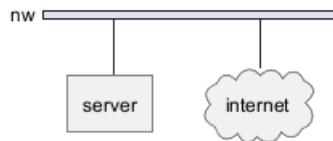


[Ref. QA-11303 and Common commands]

13.11 With or without shadow

13.11.1 With shadow (by default)

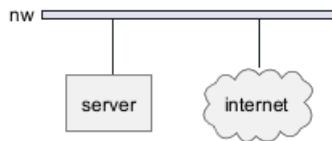
```
@startuml
nwdiag {
    network nw {
        server;
        internet;
    }
    internet [shape = cloud];
}
@enduml
```



13.11.2 Without shadow

```
@startuml
<style>
root {
    shadowing 0
}
</style>
nwdiag {
    network nw {
        server;
        internet;
    }
    internet [shape = cloud];
}
@enduml
```





[Ref. QA-14516]

13.12 Change width of the networks

You can change the width of the networks, especially in order to have the same full width for only some or all networks.

Here are some examples, with all the possibilities.

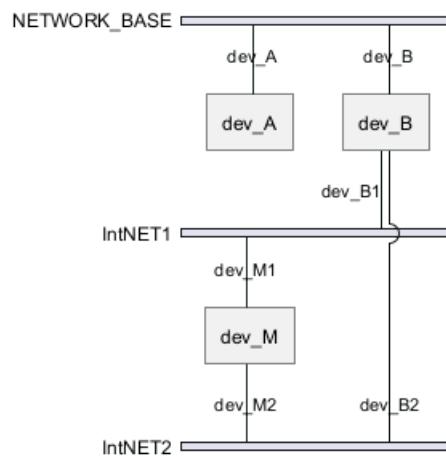
13.12.1 First example

- 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
    }
}

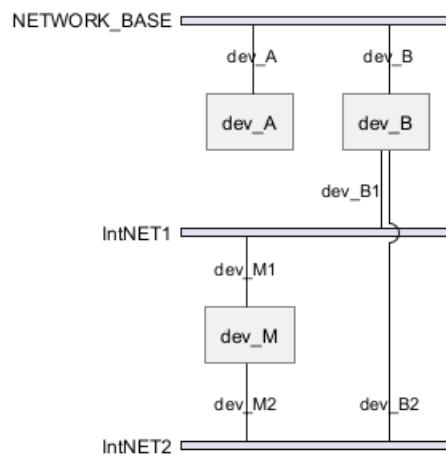
```



```

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

```



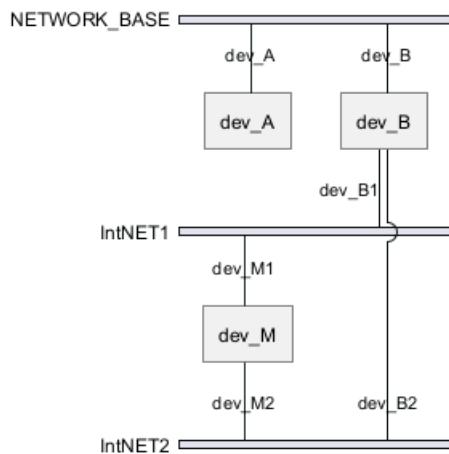
- the first and the second

```

@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

```



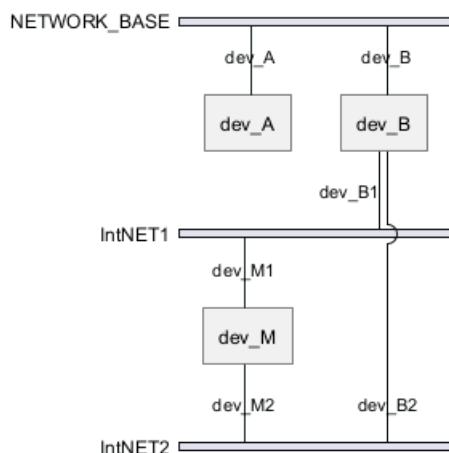


- all the network (with same full width)

```

@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

```



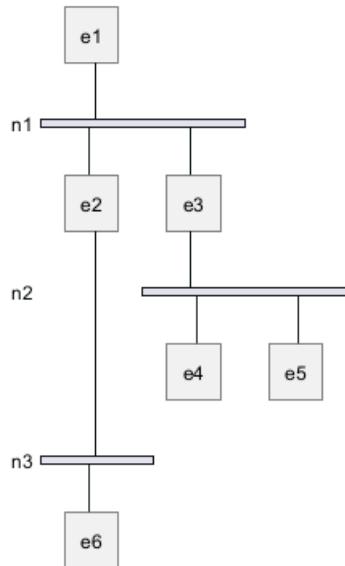
13.12.2 Second example

- without

```
@startuml
nwdiag {
    e1
    network n1 {
        e1
        e2
        e3
    }

    network n2 {
        e3
        e4
        e5
    }

    network n3 {
        e2
        e6
    }
}
@enduml
```



- only the first

```
@startuml
nwdiag {
    e1
    network n1 {
        width = full
        e1
        e2
        e3
    }
}
```



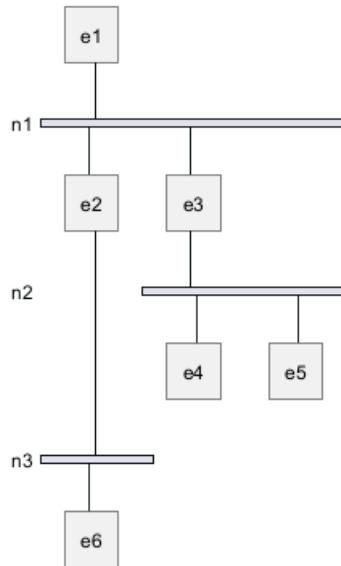
```

network n2 {
    e3
    e4
    e5
}

network n3 {
    e2
    e6
}
}

@enduml

```



- the first and the second

```

@startuml
nwdiag {
    e1
    network n1 {
        width = full
        e1
        e2
        e3
    }

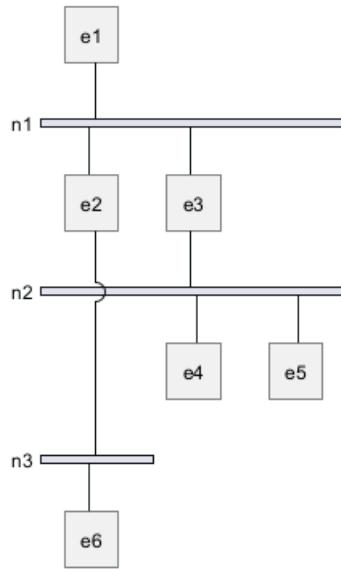
    network n2 {
        width = full
        e3
        e4
        e5
    }

    network n3 {
        e2
        e6
    }
}

```



@enduml



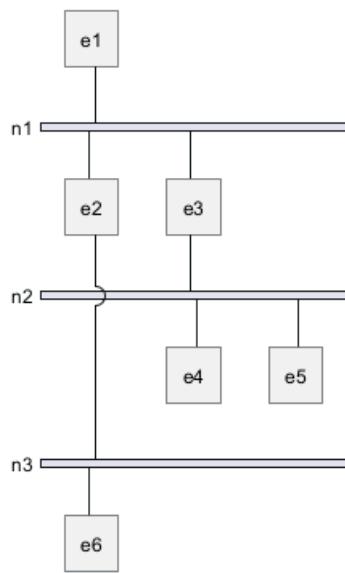
- all the network (with same full width)

```
@startuml
nwdiag {
    e1
    network n1 {
        width = full
        e1
        e2
        e3
    }

    network n2 {
        width = full
        e3
        e4
        e5
    }

    network n3 {
        width = full
        e2
        e6
    }
}
@enduml
```





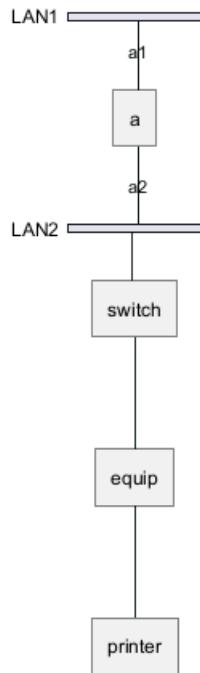
13.13 Other internal networks

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

- Without address or type

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

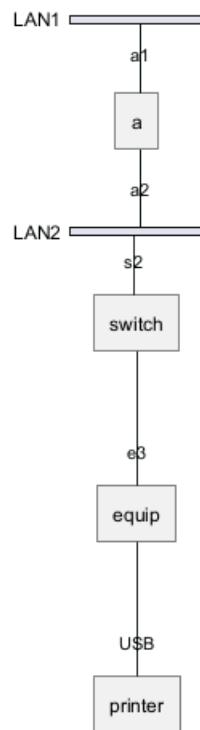




- With address or type

```
@startuml  
nwdiag {  
    network LAN1 {  
        a [address = "a1"];  
    }  
    network LAN2 {  
        a [address = "a2"];  
        switch [address = "s2"];  
    }  
    switch --> equip;  
    equip [address = "e3"];  
    equip --> printer;  
    printer [address = "USB"];  
}  
@enduml
```





[Ref. QA-12824]

13.14 Using (global) style

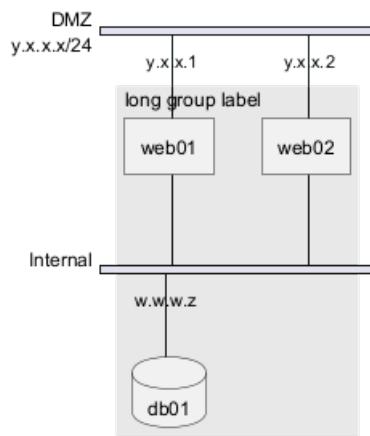
13.14.1 Without style (by default)

```
@startuml
nwdiag {
    network DMZ {
        address = "y.x.x.x/24"
        web01 [address = "y.x.x.1"];
        web02 [address = "y.x.x.2"];
    }

    network Internal {
        web01;
        web02;
        db01 [address = "w.w.w.z", shape = database];
    }

    group {
        description = "long group label";
        web01;
        web02;
        db01;
    }
}
@enduml
```





13.14.2 With style

You can use style to change rendering of elements.

```

@startuml
<style>
nwdiagDiagram {
    network {
        BackGroundColor green
        LineColor red
        LineThickness 1.0
        FontSize 18
        FontColor navy
    }
    server {
        BackGroundColor pink
        LineColor yellow
        LineThickness 1.0
        ' FontXXX only for description or label
        FontSize 18
        FontColor #blue
    }
    arrow {
        ' FontXXX only for address
        FontSize 17
        FontColor #red
        FontName Monospaced
        LineColor black
    }
    group {
        BackGroundColor cadetblue
        LineColor black
        LineThickness 2.0
        FontSize 11
        FontStyle bold
        Margin 5
        Padding 5
    }
}
</style>
nwdiag {
  
```



```

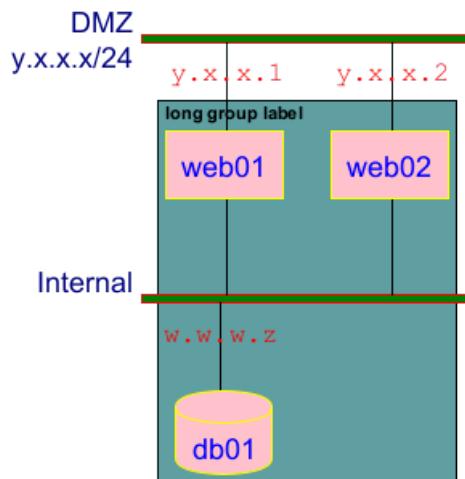
network DMZ {
    address = "y.x.x.x/24"
    web01 [address = "y.x.x.1"];
    web02 [address = "y.x.x.2"];
}

network Internal {
    web01;
    web02;
    db01 [address = "w.w.w.z", shape = database];
}

group {
    description = "long group label";
    web01;
    web02;
    db01;
}
}

@enduml

```



[Ref. QA-14479]

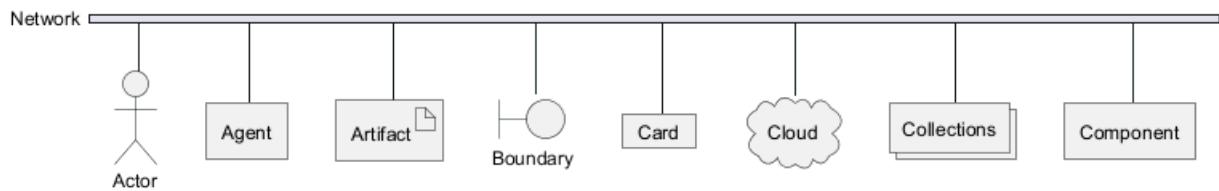
13.15 Appendix: Test of all shapes on Network diagram (nwdiag)

```

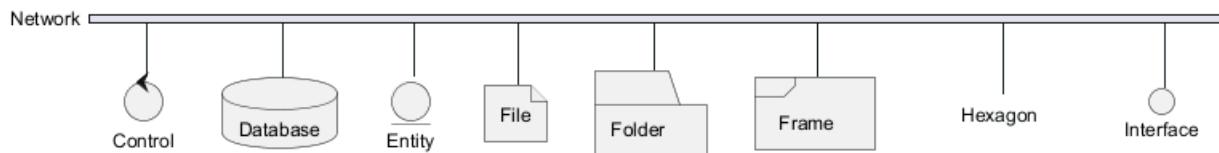
@startuml
nwdiag {
    network Network {
        Actor      [shape = actor]
        Agent      [shape = agent]
        Artifact   [shape = artifact]
        Boundary   [shape = boundary]
        Card       [shape = card]
        Cloud      [shape = cloud]
        Collections [shape = collections]
        Component   [shape = component]
    }
}
@enduml

```

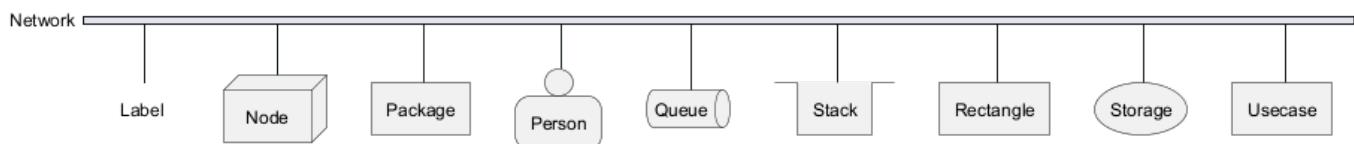




```
@startuml
nwdiag {
    network Network {
        Control      [shape = control]
        Database     [shape = database]
        Entity       [shape = entity]
        File         [shape = file]
        Folder       [shape = folder]
        Frame        [shape = frame]
        Hexagon      [shape = hexagon]
        Interface    [shape = interface]
    }
}
@enduml
```



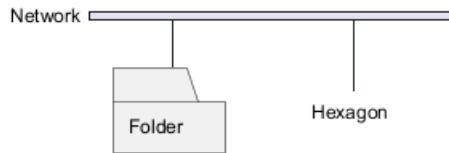
```
@startuml
nwdiag {
    network Network {
        Label        [shape = label]
        Node         [shape = node]
        Package      [shape = package]
        Person       [shape = person]
        Queue        [shape = queue]
        Stack        [shape = stack]
        Rectangle    [shape = rectangle]
        Storage      [shape = storage]
        Usecase      [shape = usecase]
    }
}
@enduml
```



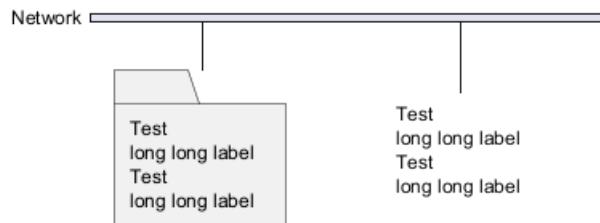
TODO: FIXME Overlap of label for folder Hexagon shape is missing



```
@startuml  
nwdiag {  
network Network {  
Folder [shape = folder]  
Hexagon [shape = hexagon]  
}  
}  
@enduml
```



```
@startuml  
nwdiag {  
network Network {  
Folder [shape = folder, description = "Test, long long label\\nTest, long long label"]  
Hexagon [shape = hexagon, description = "Test, long long label\\nTest, long long label"]  
}  
}  
@enduml
```



TODO: FIXME



14 Salt (Wireframe)

Salt - это подпроект PlantUML, который может помочь вам в разработке графического интерфейса или *Схема сайта или Схема страницы или Чертеж экрана*.

Он очень полезен при создании **графических интерфейсов**, схем и чертежей. Он помогает согласовать **концептуальные структуры с визуальным дизайном**, делая акцент на **функциональности, а не на эстетике**. **Wireframes**, занимающие центральное место в этом процессе, используются в различных дисциплинах.

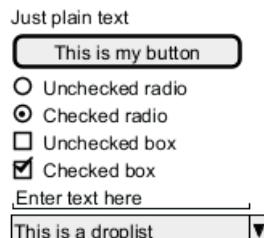
Разработчики, дизайнеры и специалисты по работе с пользователями используют их для визуализации **элементов интерфейса, навигационных систем** и облегчения совместной работы. Они различаются по степени **достоверности**- от эскизов с низкой детализацией до высокодетальных изображений, что очень важно для **создания прототипов и итеративного проектирования**. Этот совместный процесс объединяет различные знания и опыт, от **бизнес-анализа до исследования пользователей**, обеспечивая соответствие конечного проекта **требованиям бизнеса и пользователей**.

14.1 Простые виджеты

Окно должно начинаться и заканчиваться скобками. Вы можете определить:

- Кнопку, используя [и].
- Radio button (переключатель), используя (и).
- Checkbox (флажок), используя [и].
- Поле ввода текста, используя ".
- Раскрывающийся список (Droplist), используя ^.

```
@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
```



Цель этого инструмента - обсуждать простые и типовые окна.

14.2 Text area

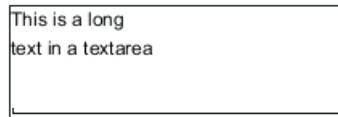
Here is an attempt to create a text area:

```
@startsalt
{+
    This is a long
    text in a textarea
    .
    "
}
```



```
}
```

```
@endsalt
```



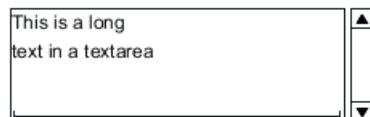
Note:

- the dot (.) to fill up vertical space;
- the last line of space (" ") to make the area wider.

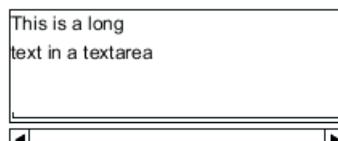
[Ref. QA-14765]

Then you can add scroll bar:

```
@startsalt
{SI
    This is a long
    text in a textarea
    .
    "
}
@endsalt
```



```
@startsalt
{S-
    This is a long
    text in a textarea
    .
    "
}
@endsalt
```



14.3 Open, close dropdown

You can open a dropdown, by adding values enclosed by ^, as:

```
@startsalt
{
    ^This is a closed dropdown^ |
    ^This is an open dropdown^^ item 1^^ item 2^ |
    ^This is another open dropdown^ item 1^ item 2^
}
@endsalt
```



[Ref. QA-4184]



14.4 Использование сетки

Таблица автоматически создаётся, когда вы используете открывающую скобку {.

Ещё вам нужно использовать | для деления на колонки.

Например:

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

Login	<input type="text" value="MyName"/>
Password	<input type="password" value="****"/>
	<input type="button" value="Cancel"/> <input type="button" value="OK"/>

Сразу после открывающей скобки вы можете использовать символ, чтобы определить, где закончится строка и начнётся колонка сетки:

Symbol	Result
#	Чтобы показать вертикальные и горизонтальные линии
!	Чтобы показать вертикальные линии
-	Чтобы показать горизонтальные линии
+	Чтобы показать внешние линии

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

Login	<input type="text" value="MyName"/>
Password	<input type="password" value="****"/>
	<input type="button" value="Cancel"/> <input type="button" value="OK"/>

14.5 Группа

Виджеты могут быть объединены в группу со своим наименованием.

Используйте символ ^, следующий сразу после открывающей скобки {.

Наименование группы задается в кавычках " сразу после символа группы ^.

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

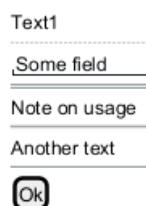
My group box	
Login	<input type="text" value="MyName"/>
Password	<input type="password" value="****"/>
	<input type="button" value="Cancel"/> <input type="button" value="OK"/>



14.6 Использование разделителя

Вы можете использовать несколько горизонтальных линий как разделитель.

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



14.7 Древовидный виджет

Чтобы создать дерево, вам нужно начать с {T и использовать + чтобы определять иерархию.

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



14.8 Таблица дерева иерархии

Возможно придание таблицам древовидной (иерархической) структуры.

```
@startsalt
{
{T
+Region      | Population    | Age
+ World      | 7.13 billion  | 30
++ America   | 964 million   | 30
+++ Canada   | 35 million    | 30
+++ USA      | 319 million   | 30
++++ NYC     | 8 million     | 30
++++ Boston   | 617 thousand  | 30
+++ Mexico   | 117 million   | 30
++ Europe    | 601 million   | 30
+++ Italy    | 61 million    | 30
+++ Germany  | 82 million    | 30
++++ Berlin   | 3 million     | 30
++ Africa    | 1 billion     | 30
}
}
@endsalt
```

Region	Population	Age
World	7.13 billion	30
America	964 million	30
Canada	35 million	30
USA	319 million	30
NYC	8 million	30
Boston	617 thousand	30
Mexico	117 million	30
Europe	601 million	30
Italy	61 million	30
Germany	82 million	30
Berlin	3 million	30
Africa	1 billion	30

Таблица может не иметь границ, как в примере выше, либо иметь только вертикальные (T!), только горизонтальные (T-), внешние границы (T+) или все границы (T#). Используйте указанные параметры сразу после открывающей скобки {.

```
@startsalt
{
..
== with T!
{T!
+Region      | Population    | Age
+ World      | 7.13 billion  | 30
++ America   | 964 million   | 30
}
..
== with T-
{T-
+Region      | Population    | Age
+ World      | 7.13 billion  | 30
++ America   | 964 million   | 30
}
..
== with T+
{T+
+Region      | Population    | Age
+ World      | 7.13 billion  | 30
```



```

++ America      | 964 million | 30
}
..
== with T#
{T#
+Region      | Population | Age
+ World      | 7.13 billion | 30
++ America    | 964 million | 30
}
..
}
@endsalt

```

with T!

Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T-

Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T+

Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T#

Region	Population	Age
World	7.13 billion	30
America	964 million	30

[Ref. QA-1265]

14.9 Окружающие скобки

Вы можете задать подэлементы, открывая новую скобку.

```

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

```

Name	<input type="text"/>
Modifiers:	<input checked="" type="radio"/> public <input type="radio"/> default <input type="radio"/> private <input type="radio"/> protected <input type="checkbox"/> abstract <input type="checkbox"/> final <input type="checkbox"/> static
Superclass:	<input type="text"/> java.lang.Object <input type="button" value="Browse..."/>

14.10 Добавление вкладок

Вы можете добавить вкладки, используя нотацию {/. Заметьте, что вы можете использовать HTML код для выделения текста жирным.

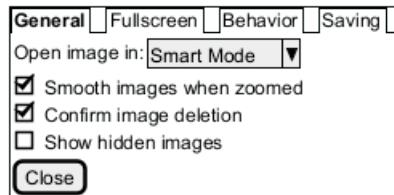
```

@startsalt
{+
{/ <b>General | Fullscreen | Behavior | Saving >
{
< Open image in: | ^Smart Mode^ >
[X] Smooth images when zoomed

```

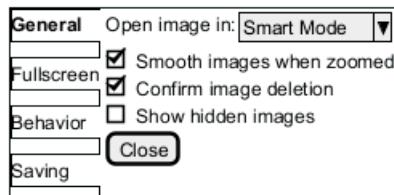


```
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```



Вкладки также могут быть вертикально ориентированы:

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

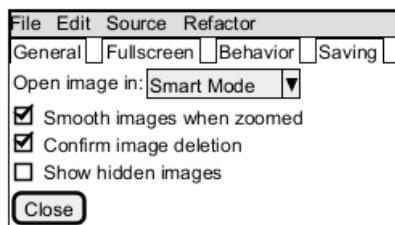


14.11 Использование меню

Вы можете добавить меню используя нотацию {*}.

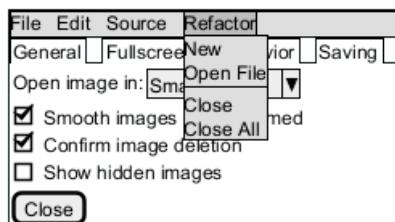
```
@startsalt
{+
{* File | Edit | Source | Refactor }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```





Также можно открыть меню:

```
@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
```



[Ref. [QA-4184](https://forum.plantuml.net/4184)]

14.12 Продвинутая таблица

Вы можете использовать две специальные нотации для таблиц :

- * чтобы показать что ячейка должна быть объединена с левой
- . чтобы обозначить пустую ячейку

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

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

14.13 Полосы прокрутки

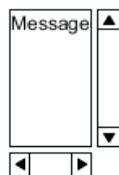
Используйте параметр S, следующий сразу после открывающей скобки {, для добавления вертикальных (S|), горизонтальных (S-) или всех (S) полос прокрутки.

```
@startsalt
{S
```



```
Message
```

```
.
.
.
}
@endsalt
```



```
@startsalt
{SI
Message
```

```
.
.
.
}
@endsalt
```



```
@startsalt
{S-
Message
```

```
.
.
.
}
@endsalt
```



14.14 Цвет элементов

Изменяйте цвет текста в виджетах по аналогии с другими диаграммами.

Для этого поставьте код `<color: _ >` перед текстом виджета.

```
@startsalt
{
<color:Blue>Just plain text
[This is my default button]
[<color:green>This is my green button]
[<color:#9a9a9a>This is my disabled button]
[] <color:red>Unchecked box
[X] <color:green>Checked box
```



```

"Enter text here      "
^This is a dropdown^
^<color:#9a9a9a>This is a disabled dropdown^
^<color:red>This is a red dropdown^
}
@endsalt

```



[Ref. QA-12177]

14.15 Creole on Salt

You can use Creole or HTML Creole on salt:

```

@startsalt
{{^==Creole
  This is **bold**
  This is //italics//
  This is ""monospaced"""
  This is --stricken-out--
  This is __underlined__
  This is ~~wave-underlined~~
  --test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
  Use image : <img:https://plantuml.com/logo3.png>
}|
{^<b>HTML Creole
  This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
  This is <s>stroked</s>
  This is <u>underlined</u>
  This is <w>waved</w>
  This is <s:green>stroked</s>
  This is <u:red>underlined</u>
  This is <w:#0000FF>waved</w>
  -- other examples --
  This is <color:blue>Blue</color>
  This is <back:orange>Orange background</back>
  This is <size:20>big</size>
}|
{^Creole line
  You can have horizontal line
  -----
  Or double line
  =====
  Or strong line
  -----

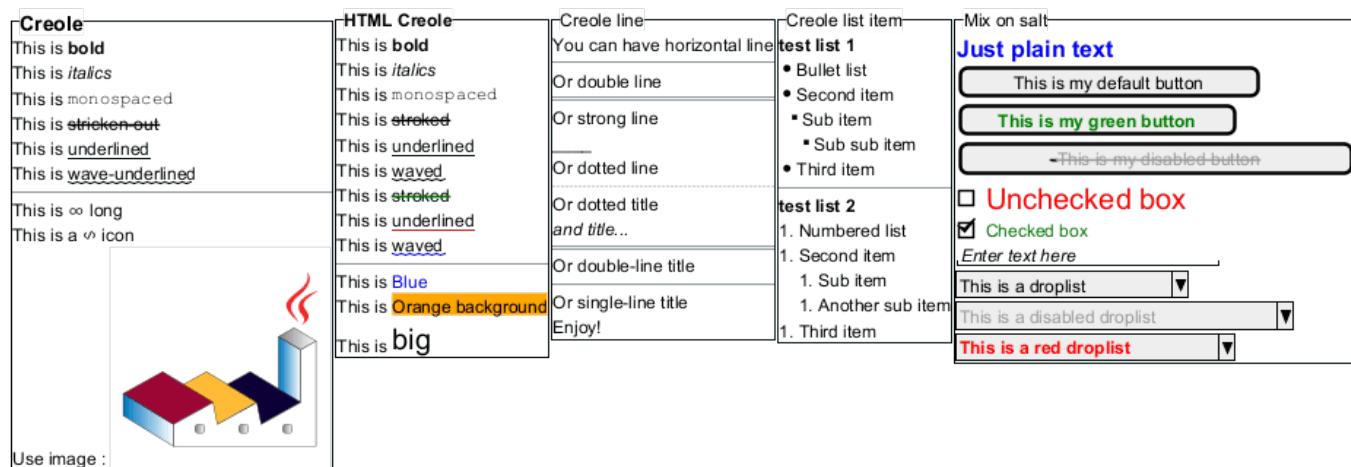
```



```

Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!
}|
{^Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}|
{^Mix on salt
==<color:Blue>Just plain text
[This is my default button]
[<b><color:green>This is my green button]
[ ---<color:#9a9a9a>This is my disabled button-- ]
[] <size:20><color:red>Unchecked box
[X] <color:green>Checked box
"/>Enter text here//"
~This is a dropdown^
`<color:#9a9a9a>This is a disabled dropdown^
`<b><color:red>This is a red dropdown^
}}
@endsalt

```



14.16 Спрайты

Используя << и >> вы можете определить псевдо-графическое изображение (спрайт) и задать его имя, с помощью которого переиспользовать его в дальнейшем.

Спрайт представляет собой текстовое поле рамером 10 строк на 12 колонок, состоящее их пустых пикселей, задаваемых точкой (.), и заполненных пикселей, задаваемых символом X.

```
@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
.....
.XXXX. .....
.X...X.....
.XXXXXXXXXXX.
.X.....X.
.X.....X.
.X.....X.
.X.....X.
.XXXXXXXX.
.....
>>|<color:blue>other folder|<<folder>>
^Dropist^
}
@endsalt
```



[Ref. QA-5849]

14.17 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

List Open Iconic	bell	cloud	excerpt	justify-right	musical-note	star
Credit to	luetooth	cloudy	expand-down	key	paperclip	sun
https://useiconic.com/open	bold	code	expand-left	laptop	pencil	tablet
↳ account-login	book	cog	expand-right	layers	people	tag
↳ account-logout	bookmark	collapse-down	expand-up	lightbulb	person	tags
↷ action-redo	box	collapse-left	external-link	link-broken	phone	target
↶ action-undo	briefcase	collapse-right	eye	link-intact	pie-chart	task
≡ align-center	british-pound	collapse-up	eyedropper	list-rich	pin	terminal
≡ align-left	browser	command	file	list	play-circle	text
≡ align-left	brush	comment-square	fire	location	plus	thumb-down
≡ align-right	bug	compass	flag	lock-locked	power-standby	thumb-up
⌚ aperture	bulhorn	contrast	flash	lock-unlocked	print	timer
↓ arrow-bottom	calculator	copywriting	folder	loop-circular	project	transfer
⌚ arrow-circle-bottom	calendar	credit-card	fork	loop-square	pulse	trash
⌚ arrow-circle-left	camera-slr	crop	fullscreen-enter	loop	puzzle-piece	underline
⌚ arrow-circle-right	dashboard	dashboard	fullscreen-exit	globe	question-mark	vertical-align-bottom
⌚ arrow-circle-top	caret-bottom	data-transfer-download	grid-four-up	magnifying-glass	rain	vertical-align-center
← arrow-left	caret-left	data-transfer-upload	grid-three-up	map-marker	random	video
→ arrow-right	caret-right	delete	grid-two-up	map	reload	volume-high
↓ arrow-thick-bottom	caret-top	dial	hard-drive	media-pause	resize-both	volume-low
← arrow-thick-left	cart	document	header	media-play	resize-height	volume-off
→ arrow-thick-right	chat	dollar	headphones	media-record	resize-width	warning
↑ arrow-thick-top	check	double-quote-sans-left	heart	media-skip-backward	rss-alt	wifi
↑ arrow-top	chevron-bottom	double-quote-sans-right	home	media-skip-forward	rss	wrench
wifi audio-spectrum	chevron-left	double-quote-serif-left	image	media-step-backward	script	x
audio	chevron-right	double-quote-serif-right	inbox	media-step-forward	share-boxed	yen
badge	chevron-top	droplet	infinity	medical-cross	share	zoom-in
ban	circle-check	eject	info	menu	shield	zoom-out
bar-chart	circle-x	elevator	italic	microphone	signal	sort-ascending
basket	clipboard	ellipses	justify-center	minus	signpost	sort-descending
battery-empty	clock	envelope-closed	justify-left	monitor	spreadsheet	spreadsheet
battery-full	cloud-download	envelope-open	€ euro	moon		
beaker	cloud-upload			move		

14.18 Add title, header, footer, caption or legend

```
@startsalt
title My title
header some header
footer some footer
caption This is caption
legend
The legend
end legend
```

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

@endsalt

some header

My title

Login	<input type="text" value="MyName"/>
Password	<input type="password" value="****"/>
<input type="button" value="Cancel"/> <input type="button" value="OK"/>	

The legend

This is caption

some footer

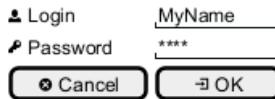


(See also: Common commands)

14.19 Zoom, DPI

14.19.1 Whitout zoom (by default)

```
@startsalt
{
    <&person> Login | "MyName"
    <&key> Password | "****"
    [<&circle-x> Cancel] | [<&account-login> OK]
}
@endsalt
```

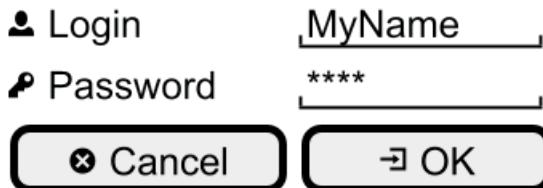


14.19.2 Scale

You can use the `scale` command to zoom the generated image.

You can use either a number or a fraction to define the scale factor. You can also specify either width or height (in pixel). And you can also give both width and height: the image is scaled to fit inside the specified dimension.

```
@startsalt
scale 2
{
    <&person> Login | "MyName"
    <&key> Password | "****"
    [<&circle-x> Cancel] | [<&account-login> OK]
}
@endsalt
```



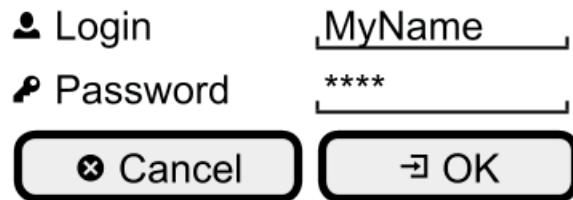
(See also: Zoom on Common commands)

14.19.3 DPI

You can also use the `skinparam dpi` command to zoom the generated image.

```
@startsalt
skinparam dpi 200
{
    <&person> Login | "MyName"
    <&key> Password | "****"
    [<&circle-x> Cancel] | [<&account-login> OK]
}
@endsalt
```





14.20 Include Salt "on activity diagram"

You can read the following explanation.

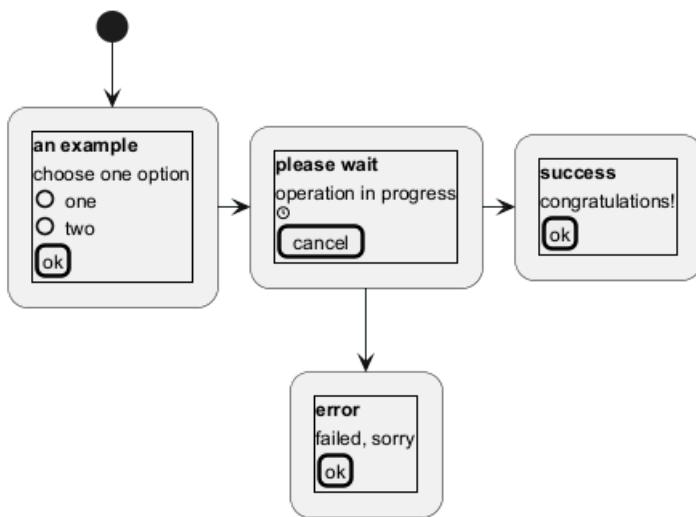
```
@startuml
(*) --> "
{{{
salt
{+
<b>an example
choose one option
()one
()two
[ok]
}
}}
" as choose

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

wait -down-> "
{{{
salt
{+
<b>error
failed, sorry
[ok]
}
}}
"
```



@enduml



It can also be combined with define macro.

```
@startuml
!unquoted procedure SALT($x)
"{
salt
%invoke_procedure("_"+$x)
}" as $x
!endprocedure
```

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

```
!procedure _wait()
{+
<b>please wait
operation in progress
<&clock>
[cancel]
}
!endprocedure
```

```
!procedure _success()
{+
<b>success
congratulations!
[ok]
}
!endprocedure
```

```
!procedure _error()
{+
<b>error
}
```

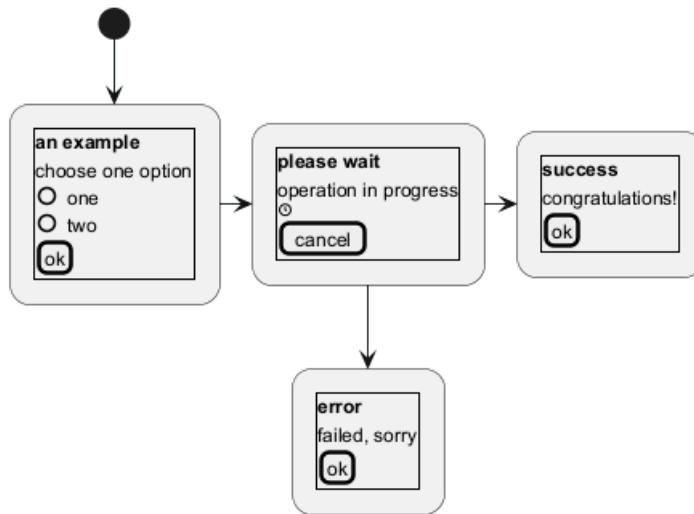


```

failed, sorry
[ok]
}
!endprocedure

(*) --> SALT(choose)
-right-> SALT(wait)
wait -right-> SALT(success)
wait -down-> SALT(error)
@enduml

```



14.21 Include salt "on while condition of activity diagram"

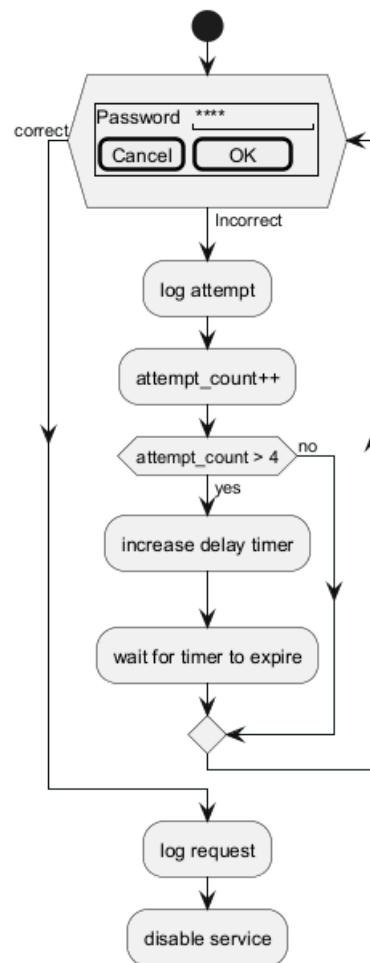
You can include salt on while condition of activity diagram.

```

@startuml
start
while (\n{`\nsalt\n{`\nPassword | "****" "\n[Cancel] | [ OK ]`}\n}) is (Incorrect)
:log attempt;
:attempt_count++;
if (attempt_count > 4) then (yes)
:increase delay timer;
:wait for timer to expire;
else (no)
endif
endwhile (correct)
:log request;
:disable service;
@enduml

```





[Ref. QA-8547]

14.22 Include salt "on repeat while condition of activity diagram"

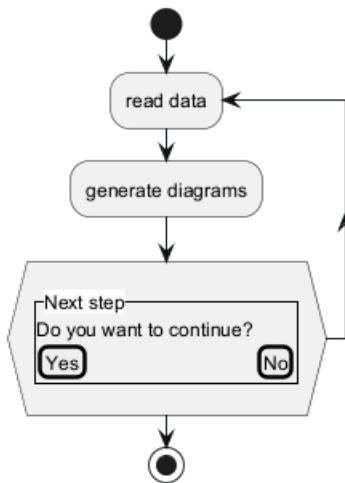
You can include salt on 'repeat while' condition of activity diagram.

```

@startuml
start
repeat :read data;
    :generate diagrams;
repeat while (\n{\nsalt\n{"Next step"\n  Do you want to continue? \n[Yes] | [No]\n}}\n)
stop
@enduml

```





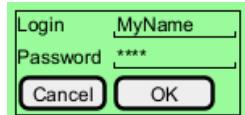
[Ref. QA-14287]

14.23 Skinparam

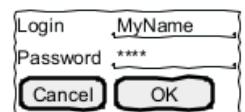
You can use [only] some skinparam command to change the skin of the drawing.

Some example:

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



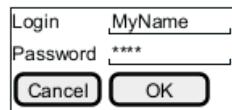
```
@startsalt
skinparam handwritten true
{+
  Login | "MyName"
  Password | "****"
  [Cancel] | [ OK ]
}
@endsalt
```



TODO: FIXME ☐ FYI, some other skinparam does not work with salt, as:

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





14.24 Style

You can use [only] some style command to change the skin of the drawing.

Some example:

```
@startsalt
<style>
saltDiagram {
    backgroundColor palegreen
}
</style>
{+
    Login | "MyName"
    Password | "****"
    [Cancel] | [ OK ]
}
@endsalt
```



TODO: FIXME ☐ FYI, some other style does not work with salt, as:

```
@startsalt
<style>
saltDiagram {
    Fontname Monospaced
    FontSize 10
    FontStyle italic
    LineThickness 0.5
    LineColor red
}
</style>
{+
    Login | "MyName"
    Password | "****"
    [Cancel] | [ OK ]
}
@endsalt
```



[Ref. QA-13460]



15 ArchiMate Diagram

ArchiMate is an open and independent **enterprise architecture modeling language** that supports the description, analysis, and visualization of architecture within and across business domains. An **ArchiMate Diagram** provides a structured representation of the various components of an enterprise, their **interrelationships**, and their integration with **IT infrastructure**.

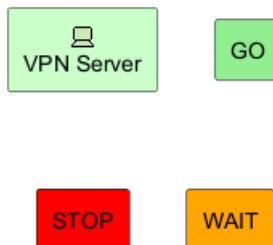
While both ArchiMate and UML are modeling languages, they serve different purposes. UML is primarily used for software design and system modeling, focusing on the structural and behavioral aspects of systems. In contrast, **ArchiMate** is tailored for **enterprise architecture**, offering a holistic view of the organizational, informational, and technical layers of an enterprise.

15.1 Archimate keyword

You can use the archimate keyword to define an element. Stereotype can optionally specify an additional icon. Some colors (Business, Application, Motivation, Strategy, Technology, Physical, Implementation) are also available.

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

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



15.2 Defining Junctions

Using the `circle` keyword and the preprocessor, you can also create junctions.

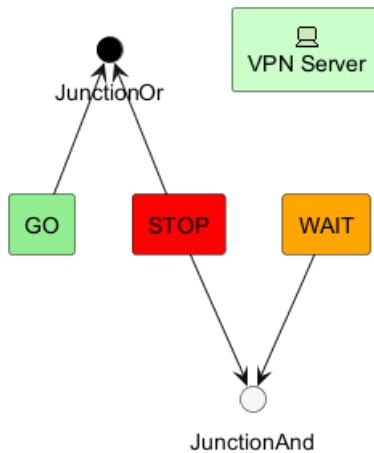
```
@startuml
!define Junction_Or circle #black
!define Junction_And circle #whitesmoke

Junction_And JunctionAnd
Junction_Or JunctionOr

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
GO -up-> JunctionOr
STOP -up-> JunctionOr
STOP -down-> JunctionAnd
WAIT -down-> JunctionAnd
@enduml
```





15.3 Example 1

```

@startuml
skinparam rectangle<<behavior>> {
roundCorner 25
}
sprite $bProcess jar:archimate/business-process
sprite $aService jar:archimate/application-service
sprite $aComponent jar:archimate/application-component

rectangle "Handle claim" as HC <<$bProcess>><<behavior>> #Business
rectangle "Capture Information" as CI <<$bProcess>><<behavior>> #Business
rectangle "Notify\nAdditional Stakeholders" as NAS <<$bProcess>><<behavior>> #Business
rectangle "Validate" as V <<$bProcess>><<behavior>> #Business
rectangle "Investigate" as I <<$bProcess>><<behavior>> #Business
rectangle "Pay" as P <<$bProcess>><<behavior>> #Business

HC *--down- CI
HC *--down- NAS
HC *--down- V
HC *--down- I
HC *--down- P

CI -right->> NAS
NAS -right->> V
V -right->> I
I -right->> P

rectangle "Scanning" as scanning <<$aService>><<behavior>> #Application
rectangle "Customer admnistration" as customerAdministration <<$aService>><<behavior>> #Application
rectangle "Claims admnistration" as claimsAdministration <<$aService>><<behavior>> #Application
rectangle Printing <<$aService>><<behavior>> #Application
rectangle Payment <<$aService>><<behavior>> #Application

scanning -up-> CI
customerAdministration -up-> CI
claimsAdministration -up-> NAS
claimsAdministration -up-> V
claimsAdministration -up-> I
Payment -up-> P

Printing -up-> V
Printing -up-> P
  
```



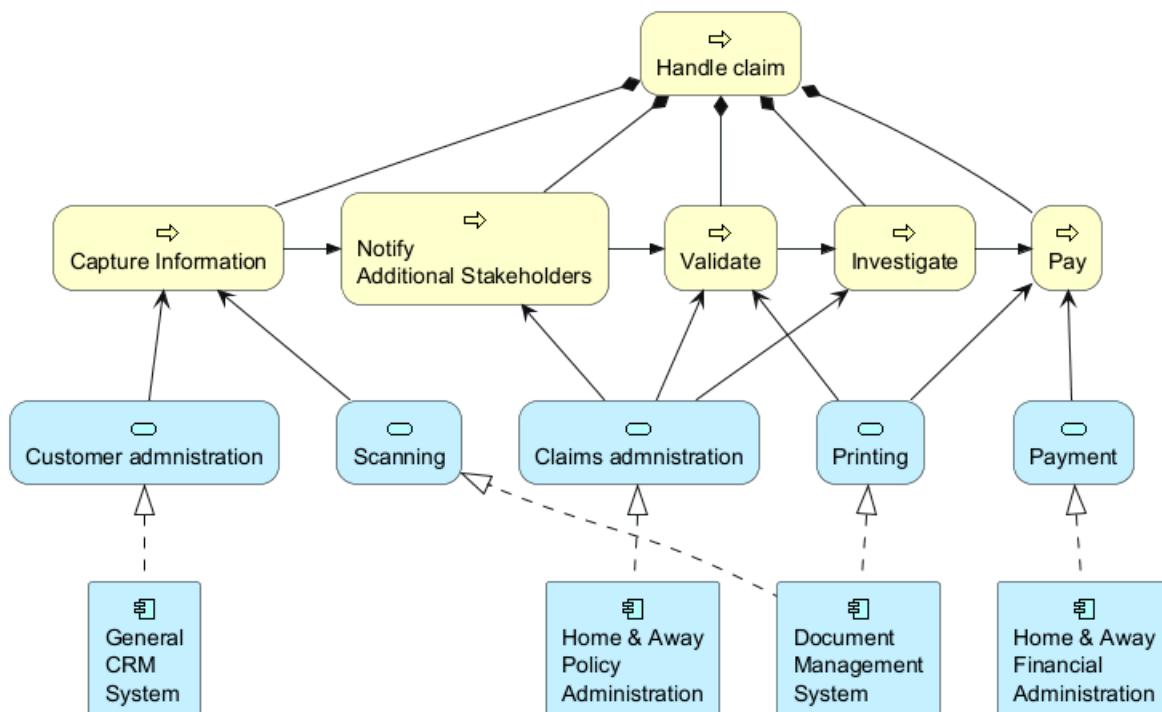
```

rectangle "Document\Management\System" as DMS <<$aComponent>> #Application
rectangle "General\CRM\System" as CRM <<$aComponent>> #Application
rectangle "Home & Away\Policy\Administration" as HAPA <<$aComponent>> #Application
rectangle "Home & Away\Financial\Administration" as HFPA <<$aComponent>> #Application

DMS .up.|> scanning
DMS .up.|> Printing
CRM .up.|> customerAdministration
HAPA .up.|> claimsAdministration
HFPA .up.|> Payment

legend left
Example from the "Archisurance case study" (OpenGroup).
See
=====
<$bProcess> :business process
=====
<$aService> : application service
=====
<$aComponent> : application component
endlegend
@enduml

```



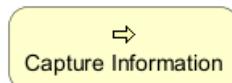
Example from the "Archisurance case study" (OpenGroup).
See
⇒ :business process
□ : application service
■ : application component

15.4 Example 2

@startuml



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



15.5 List possible sprites

You can list all possible sprites for Archimate using the following diagram:

```
@startuml
listsprite
@enduml
```

List Current Sprites

Credit to
http://www.archimatetool.com

archimate :

- access
- activity
- actor
- aggregation
- application-collaboration
- application-component
- application-data-object
- application-event
- application-function
- application-interaction
- application-interface
- application-process
- application-service
- assessment-filled
- assessment
- assignment
- association-unidirect
- association
- business-activity
- business-actor
- business-collaboration
- business-contract
- business-event
- business-function
- business-interaction
- business-interface
- business-location
- business-meaning

- business-object
- business-process
- business-product
- business-representation
- business-role
- business-service
- business-value
- collaboration
- communication-path
- component
- composition
- constraint-filled
- constraint
- contract
- deliverable-filled
- deliverable
- device
- driver-filled
- driver
- event
- flow
- function
- gap-filled
- gap
- goal-filled
- goal
- implementation-deliverable
- implementation-event
- implementation-gap
- implementation-plateau
- implementation-workpackage
- influence
- interaction
- interface-required

- junction-and
- junction-or
- junction
- location
- meaning
- motivation-assessment
- motivation-constraint
- motivation-driver
- motivation-goal
- motivation-meaning
- motivation-outcome
- motivation-principle
- motivation-requirement
- motivation-stakeholder
- motivation-value
- network
- node
- object
- physical-distribution-network
- physical-equipment
- physical-facility
- physical-material
- plateau
- principle-filled
- principle
- process
- product
- realisation
- representation
- requirement-filled
- requirement
- role

- service
- serving
- specialisation
- specialization
- stakeholder-filled
- strategy-capability
- strategy-course-of-action
- strategy-resource
- strategy-value-stream
- system-software
- technology-artifact
- technology-collaboration
- technology-communication-network
- technology-communication-path
- technology-device
- technology-event
- technology-function
- technology-infra-interface
- technology-infra-service
- technology-interaction
- technology-interface
- technology-network
- technology-node
- technology-path
- technology-process
- technology-service
- technology-system-software
- triggering
- used-by
- value
- workpackage-filled

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 (nameOfTheElement "description")

For example:

- To define a *Stakeholder* element, which is part of Motivation category, the syntax will be Motivation_Stakeholder (Stakeholder "Description"):

```
@startuml
```

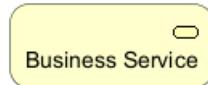


```
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
@enduml
```



- To define a *Business Service* element, `Business_Service(BService, "Business Service")`:

```
@startuml
!include <archimate/Archimate>
Business_Service(BService, "Business Service")
@enduml
```



15.6.3 Archimate relationships

The ArchiMate relationships are defined with the following pattern: `Rel_RelationType(fromElement, toElement, "description")` and to define the direction/orientation of the two elements: `Rel_RelationType_Direction(fromElement, 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:

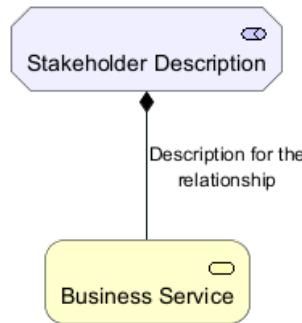
- To denote a composition relationship between the *Stakeholder* and *Business Service* defined above, the syntax will be

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

@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
Business_Service(BService, "Business Service")
```



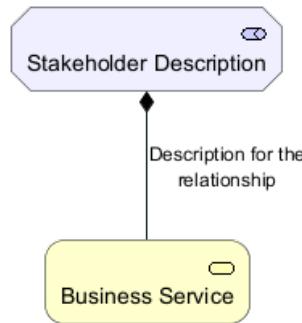
```
Rel_Composition(StakeholderElement, BService, "Description for the relationship")
@enduml
```



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

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

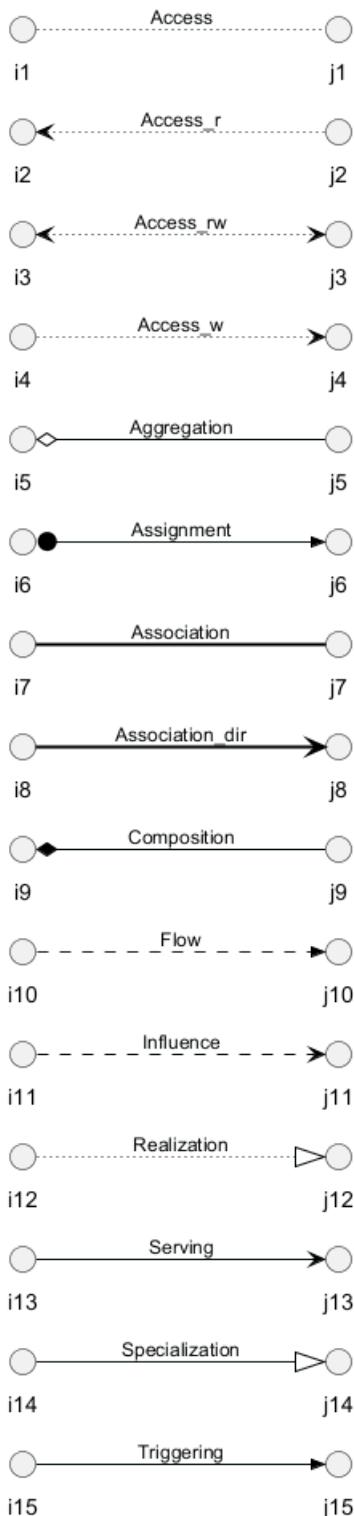
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
Business_Service(BService, "Business Service")
Rel_Composition_Down(StakeholderElement, BService, "Description for the relationship")
@enduml
```



15.6.4 Appendix: Examples of all Archimate RelationTypes

```
@startuml
left to right direction
skinparam nodesep 4
!include <archimate/Archimate>
Rel_Triggering(i15, j15, Triggering)
Rel_Specialization(i14, j14, Specialization)
Rel_Serving(i13, j13, Serving)
Rel_Realization(i12, j12, Realization)
Rel_Influence(i11, j11, Influence)
Rel_Flow(i10, j10, Flow)
Rel_Composition(i9, j9, Composition)
Rel_Association_dir(i8, j8, Association_dir)
Rel_Association(i7, j7, Association)
Rel_Assignment(i6, j6, Assignment)
Rel_Aggregation(i5, j5, Aggregation)
Rel_Access_w(i4, j4, Access_w)
Rel_Access_rw(i3, j3, Access_rw)
Rel_Access_r(i2, j2, Access_r)
Rel_Access(i1, j1, Access)
@enduml
```





```
@startuml
title ArchiMate Relationships Overview
skinparam nodesep 5
<style>
interface {
    shadowing 0
    backgroundcolor transparent
    linecolor transparent
    FontColor transparent
```



```
}

</style>
!include <archimate/Archimate>
left to right direction

rectangle Other {
() i14
() j14
}

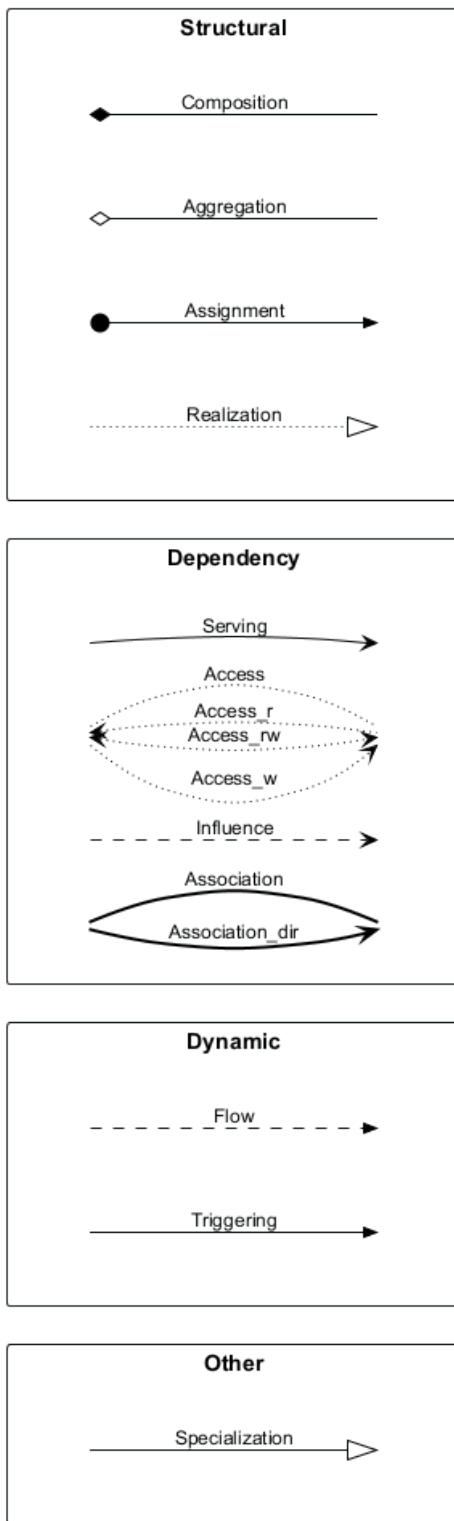
rectangle Dynamic {
() i10
() j10
() i15
() j15
}

rectangle Dependency {
() i13
() j13
() i4
() j4
() i11
() j11
() i7
() j7
}

rectangle Structural {
() i9
() j9
() i5
() j5
() i6
() j6
() i12
() j12
}

Rel_Triggering(i15, j15, Triggering)
Rel_Specialization(i14, j14, Specialization)
Rel_Serving(i13, j13, Serving)
Rel_Realization(i12, j12, Realization)
Rel_Influence(i11, j11, Influence)
Rel_Flow(i10, j10, Flow)
Rel_Composition(i9, j9, Composition)
Rel_Association_dir(i7, j7, \nAssociation_dir)
Rel_Association(i7, j7, Association)
Rel_Assignment(i6, j6, Assignment)
Rel_Aggregation(i5, j5, Aggregation)
Rel_Access_w(i4, j4, Access_w)
Rel_Access_rw(i4, j4, Access_rw)
Rel_Access_r(i4, j4, Access_r)
Rel_Access(i4, j4, Access)
@enduml
```



ArchiMate Relationships Overview

[Adapted from Archimate PR#25]



16 Диаграмма Ганта

Диаграмма Ганта - это мощный инструмент, используемый для **управления проектами**. Она наглядно представляет **график проекта**, позволяя руководителям и членам команды сразу увидеть даты начала и окончания всего проекта. Диаграмма отображает задачи или виды деятельности вдоль горизонтальной оси времени, показывая **продолжительность** каждой задачи, их **последовательность**, а также то, как они пересекаются или выполняются одновременно.

На диаграмме Ганта каждая задача представлена в виде столбика, длина и положение которого отражают **дату начала, продолжительность и дату окончания** задачи. Такой формат позволяет легко понять **зависимость** между задачами, когда одна задача должна быть завершена до начала выполнения другой. Кроме того, диаграммы Ганта могут включать **вехи** - значимые события или цели на временной шкале проекта, обозначенные отдельным символом.

В контексте создания диаграмм Ганта **PlantUML** обладает рядом преимуществ. Он обеспечивает **текстовый подход к созданию** диаграмм, что позволяет легко отслеживать изменения с помощью **систем контроля версий**. Такой подход особенно удобен для команд, которые уже привыкли к текстовым средам кодирования. Синтаксис диаграмм Ганта в PlantUML **прост**, что позволяет быстро вносить изменения и дополнения в график проекта. Кроме того, **интеграция PlantUML с другими инструментами** и способность динамически генерировать диаграммы из текста делают его универсальным выбором для команд, стремящихся автоматизировать и упорядочить документацию по управлению проектами. Таким образом, использование PlantUML для диаграмм Ганта сочетает в себе **наглядность и эффективность** визуального планирования проекта с **гибкостью и управляемостью** текстовой системы.

16.1 Объявление задач

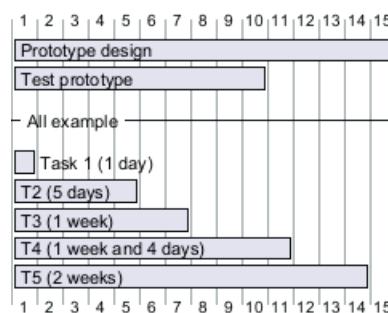
Гантт описывается на *естественному языке*, с использованием очень простых предложений (субъект-глагол-комплимент).

Задачи определяются с помощью квадратных скобок.

16.1.1 Объем работы

Объем работы для каждой задачи задается с помощью глагола `requires`, указывающего на объем необходимой работы в днях.

```
@startgantt
[Prototype design] requires 15 days
[Test prototype] requires 10 days
-- All example --
[Task 1 (1 day)] requires 1 day
[T2 (5 days)] requires 5 days
[T3 (1 week)] requires 1 week
[T4 (1 week and 4 days)] requires 1 week and 4 days
[T5 (2 weeks)] requires 2 weeks
@endgantt
```



Под неделей обычно понимается промежуток времени, состоящий из семи дней. Однако в условиях, когда некоторые дни считаются "закрытыми" (например, выходные), неделя может быть переформулирована в терминах "незакрытых" дней. Например, если суббота и воскресенье обозначены как закрытые, то неделя в данном контексте будет равна пятидневной рабочей нагрузке, соответствующей оставшимся дням недели.



16.1.2 Начало

Их начало определяется с помощью глагола start:

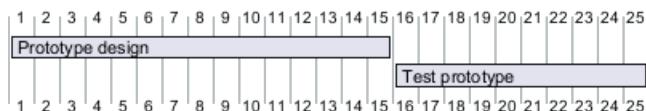
```
@startgantt
[Prototype design] requires 15 days
[Test prototype] requires 10 days
```

```
Project starts 2020-07-01
[Prototype design] starts 2020-07-01
[Test prototype] starts 2020-07-16
@endgantt
```



```
@startgantt
[Prototype design] requires 15 days
[Test prototype] requires 10 days
```

```
[Prototype design] starts D+0
[Test prototype] starts D+15
@endgantt
```



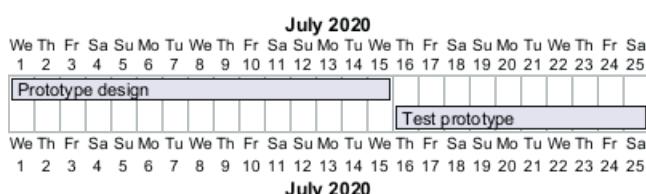
[Ссылка на форму D+nn: QA-14494]

16.1.3 Конец

Их окончание определяется с помощью глагола end:

```
@startgantt
[Prototype design] requires 15 days
[Test prototype] requires 10 days
```

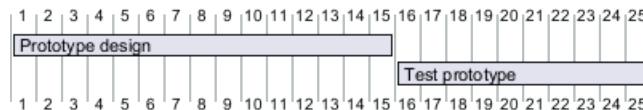
```
Project starts 2020-07-01
[Prototype design] ends 2020-07-15
[Test prototype] ends 2020-07-25
@endgantt
```



```
@startgantt
[Prototype design] requires 15 days
[Test prototype] requires 10 days
```

```
[Prototype design] ends D+14
[Test prototype] ends D+24
@endgantt
```





16.1.4 Start/End

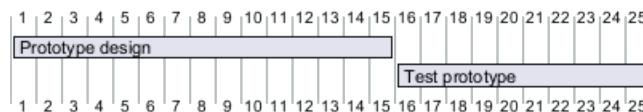
Можно определить оба абсолютно, указав даты:

```
@startgantt
Project starts 2020-07-01
[Prototype design] starts 2020-07-01
[Test prototype] starts 2020-07-16
[Prototype design] ends 2020-07-15
[Test prototype] ends 2020-07-25
```

```
@endgantt
```



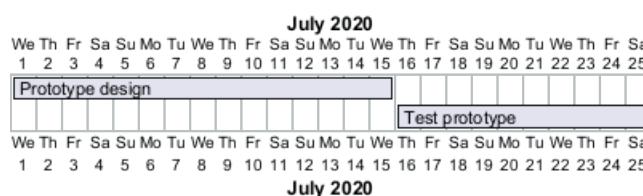
```
@startgantt
[Prototype design] starts D+0
[Test prototype] starts D+15
[Prototype design] ends D+14
[Test prototype] ends D+24
@endgantt
```



16.2 Однострочное объявление (с конъюнкцией and)

Возможно объединение объявления в одной строке с конъюнкцией and.

```
@startgantt
Project starts 2020-07-01
[Prototype design] starts 2020-07-01 and ends 2020-07-15
[Test prototype] starts 2020-07-16 and requires 10 days
@endgantt
```

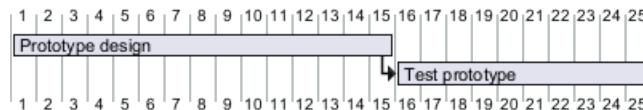


16.3 Добавление ограничений

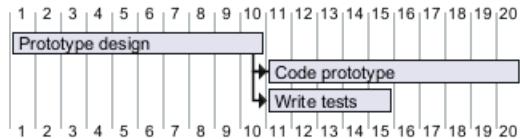
Существует возможность добавления ограничений между задачами.

```
@startgantt
[Prototype design] requires 15 days
[Test prototype] requires 10 days
[Test prototype] starts at [Prototype design]'s end
@endgantt
```





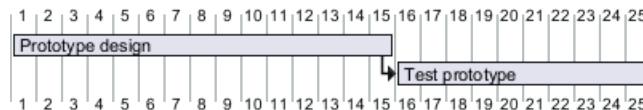
```
@startgantt
[Prototype design] requires 10 days
[Code prototype] requires 10 days
[Write tests] requires 5 days
[Code prototype] starts at [Prototype design]'s end
[Write tests] starts at [Code prototype]'s start
@endgantt
```



16.4 Короткие имена и псевдонимы задач (алиасы)

С помощью ключевого слова as можно определить короткое имя для задач.

```
@startgantt
[Prototype design] as [D] requires 15 days
[Test prototype] as [T] requires 10 days
[T] starts at [D]'s end
@endgantt
```



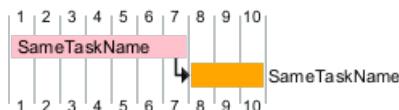
16.5 Задачи с одинаковыми наименованиями

[Начиная с версии V1.2024.6], появилась возможность создавать задачи с одинаковыми наименованиями.

```
@startgantt
Project starts 2020-11-08
[Task 7 days] as [T7] starts at 2020-11-09
[T7] ends at 2020-11-15
[Task 7 days] as [T7bis] starts at 2020-11-09
[T7bis] ends at 2020-11-15
@endgantt
```



```
@startgantt
[SameTaskName] as [T1] lasts 7 days and is colored in pink
[SameTaskName] as [T2] lasts 3 days and is colored in orange
[T1] -> [T2]
@endgantt
```



[Cм. QA-12176 и GH-1809]



16.6 Настройка цветов

Также возможна настройка цветов с помощью `is colored in`.

```
@startgantt
[Prototype design] requires 13 days
[Test prototype] requires 4 days
[Test prototype] starts at [Prototype design]'s end
[Prototype design] is colored in Fuchsia/FireBrick
[Test prototype] is colored in GreenYellow/Green
@endgantt
```



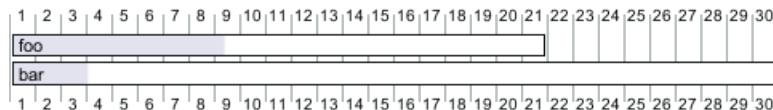
16.7 Статус завершения

16.7.1 Добавление процента завершности

Вы можете установить статус завершения задания с помощью команды:

- `is xx% completed`
- `is xx% complete`

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



16.7.2 Изменить цвет завершения (при помощи модуля <style>)

```
@startgantt

<style>
ganttDiagram {
    task {
        BackGroundColor GreenYellow
        LineColor Green
        unstarted {
            BackGroundColor Fuchsia
            LineColor FireBrick
        }
    }
}
</style>
```

```
[Prototype design] requires 7 days
[Test prototype 0] requires 4 days
[Test prototype 10] requires 4 days
[Test prototype 20] requires 4 days
[Test prototype 30] requires 4 days
[Test prototype 40] requires 4 days
[Test prototype 50] requires 4 days
[Test prototype 60] requires 4 days
```

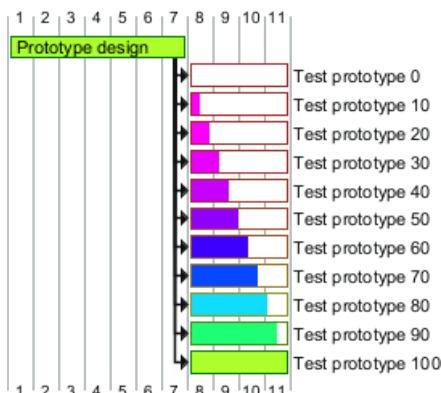


```
[Test prototype 70] requires 4 days
[Test prototype 80] requires 4 days
[Test prototype 90] requires 4 days
[Test prototype 100] requires 4 days

[Test prototype 0] starts at [Prototype design]'s end
[Test prototype 10] starts at [Prototype design]'s end
[Test prototype 20] starts at [Prototype design]'s end
[Test prototype 30] starts at [Prototype design]'s end
[Test prototype 40] starts at [Prototype design]'s end
[Test prototype 50] starts at [Prototype design]'s end
[Test prototype 60] starts at [Prototype design]'s end
[Test prototype 70] starts at [Prototype design]'s end
[Test prototype 80] starts at [Prototype design]'s end
[Test prototype 90] starts at [Prototype design]'s end
[Test prototype 100] starts at [Prototype design]'s end

[Test prototype 0] is 0% complete
[Test prototype 10] is 10% complete
[Test prototype 20] is 20% complete
[Test prototype 30] is 30% complete
[Test prototype 40] is 40% complete
[Test prototype 50] is 50% complete
[Test prototype 60] is 60% complete
[Test prototype 70] is 70% complete
[Test prototype 80] is 80% complete
[Test prototype 90] is 90% complete
[Test prototype 100] is 100% complete
```

@endgantt



[Ref. QA-8297]

16.7.3 Изменение цвета незаконченной части задачи (при помощи модуля <code><style></code>)

```
@startgantt
<style>
ganttDiagram {
    task {
        BackGroundColor GreenYellow
        LineColor Green
    }
    undone {
        BackGroundColor red
    }
}
```



```
</style>
```

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



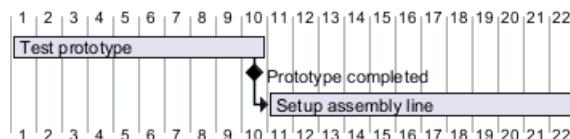
[Ref. QA-15299]

16.8 Вехи

Вы можете определить вехи с помощью команды happen.

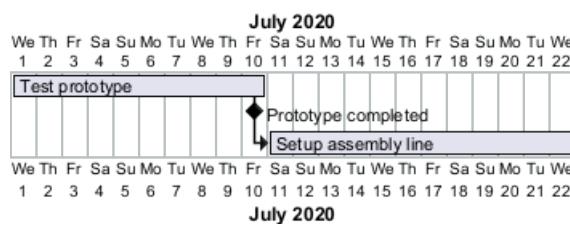
16.8.1 Относительная веха (использование ограничений)

```
@startgantt
[Test prototype] requires 10 days
[Prototype completed] happens at [Test prototype]'s end
[Setup assembly line] requires 12 days
[Setup assembly line] starts at [Test prototype]'s end
@endgantt
```



16.8.2 Абсолютная веха (использование фиксированной даты)

```
@startgantt
Project starts 2020-07-01
[Test prototype] requires 10 days
[Prototype completed] happens 2020-07-10
[Setup assembly line] requires 12 days
[Setup assembly line] starts at [Test prototype]'s end
@endgantt
```



16.8.3 Веха максимального завершения задач

```
@startgantt
[Task1] requires 4 days
then [Task1.1] requires 4 days
[Task1.2] starts at [Task1]'s end and requires 7 days

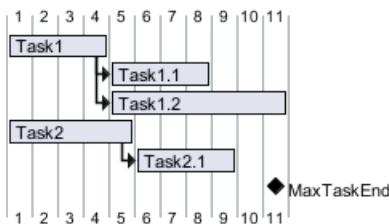
[Task2] requires 5 days
then [Task2.1] requires 4 days

[MaxTaskEnd] happens at [Task1.1]'s end
```



```
[MaxTaskEnd] happens at [Task1.2]'s end
[MaxTaskEnd] happens at [Task2.1]'s end
```

```
@endgantt
```

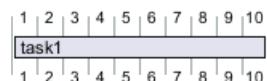


[Ref. QA-10764]

16.9 Гиперссылки

Вы можете добавлять гиперссылки в задачи.

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



16.10 Календарь

Вы можете указать дату начала выполнения всего проекта. По умолчанию первая задача начинается с этой даты.

```
@startgantt
Project starts the 20th of september 2017
[Prototype design] as [TASK1] requires 13 days
[TASK1] is colored in Lavender/LightBlue
@endgantt
```



16.11 Раскраска дней

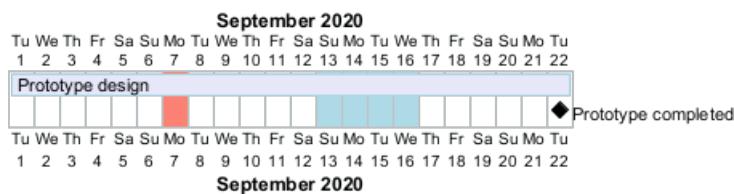
Можно добавить цвета в некоторые дни.

```
@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] requires 22 days
[TASK1] is colored in Lavender/LightBlue
[Prototype completed] happens at [TASK1]'s end
@endgantt
```





16.12 Календарные шкалы

Вы можете изменить масштаб для очень длинного проекта, используя один из этих параметров:

- printscale
- ganttscale
- projectcale

и одно из значений:

- daily —дневная шкала(*по умолчанию*)
- weekly —недельная
- monthly —месячная
- quarterly —квартальная
- yearly —годовая.

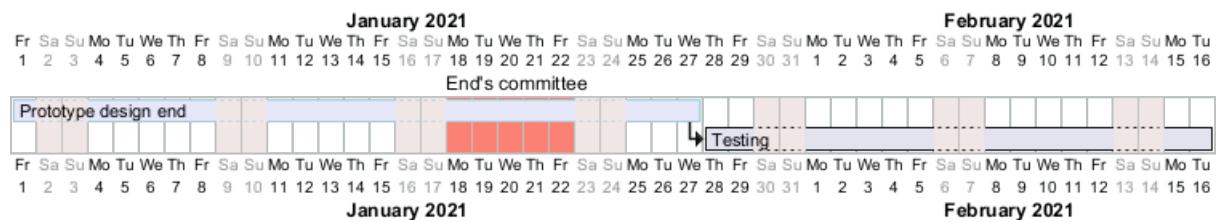
(См. QA-11272, QA-9041 и QA-10948)

16.12.1 Дневная шкала (*по умолчанию*)

```
@startgantt
saturday are closed
sunday are closed
```

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] requires 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] requires 14 days
 [TASK1] -> [Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.12.2 Недельная шкала

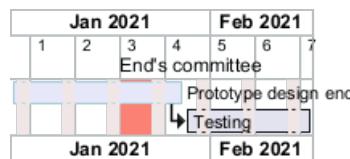
```
@startgantt
printscale weekly
saturday are closed
sunday are closed
```

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] requires 19 days



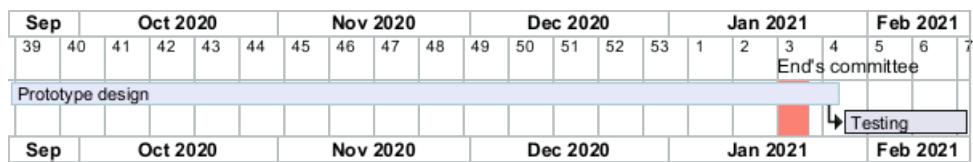
[TASK1] is colored in Lavender/LightBlue
 [Testing] requires 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



@startgantt
 printscale weekly
 Project starts the 20th of september 2020
 [Prototype design] as [TASK1] requires 130 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] requires 20 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.12.3 Месячная шкала

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

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



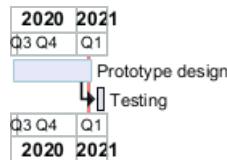
16.12.4 Квартальная шкала

@startgantt
 projectscale quarterly
 Project starts the 20th of september 2020
 [Prototype design] as [TASK1] requires 130 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] requires 20 days



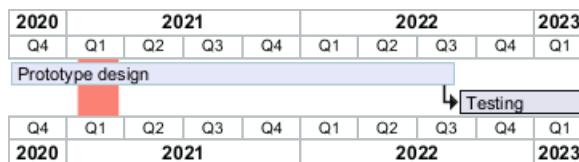
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



@startgantt
 projectscale quarterly
 Project starts the 1st of october 2020
 [Prototype design] as [TASK1] requires 700 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] requires 200 days
 [TASK1]->[Testing]

2021-01-18 to 2021-03-22 are colored in salmon
 @endgantt



16.12.5 Годовая шкала

@startgantt
 projectscale yearly
 Project starts the 1st of october 2020
 [Prototype design] as [TASK1] requires 700 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] requires 200 days
 [TASK1]->[Testing]

2021-01-18 to 2021-03-22 are colored in salmon
 @endgantt



16.12.6 Шкала точного периода командой between

16.12.7 Без точного периода

@startgantt
 saturday are closed
 sunday are closed

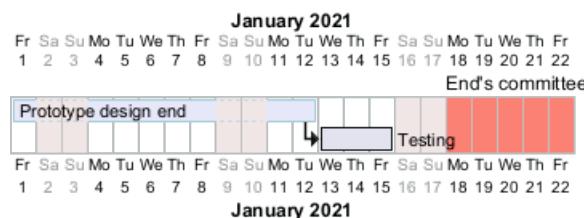
Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] requires 8 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] requires 3 days
 [TASK1]->[Testing]



2021-01-18 to 2021-01-22 are named [End's committee]

2021-01-18 to 2021-01-22 are colored in salmon

@endgantt



16.12.8 С точным периодом

@startgantt

Print between 2021-01-12 and 2021-01-22

Saturday are closed

sunday are closed

Project starts the 1st of january 2021

[Prototype design end] as [TASK1] requires 8 days

[TASK1] is colored in Lavender/LightBlue

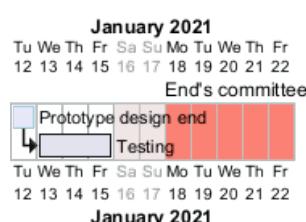
[Testing] requires 3 days

[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]

2021-01-18 to 2021-01-22 are colored in salmon

@endgantt



16.13 Масштабирование (пример для всех шкал)

С помощью параметра ниже можно изменить масштаб:

- zoom <integer>

16.13.1 Масштаб на дневной шкале

16.13.2 Без увеличения

@startgantt
printscale daily
saturday are closed
sunday are closed

Project starts the 1st of january 2021

[Prototype design end] as [TASK1] requires 8 days

[TASK1] is colored in Lavender/LightBlue

[Testing] requires 3 days

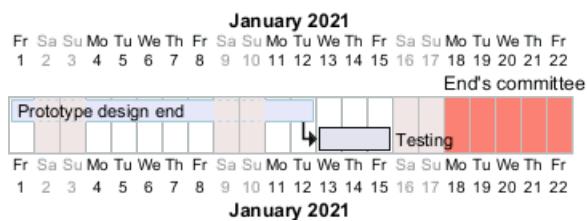
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]

2021-01-18 to 2021-01-22 are colored in salmon

@endgantt



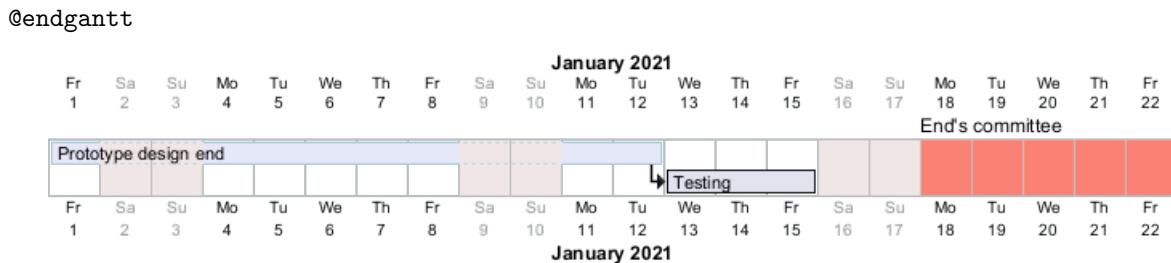


16.13.3 С увеличением

```
@startgantt
printscale daily zoom 2
saturday are closed
sunday are closed

Project starts the 1st of january 2021
[Prototype design end] as [TASK1] requires 8 days
[TASK1] is colored in Lavender/LightBlue
[Testing] requires 3 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon



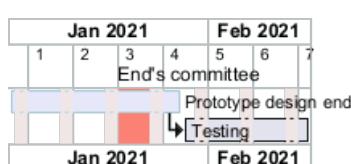
[Ref. QA-13725]

16.13.4 Масштаб на недельной шкале

```
@startgantt
printscale weekly
saturday are closed
sunday are closed
```

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] requires 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] requires 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



16.13.6 С увеличением

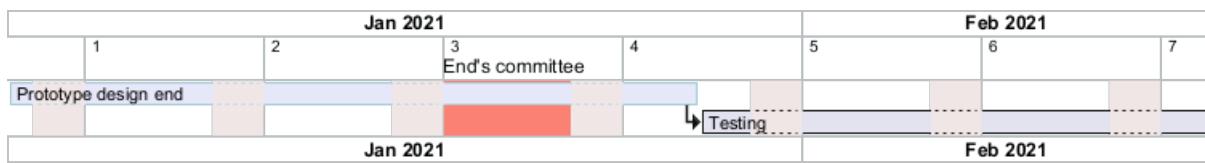
```
@startgantt
printscale weekly zoom 4
saturday are closed
sunday are closed

Project starts the 1st of january 2021
[Prototype design end] as [TASK1] requires 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] requires 14 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-01-22 are named [End's committee]

2021-01-18 to 2021-01-22 are colored in salmon

@endgantt



16.13.7 Масштаб на месячной шкале

16.13.8 Без увеличения

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

2021-01-18 to 2021-01-22 are named [End's committee]

2021-01-18 to 2021-01-22 are colored in salmon

@endgantt



16.13.9 С увеличением

```
@startgantt
projectscale monthly zoom 3
Project starts the 20th of september 2020
[Prototype design] as [TASK1] requires 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] requires 20 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-01-22 are named [End's committee]

2021-01-18 to 2021-01-22 are colored in salmon

@endgantt



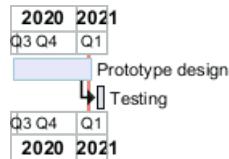


16.13.10 Масштаб на квартальной шкале

16.13.11 Без увеличения

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

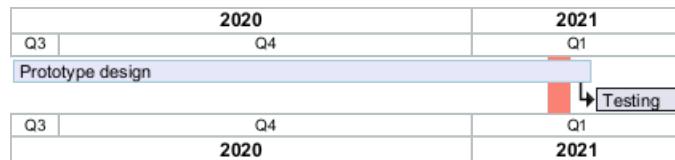
2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
@endgantt



16.13.12 С увеличением

```
@startgantt
projectscale quarterly zoom 7
Project starts the 20th of september 2020
[Prototype design] as [TASK1] requires 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] requires 20 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
@endgantt



16.13.13 Масштаб на годовой шкале

16.13.14 Без увеличения

```
@startgantt
projectscale yearly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] requires 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] requires 200 days
[TASK1]->[Testing]
```



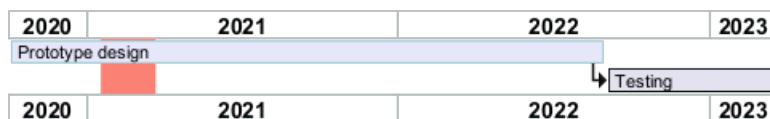
2021-01-18 to 2021-03-22 are colored in salmon
@endgantt



16.13.15 С увеличением

```
@startgantt
projectscale yearly zoom 2
Project starts the 1st of october 2020
[Prototype design] as [TASK1] requires 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] requires 200 days
[TASK1]->[Testing]
```

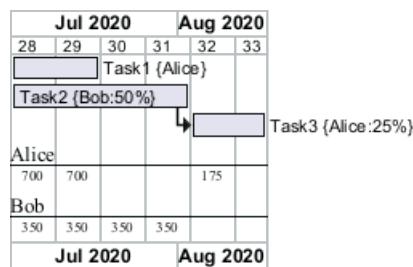
2021-01-18 to 2021-03-22 are colored in salmon
@endgantt



16.14 Недельный масштаб с номерами недель или датами

16.14.1 С указанием номеров недель (настройка по умолчанию)

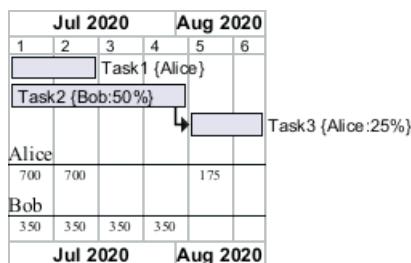
```
@startgantt
printscale weekly
Project starts the 6th of July 2020
[Task1] on {Alice} requires 2 weeks
[Task2] on {Bob:50%} requires 2 weeks
then [Task3] on {Alice:25%} requires 3 days
@endgantt
```



16.14.2 С номерами недель (начиная с 1)

```
@startgantt
printscale weekly with week numbering from 1
Project starts the 6th of July 2020
[Task1] on {Alice} requires 2 weeks
[Task2] on {Bob:50%} requires 2 weeks
then [Task3] on {Alice:25%} requires 3 days
@endgantt
```

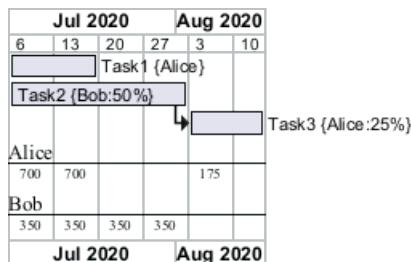




[Cм. GH-525]

16.14.3 С указанием дат

```
@startgantt
printscale weekly with calendar date
Project starts the 6th of July 2020
[Task1] on {Alice} requires 2 weeks
[Task2] on {Bob:50%} requires 2 weeks
then [Task3] on {Alice:25%} requires 3 days
@endgantt
```



[Ref. QA-11630]

16.15 Объявление дней нерабочими (закрытыми)

Можно закрыть любой день.

```
@startgantt
project starts the 2018/04/09
saturday are closed
sunday are closed
2018/05/01 is closed
2018/04/17 to 2018/04/19 is closed
[Prototype design] requires 14 days
[Test prototype] requires 4 days
[Test prototype] starts at [Prototype design]'s end
[Prototype design] is colored in Fuchsia/FireBrick
[Test prototype] is colored in GreenYellow/Green
@endgantt
```

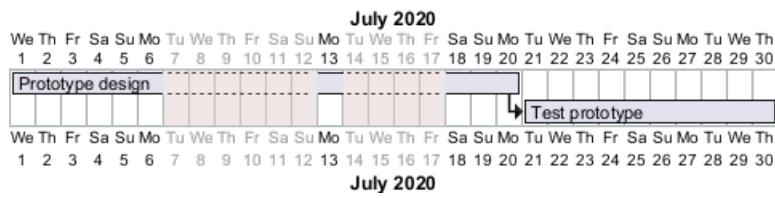


Затем можно открыть некоторый закрытый день.

```
@startgantt
2020-07-07 to 2020-07-17 is closed
2020-07-13 is open
```



Project starts the 2020-07-01
 [Prototype design] requires 10 days
 Then [Test prototype] requires 10 days
 @endgantt



16.16 Зависимость размера недели от количества нерабочих дней в ней

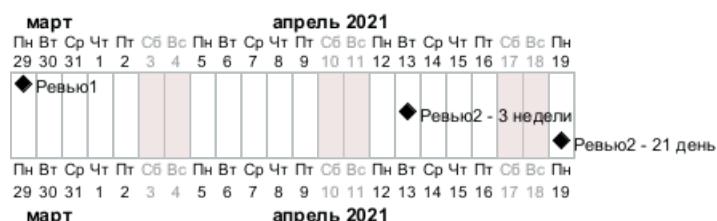
Неделя обозначается ключевым словом `week` и рассчитывается по количеству рабочих дней в ней. По умолчанию в неделе нет выходных дней, поэтому 1 неделя равняется 7 дням:

```
@startgantt
language ru
Project starts 2021-03-29
[ 1] happens at 2021-03-29
[ 2 - 3 ] happens on 3 weeks after [ 1]'s end
[ 2 - 21 ] happens on 21 days after [ 1]'s end
@endgantt
```



Если указать при помощи конструкции `saturday are closed` и `sunday are closed`, что суббота и воскресенье будут нерабочими днями, тогда размер 1 недели станет 5 дней:

```
@startgantt
language ru
Project starts 2021-03-29
saturday are closed
sunday are closed
[ 1] happens at 2021-03-29
[ 2 - 3 ] happens on 3 weeks after [ 1]'s end
[ 2 - 21 ] happens on 21 days after [ 1]'s end
@endgantt
```



[Ref. QA-13434]

16.17 Рабочие дни

Вы можете отдельно управлять перечнем рабочих дней.

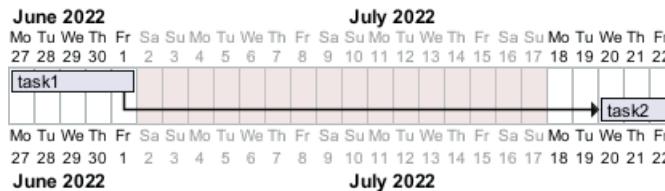
```
@startgantt
```



saturday are closed
 sunday are closed
 2022-07-04 to 2022-07-15 is closed

Project starts 2022-06-27
 [task1] starts at 2022-06-27 and requires 1 week
 [task2] starts 2 working days after [task1]'s end and requires 3 days

@endgantt

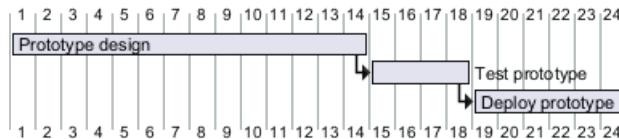


[Ref. QA-16188]

16.18 Упрощенная последовательность задач

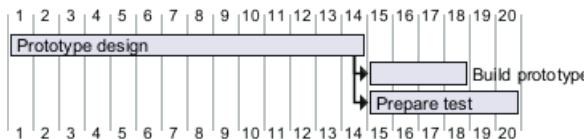
Для обозначения последовательных задач можно использовать ключевое слово `then`.

```
@startgantt
[Prototype design] requires 14 days
then [Test prototype] requires 4 days
then [Deploy prototype] requires 6 days
@endgantt
```



Также можно использовать стрелку ->

```
@startgantt
[Prototype design] requires 14 days
[Build prototype] requires 4 days
[Prepare test] requires 6 days
[Prototype design] -> [Build prototype]
[Prototype design] -> [Prepare test]
@endgantt
```

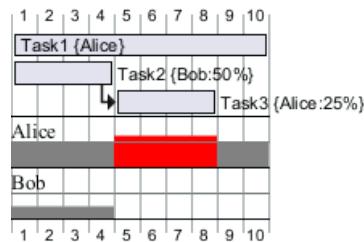


16.19 Работа с ресурсами

Вы можете воздействовать на задачи, связанные с ресурсами, используя ключевое слово `on` и скобки для имени ресурса.

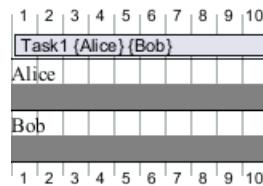
```
@startgantt
[Task1] on {Alice} requires 10 days
[Task2] on {Bob:50%} requires 2 days
then [Task3] on {Alice:25%} requires 1 days
@endgantt
```





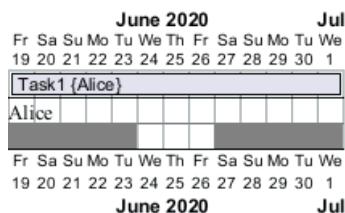
Задаче может быть назначено несколько ресурсов:

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



Ресурсы могут быть помечены как незадействованные в определенные дни:

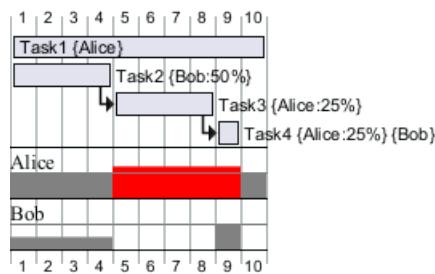
```
@startgantt
project starts on 2020-06-19
[Task1] on {Alice} requires 10 days
{Alice} is off on 2020-06-24 to 2020-06-26
@endgantt
```



16.20 Настройки отображения ресурсов

16.20.1 Ресурсы отображены (по умолчанию)

```
@startgantt
[Task1] on {Alice} requires 10 days
[Task2] on {Bob:50%} requires 2 days
then [Task3] on {Alice:25%} requires 1 days
then [Task4] on {Alice:25%} {Bob} requires 1 days
@endgantt
```

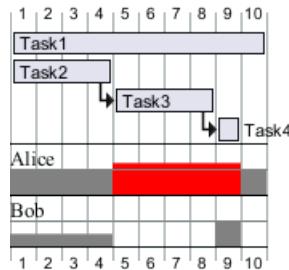


16.20.2 Имена ресурсов скрыты

Вы можете скрыть имена и процентное соотношение ресурсов в задачах, используя ключевые слова `hide resources names`.



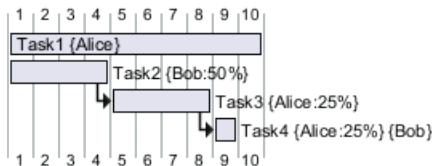
```
@startgantt
hide ressources names
[Task1] on {Alice} requires 10 days
[Task2] on {Bob:50%} requires 2 days
then [Task3] on {Alice:25%} requires 1 days
then [Task4] on {Alice:25%} {Bob} requires 1 days
@endgantt
```



16.20.3 Скрыт подвал с ресурсами

Вы также можете скрыть названия ресурсов в нижней части диаграммы, используя ключевые слова `hide ressources footbox`.

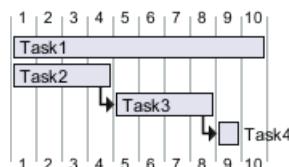
```
@startgantt
hide ressources footbox
[Task1] on {Alice} requires 10 days
[Task2] on {Bob:50%} requires 2 days
then [Task3] on {Alice:25%} requires 1 days
then [Task4] on {Alice:25%} {Bob} requires 1 days
@endgantt
```



16.20.4 Ресурсы скрыты на задачах и в подвале

Вы можете скрыть имена в обеих частях диаграммы.

```
@startgantt
hide ressources names
hide ressources footbox
[Task1] on {Alice} requires 10 days
[Task2] on {Bob:50%} requires 2 days
then [Task3] on {Alice:25%} requires 1 days
then [Task4] on {Alice:25%} {Bob} requires 1 days
@endgantt
```



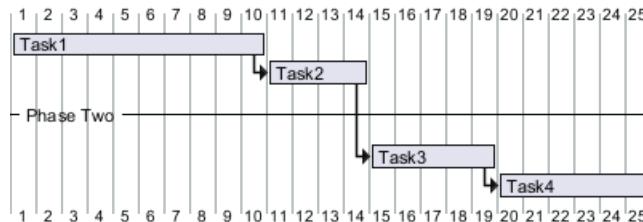
16.21 Горизонтальный разделитель

Вы можете использовать `--` для разделения наборов задач.

```
@startgantt
```



```
[Task1] requires 10 days
then [Task2] requires 4 days
-- Phase Two --
then [Task3] requires 5 days
then [Task4] requires 6 days
@endgantt
```



16.22 Вертикальный разделитель

Вы можете добавить вертикальный разделитель при помощи конструкции: Separator just [at].

```
@startgantt
[task1] requires 1 week
[task2] starts 20 days after [task1]'s end and requires 3 days
```

Separator just at [task1]'s end
 Separator just 2 days after [task1]'s end

Separator just at [task2]'s start
 Separator just 2 days before [task2]'s start
 @endgantt



[Ref. QA-16247]

16.23 Сложный пример

Также можно использовать связку and.

В ограничениях также можно добавлять задержки.

```
@startgantt
[ ] requires 13 days and is colored in Lavender/LightBlue
[ ] requires 9 days and is colored in Coral/Green and starts 3 days after [
[ ] requires 5 days and ends at [ ]'s end
[ - ] requires 6 days and ends at [ ]'s start
[ . ] is colored in Coral/Green
[ . ] starts 1 day before [ ]'s start and ends at [ ]'s end
@endgantt
```



16.24 Комментарии

Как указано на [странице "Общие команды"]([commons#560kta2oz3a2k362kjbm](#)): `blockquote` Все, что начинается с `<zcode>simple quote '</zcode>`, является комментарием.

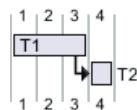
Вы также можете разместить комментарии на нескольких строках, используя `<zcode>/</zcode>` для начала и `<zcode>'</zcode>` для конца. `blockquote` (*m.e.: первым символом (кроме пробела) строки комментария должен быть символ simple quote '*)

```
@startgantt
' This is a comment

[T1] requires 3 days

/' this comment
is on several lines '/

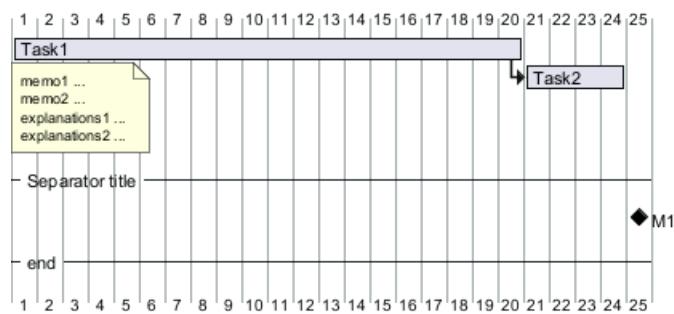
[T2] starts at [T1]'s end and requires 1 day
@endgantt
```



16.25 Использование стилей

16.25.1 Настройки по умолчанию

```
@startgantt
[Task1] requires 20 days
note bottom
    memo1 ...
    memo2 ...
    explanations1 ...
    explanations2 ...
end note
[Task2] requires 4 days
[Task1] -> [Task2]
-- Separator title --
[M1] happens on 5 days after [Task1]'s end
-- end --
@endgantt
```



16.25.2 С модулем <style>

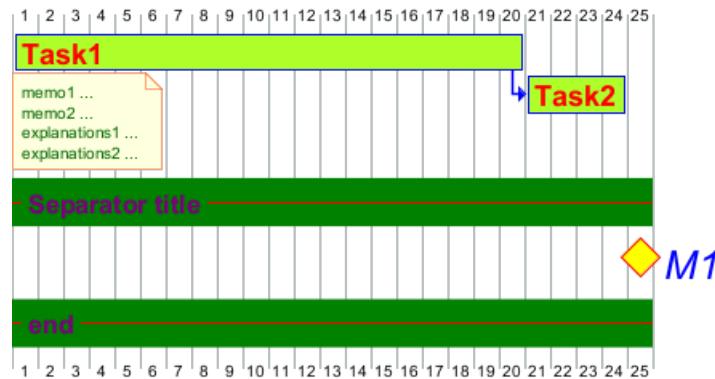
Вы можете использовать стиль для изменения отрисовки элементов.

```
@startgantt
<style>
ganttDiagram {
task {
```



```
FontName Helvetica
FontColor red
FontSize 18
FontStyle bold
BackGroundColor GreenYellow
LineColor blue
}
milestone {
FontColor blue
FontSize 25
FontStyle italic
BackGroundColor yellow
LineColor red
}
note {
FontColor DarkGreen
FontSize 10
LineColor OrangeRed
}
arrow {
FontName Helvetica
FontColor red
FontSize 18
FontStyle bold
BackGroundColor GreenYellow
LineColor blue
}
separator {
LineColor red
BackGroundColor green
FontSize 16
FontStyle bold
FontColor purple
}
}
</style>
[Task1] requires 20 days
note bottom
    memo1 ...
    memo2 ...
    explanations1 ...
    explanations2 ...
end note
[Task2] requires 4 days
[Task1] -> [Task2]
-- Separator title --
[M1] happens on 5 days after [Task1]'s end
-- end --
@endgantt
```





[Cм. QA-10835, QA-12045, QA-11877 и PR-438]

16.25.3 Индивидуальная настройка графических стилей (полный пример)

```
@startgantt
<style>
ganttDiagram {
task {
    FontName Helvetica
    FontColor red
    FontSize 18
    FontStyle bold
    BackGroundColor GreenYellow
    LineColor blue
}
milestone {
    FontColor blue
    FontSize 25
    FontStyle italic
    BackGroundColor yellow
    LineColor red
}
note {
    FontColor DarkGreen
    FontSize 10
    LineColor OrangeRed
}
arrow {
    FontName Helvetica
    FontColor red
    FontSize 18
    FontStyle bold
    BackGroundColor GreenYellow
    LineColor blue
    LineStyle 8.0;13.0
    LineThickness 3.0
}
separator {
    BackgroundColor lightGreen
    LineStyle 8.0;3.0
    LineColor red
    LineThickness 1.0
    FontSize 16
    FontStyle bold
    FontColor purple
    Margin 5
}
}
```



```

Padding 20
}
timeline {
    BackgroundColor Bisque
}
closed {
BackgroundColor pink
FontColor red
}
}
</style>
Project starts the 2020-12-01

[Task1] requires 10 days
sunday are closed

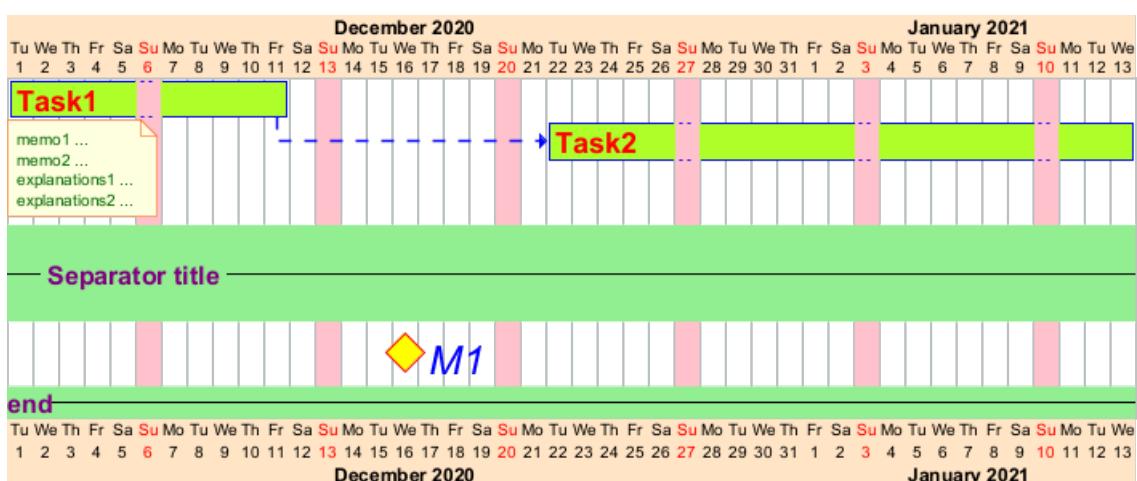
note bottom
memo1 ...
memo2 ...
explanations1 ...
explanations2 ...
end note

[Task2] requires 20 days
[Task2] starts 10 days after [Task1]'s end
-- Separator title --
[M1] happens on 5 days after [Task1]'s end

<style>
separator {
    LineColor black
Margin 0
Padding 0
}
</style>

-- end --
@endgantt

```



[Cм. QA-13570, QA-13672]

DONE Спасибо за стиль для Separator и все стили для Arrow (толщина...)

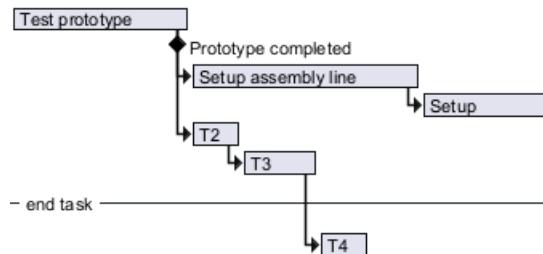


16.25.4 Минималистичный стиль диаграммы

С помощью стиля можно также очистить диаграмму Ганта (*показывая только задачи, зависимости и относительную длительность - но без фактической даты начала и фактического масштаба*):

```
@startgantt
<style>
ganttDiagram {
    timeline {
        LineColor transparent
        FontColor transparent
    }
}
</style>

hide footbox
[Test prototype] requires 7 days
[Prototype completed] happens at [Test prototype]'s end
[Setup assembly line] requires 9 days
[Setup assembly line] starts at [Test prototype]'s end
then [Setup] requires 5 days
[T2] requires 2 days and starts at [Test prototype]'s end
then [T3] requires 3 days
-- end task --
then [T4] requires 2 days
@endgantt
```



[Cм. QA-13971]

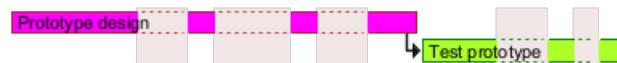
Или:

```
@startgantt
<style>
ganttDiagram {
    timeline {
        LineColor transparent
        FontColor transparent
    }
    closed {
        FontColor transparent
    }
}
</style>

hide footbox
project starts the 2018/04/09
saturday are closed
sunday are closed
2018/05/01 is closed
2018/04/17 to 2018/04/19 is closed
[Prototype design] requires 9 days
```



```
[Test prototype] requires 5 days
[Test prototype] starts at [Prototype design]'s end
[Prototype design] is colored in Fuchsia/FireBrick
[Test prototype] is colored in GreenYellow/Green
@endgantt
```



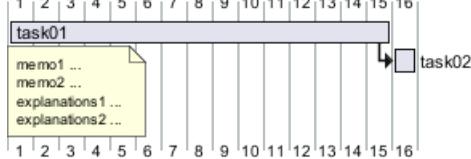
[Cм. QA-13464]

16.26 Добавление примечания

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

[task01] -> [task02]
```

@endgantt

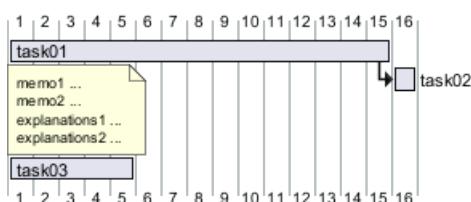


Пример с перекрытием.

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

[task01] -> [task02]
[task03] requires 5 days
```

@endgantt



@startgantt

-- test01 --



```
[task01] requires 4 days
```

```
note bottom
```

```
'note left
```

```
memo1 ...
```

```
memo2 ...
```

```
explanations1 ...
```

```
explanations2 ...
```

```
end note
```

```
[task02] requires 8 days
```

```
[task01] -> [task02]
```

```
note bottom
```

```
'note left
```

```
memo1 ...
```

```
memo2 ...
```

```
explanations1 ...
```

```
explanations2 ...
```

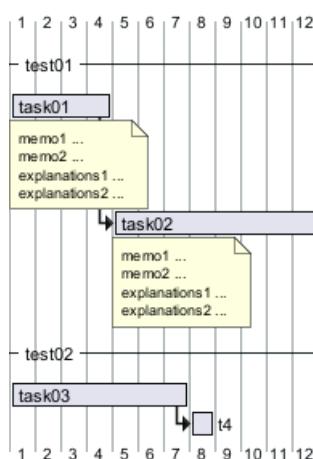
```
end note
```

```
-- test02 --
```

```
[task03] as [t3] requires 7 days
```

```
[t3] -> [t4]
```

```
@endgantt
```



TODO: DONE Спасибо за исправление (из #386 в v1.2020.18) при наложении

```
@startgantt
```

```
Project starts 2020-09-01
```

```
[taskA] starts 2020-09-01 and requires 3 days
```

```
[taskB] starts 2020-09-10 and requires 3 days
```

```
[taskB] displays on same row as [taskA]
```

```
[task01] starts 2020-09-05 and requires 4 days
```

```
then [task02] requires 8 days
```

```
note bottom
```

```
note for task02
```

```
more notes
```

```
end note
```

```
then [task03] requires 7 days
```

```
note bottom
```



```

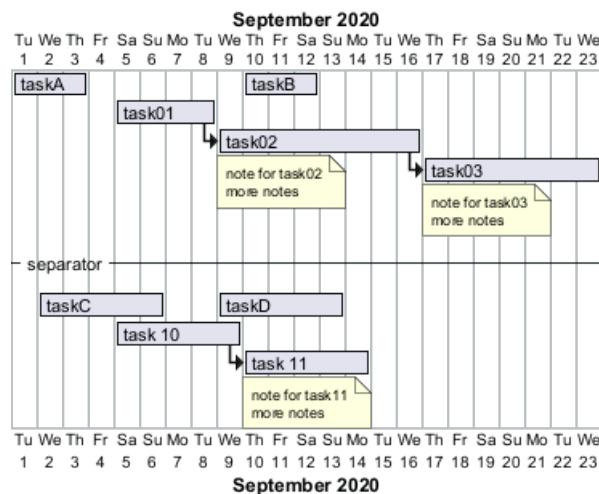
note for task03
more notes
end note

-- separator --

[taskC] starts 2020-09-02 and requires 5 days
[taskD] starts 2020-09-09 and requires 5 days
[taskD] displays on same row as [taskC]

[task 10] starts 2020-09-05 and requires 5 days
then [task 11] requires 5 days
note bottom
  note for task11
  more notes
end note
@endgantt

```



16.27 Приостановить выполнение задач

```

@startgantt
Project starts the 5th of december 2018
saturday are closed
sunday are closed
2018/12/29 is opened
[Prototype design] requires 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 requires 2 weeks
@endgantt

```



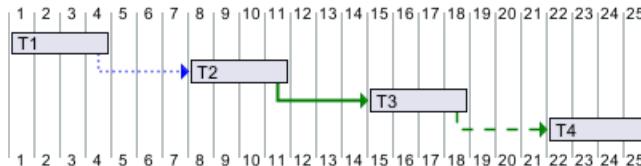
16.28 Изменение цвета связей между задачами

Вы можете менять цвета связей:



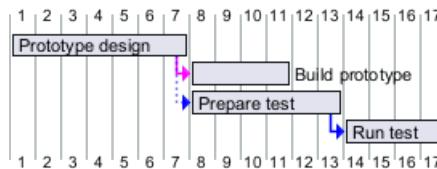
- при помощи синтаксиса: `with <color> <style> link`

```
@startgantt
[T1] requires 4 days
[T2] requires 4 days and starts 3 days after [T1]'s end with blue dotted link
[T3] requires 4 days and starts 3 days after [T2]'s end with green bold link
[T4] requires 4 days and starts 3 days after [T3]'s end with green dashed link
@endgantt
```



- или объявляя цвет в модуле `<style>`

```
@startgantt
<style>
ganttDiagram {
arrow {
LineColor blue
}
}
</style>
[Prototype design] requires 7 days
[Build prototype] requires 4 days
[Prepare test] requires 6 days
[Prototype design] -[#FF00FF]-> [Build prototype]
[Prototype design] -[dotted]-> [Prepare test]
Then [Run test] requires 4 days
@endgantt
```



[Ref. QA-13693]

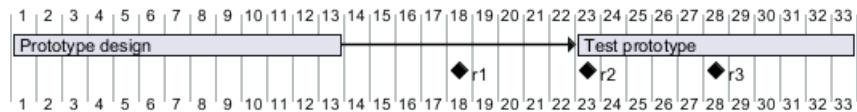
16.29 Задачи или вехи на одной линии

Задачи или вехи можно разместить на одной линии при помощи следующего синтаксиса:

- `[T|M] displays on same row as [T|M]`

```
@startgantt
[Prototype design] requires 13 days
[Test prototype] requires 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
```



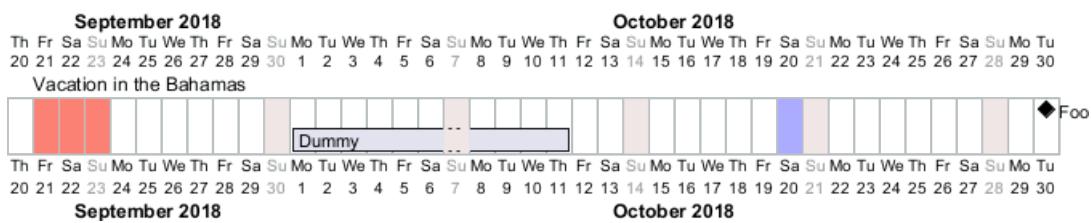


16.30 Выделение цветом дня "Сегодня"

```
@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]
```

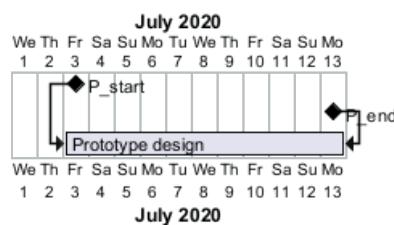
today is 30 days after start and is colored in #AAF
 [Foo] happens 40 days after start
 [Dummy] requires 10 days and starts 10 days after start

```
@endgantt
```



16.31 Задачи между двумя вехами

```
@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
```



16.32 Grammar and verbal form

Verbal form	Example
[T] starts	
[M] happens	

16.33 Заголовок, колонитулы, подписи, легенда

```
@startgantt
```

header

footer

title



```
[ ] requires 13 days
```

```
legend
```

```
end legend
```

```
caption
```

```
@endgantt
```



(See also: Common commands)

16.34 Изменение цвета легенды

```
@startgantt
[ ] requires 1 days and is colored in blue
then [ ] requires 5 days
[ ] requires 4 days
[ ] starts at [ ]'s end
[ ] is colored in Green
[ ] is colored in gray
```

```
legend
```

```
:
|= |= |
|<#gray> | |
|<#Green>| |
|<#blue> | |
end legend
```

```
@endgantt
```



[Ref. QA-19021]

16.35 Отключение подвала диаграммы (пример для всех масштабов)

Чтобы отключить подвал (нижнюю шкалу времени) диаграммы используйте команду `hide footbox` (справедливо также для диаграммы последовательностей).



Примеры для:

- Дневной шкалы (без команды начала проекта)

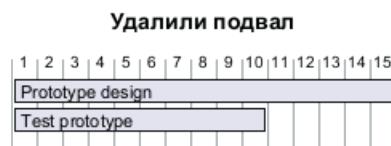
```
@startgantt
```

```
hide footbox
title
```

[Prototype design] requires 15 days

[Test prototype] requires 10 days

```
@endgantt
```



- дневной шкалы

```
@startgantt
```

Project starts the 20th of september 2017

[Prototype design] as [TASK1] requires 13 days

[TASK1] is colored in Lavender/LightBlue

```
hide footbox
@endgantt
```



- недельной шкалы

```
@startgantt
hide footbox
```

printscale weekly

saturday are closed

sunday are closed

Project starts the 1st of january 2021

[Prototype design end] as [TASK1] requires 19 days

[TASK1] is colored in Lavender/LightBlue

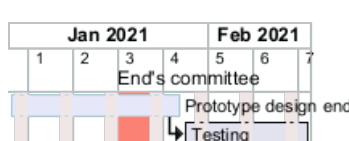
[Testing] requires 14 days

[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
```



- месячной шкалы

```
@startgantt
```

```
hide footbox
```



```
projectscale monthly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] requires 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] requires 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



- квартальной шкалы

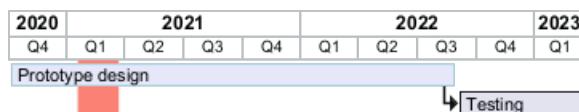
@startgantt

hide footbox

```
projectscale quarterly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] requires 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] requires 200 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-03-22 are colored in salmon

@endgantt



- годовой шкалы

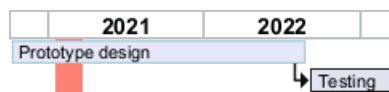
@startgantt

hide footbox

```
projectscale yearly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] requires 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] requires 200 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-03-22 are colored in salmon

@endgantt



16.36 Language of the calendar

You can choose the language of the Gantt calendar, with the `language <xx>` command where `<xx>` is the ISO 639 code of the language.

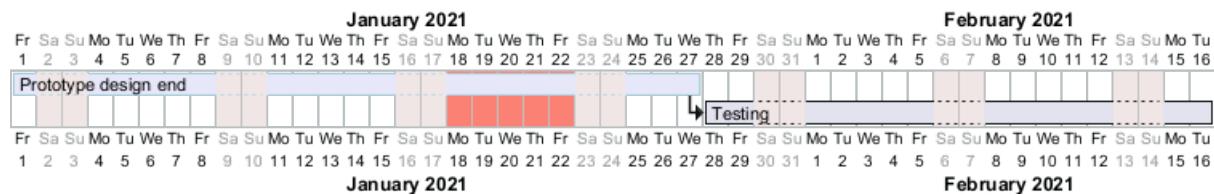


16.36.1 English (en, by default)

```
@startgantt
saturday are closed
sunday are closed
```

Project starts 2021-01-01
 [Prototype design end] as [TASK1] requires 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] requires 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt

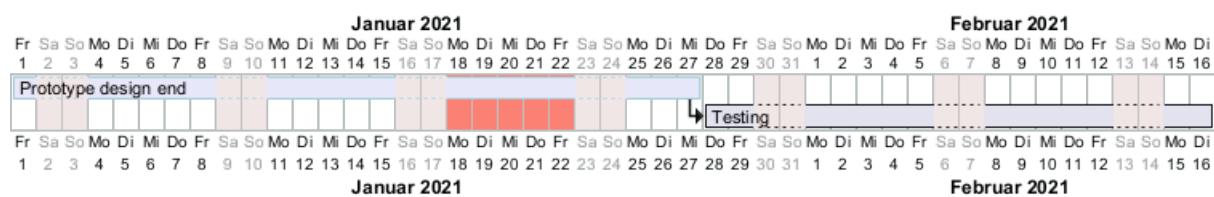


16.36.2 Deutsch (de)

```
@startgantt
language de
saturday are closed
sunday are closed
```

Project starts 2021-01-01
 [Prototype design end] as [TASK1] requires 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] requires 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



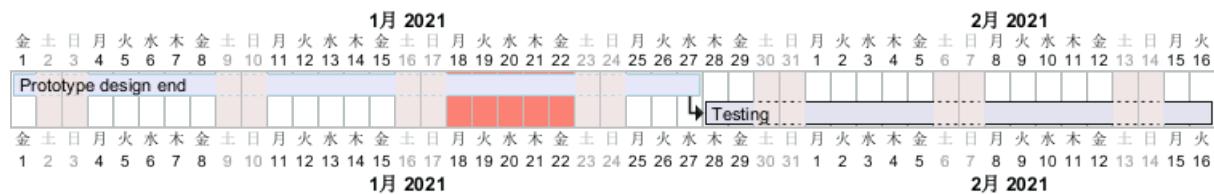
16.36.3 Japanese (ja)

```
@startgantt
language ja
saturday are closed
sunday are closed
```

Project starts 2021-01-01
 [Prototype design end] as [TASK1] requires 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] requires 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



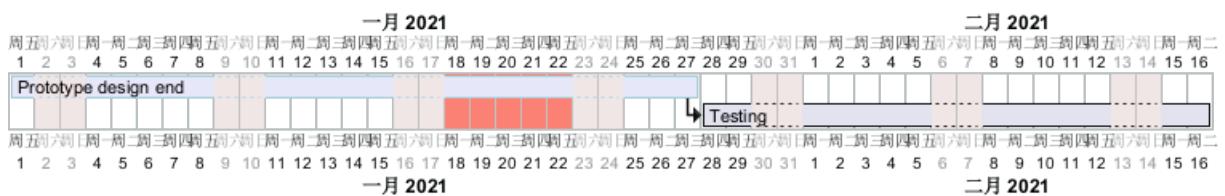


16.36.4 Chinese (zh)

```
@startgantt
language zh
saturday are closed
sunday are closed

Project starts 2021-01-01
[Prototype design end] as [TASK1] requires 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] requires 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```

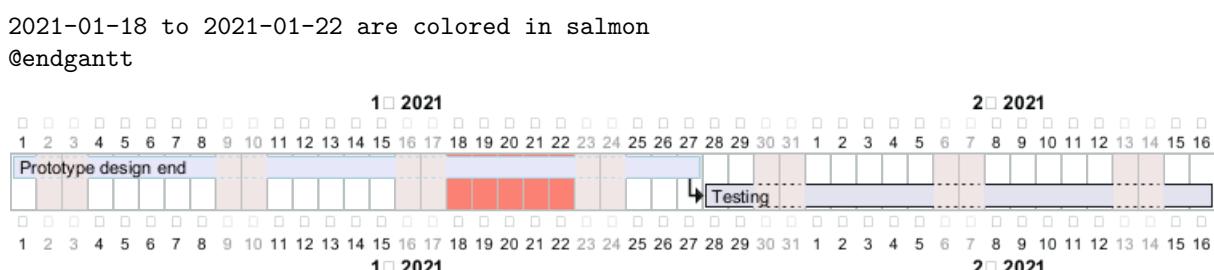


16.36.5 Korean (ko)

```
@startgantt
language ko
saturday are closed
sunday are closed

Project starts 2021-01-01
[Prototype design end] as [TASK1] requires 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] requires 14 days
[TASK1]->[Testing]

2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



16.37 Удаление задач или вех

Вы можете отметить задачу или веху удалённой при помощи конструкции `deleted` вместо стандартного обозначения завершёнными для того, чтобы отличать задачи от которых отказались, которые отложили или приняли по ним иное решение.



```
@startgantt
[ ] requires 1 weeks
then [ ] requires 4 days
[ ] happens at [ ]'s end
then [ ] requires 5 days
[ ] happens at [ ]'s end

[ ] is deleted
[ ] is deleted
@endgantt
```

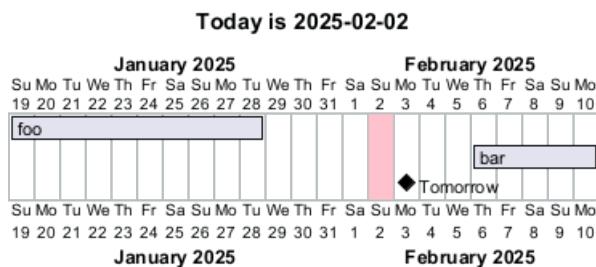


[Ref. QA-9129]

16.38 Отражение начала проекта, задачи или вехи за несколько дней до или после сегодняшнего дня

Чтобы начать проект, задачу или веху за несколько дней до или после сегодняшнего дня используйте встроенные функции %now и %date:

```
@startgantt
title Today is %date("YYYY-MM-dd")
!$now = %now()
!$past = %date("YYYY-MM-dd", $now - 14*24*3600)
Project starts $past
today is colored in pink
[foo] requires 10 days
[bar] requires 5 days and starts %date("YYYY-MM-dd", $now + 4*24*3600)
[Tomorrow] happens %date("YYYY-MM-dd", $now + 1*24*3600)
@endgantt
```



[Cм. QA-16285]

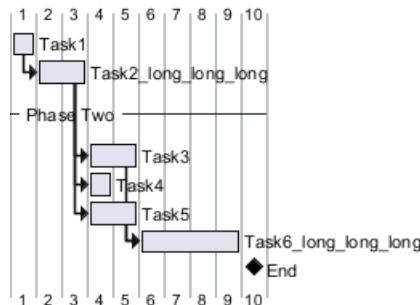
16.39 Изменение расположения ярлыка имени задачи

16.39.1 Ярлык имени расположен рядом с блоком задачи (настройка по умолчанию)

```
@startgantt
[Task1] requires 1 days
then [Task2_long_long_long] as [T2] requires 2 days
-- Phase Two --
then [Task3] as [T3] requires 2 days
[Task4] as [T4] requires 1 day
[Task5] as [T5] requires 2 days
[T2] -> [T4]
```



```
[T2] -> [T5]
[Task6_long_long_long] as [T6] requires 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```

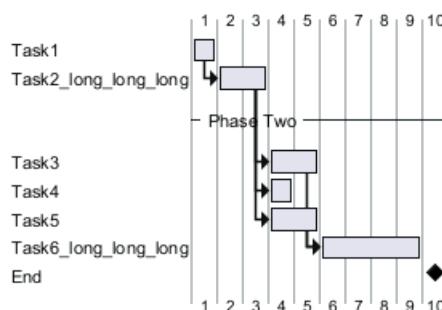


Чтобы изменить расположение ярлыка используйте команду label:

16.39.2 Ярлык в первой колонке

- Выключка влево

```
@startgantt
Label on first column and left aligned
[Task1] requires 1 days
then [Task2_long_long_long] as [T2] requires 2 days
-- Phase Two --
then [Task3] as [T3] requires 2 days
[Task4] as [T4] requires 1 day
[Task5] as [T5] requires 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] requires 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```

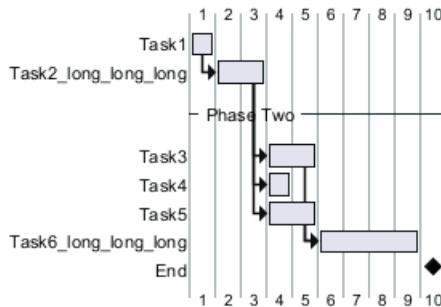


- Выключка вправо

```
@startgantt
Label on first column and right aligned
[Task1] requires 1 days
then [Task2_long_long_long] as [T2] requires 2 days
-- Phase Two --
then [Task3] as [T3] requires 2 days
[Task4] as [T4] requires 1 day
[Task5] as [T5] requires 2 days
[T2] -> [T4]
```



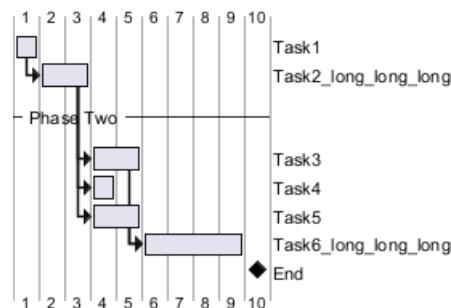
```
[T2] -> [T5]
[Task6_long_long_long] as [T6] requires 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```



16.39.3 Ярлык в последней колонке

- Выключка влево

```
@startgantt
Label on last column and left aligned
[Task1] requires 1 days
then [Task2_long_long_long] as [T2] requires 2 days
-- Phase Two --
then [Task3] as [T3] requires 2 days
[Task4] as [T4] requires 1 day
[Task5] as [T5] requires 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] requires 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```

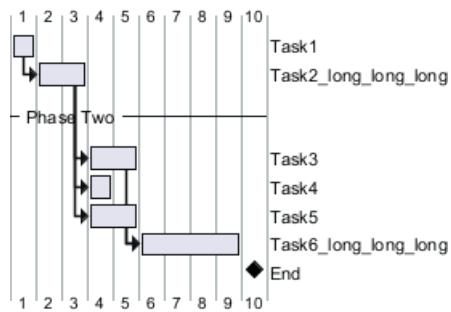


- Выключка вправо

```
@startgantt
Label on last column and right aligned
[Task1] requires 1 days
then [Task2_long_long_long] as [T2] requires 2 days
-- Phase Two --
then [Task3] as [T3] requires 2 days
[Task4] as [T4] requires 1 day
[Task5] as [T5] requires 2 days
[T2] -> [T4]
[T2] -> [T5]
```



```
[Task6_long_long_long] as [T6] requires 4 days  
[T3] -> [T6]  
[T5] -> [T6]  
[End] happens 1 day after [T6]'s end  
@endgantt
```



[Cм. QA-12433]



17 MindMap

Диаграмма **MindMap**, в контексте **PlantUML**, является эффективным инструментом для **проведения мозгового штурма**, организации идей и планирования проекта. MindMap-диаграммы, или карты мыслей, —это **визуальное представление** информации, в котором центральные идеи разветвляются на связанные темы, создавая паутину концепций. PlantUML облегчает создание таких диаграмм благодаря простому **текстовому синтаксису**, позволяющему эффективно организовывать и визуализировать сложные идеи.

Использование PlantUML для MindMap особенно выгодно благодаря его **интеграции с другими инструментами** и системами. Такая интеграция позволяет упростить процесс включения карт разума в общую проектную документацию. Текстовый подход PlantUML также позволяет легко модифицировать и **контролировать версии** карт, что делает их динамичным инструментом для совместного проведения мозговых штурмов и разработки идей.

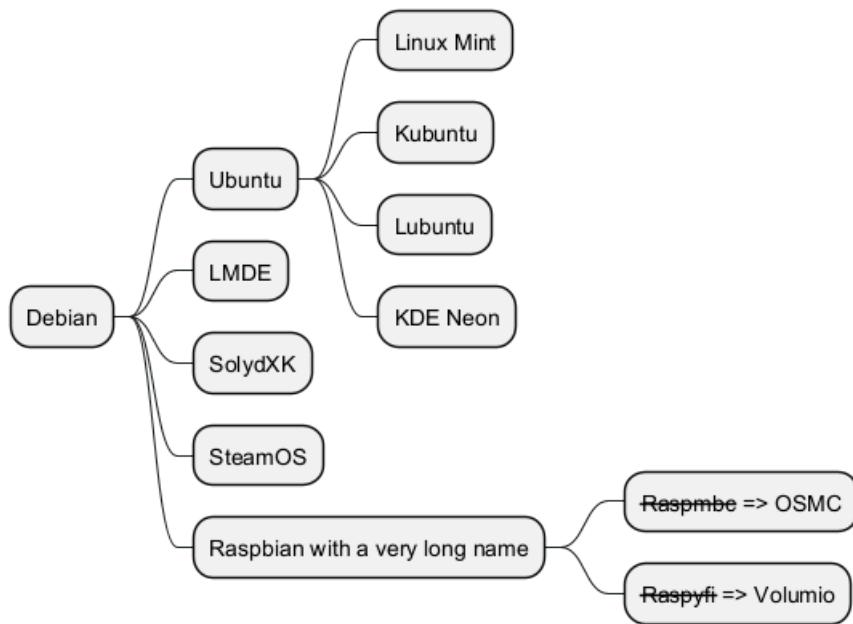
Карты разума в PlantUML могут использоваться для различных целей, от описания структуры проекта до мозгового штурма функций продукта или бизнес-стратегии. **Иерархическая и интуитивно понятная структура** карт помогает выявить взаимосвязи между различными идеями и концепциями, что облегчает восприятие общей картины и позволяет определить области, требующие дальнейшей проработки. Это делает PlantUML бесценным инструментом для руководителей проектов, разработчиков и бизнес-аналитиков, которым необходим метод визуальной организации и представления сложной информации в ясной и четкой форме.

17.1 Синтаксис OrgMode

Этот синтаксис совместим с OrgMode

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

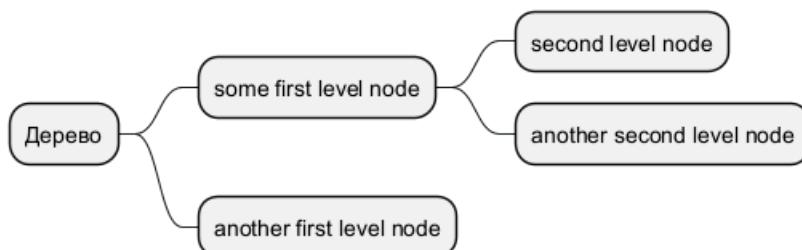




17.2 Markdown синтаксис

Этот синтаксис совместим с Markdown

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



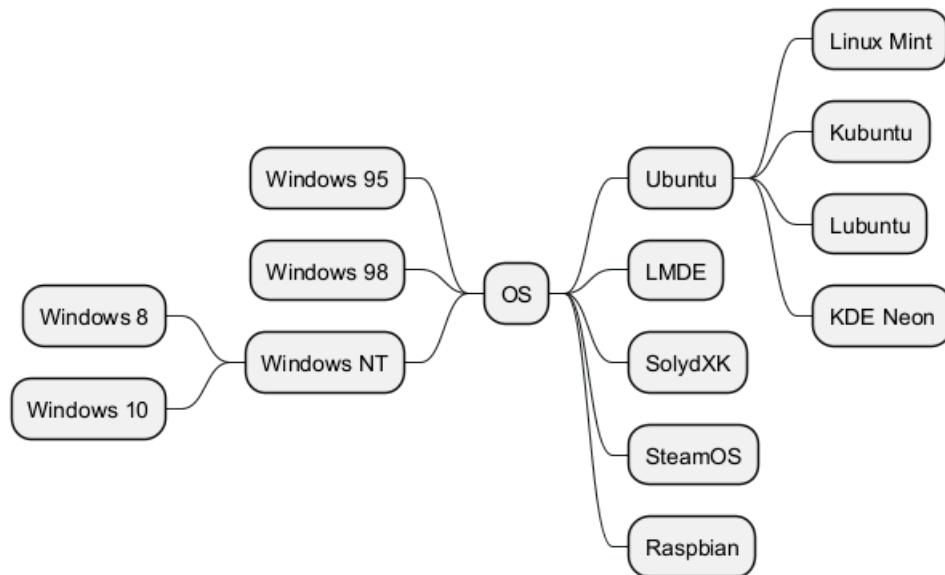
17.3 Арифметическая нотация

Вы можете использовать следующую нотацию для выбора направления ветвления.

```
@startmindmap
+ OS
++ Ubuntu
+++ Linux Mint
+++ Kubuntu
+++ Lubuntu
+++ KDE Neon
++ LMDE
++ SolydXK
++ SteamOS
++ Raspbian
```



```
-- Windows 95
-- Windows 98
-- Windows NT
--- Windows 8
---- Windows 10
@endmindmap
```

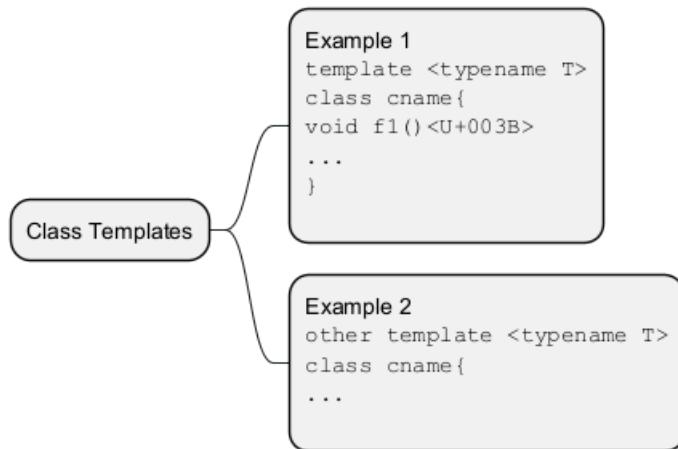


17.4 Многострочные блоки

Вы можете использовать символы : и ; для того, чтобы создавать многострочные блоки.

```
@startmindmap
* Class Templates
**:Example 1
<code>
template <typename T>
class cname{
void f1()<U+003B>
...
}
</code>
;
**:Example 2
<code>
other template <typename T>
class cname{
...
</code>
;
@endmindmap
```

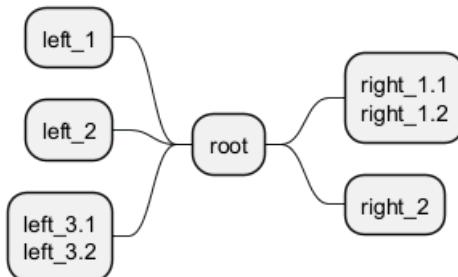




```
@startmindmap
+ root
**:right_1.1
right_1.2;
++ right_2
```

left side

```
-- left_1
-- left_2
**:left_3.1
left_3.2;
@endmindmap
```

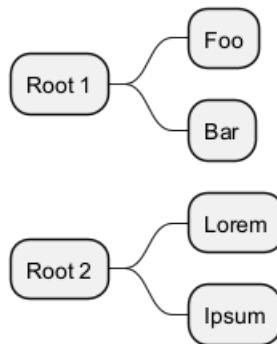


17.5 Диаграммы связей с несколькими исходными узлами

Вы можете создавать диаграммы с несколькими корневыми блоками следующим образом:

```
@startmindmap
* Root 1
** Foo
** Bar
* Root 2
** Lorem
** Ipsum
@endmindmap
```





[Ref. QH-773]

17.6 Цвета

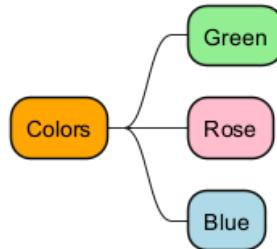
Можно задавать цвет узлов диаграммы.

17.6.1 Объявление настроек в строке блока

- Нотация в OrgMode

```

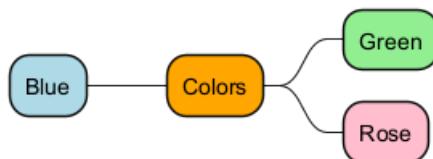
@startmindmap
*[#Orange] Colors
**[#lightgreen] Green
**[#FFBBCC] Rose
**[#lightblue] Blue
@endmindmap
  
```



- Арифметическая нотация

```

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



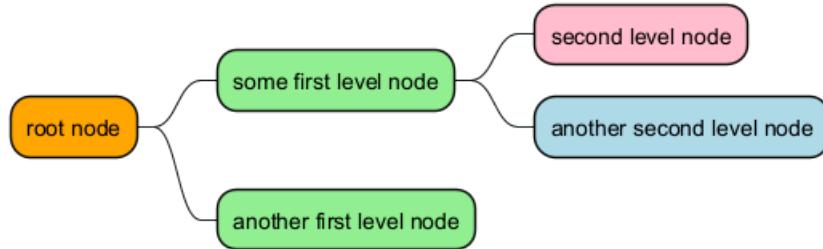
- Нотация в Markdown

```

@startmindmap
*[#Orange] root node
  
```



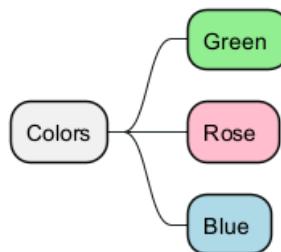
```
*[#lightgreen] some first level node
*[#FFBBCC] second level node
*[#lightblue] another second level node
*[#lightgreen] another first level node
@endmindmap
```



17.6.2 Объявление в блоке <style>

- Нотация OrgMode

```
@startmindmap
<style>
mindmapDiagram {
    .green {
        BackgroundColor lightgreen
    }
    .rose {
        BackgroundColor #FFBBCC
    }
    .your_style_name {
        BackgroundColor lightblue
    }
}
</style>
* Colors
** Green <<green>>
** Rose <<rose>>
** Blue <<your_style_name>>
@endmindmap
```



- Арифметическая нотация

```
@startmindmap
<style>
mindmapDiagram {
    .green {
        BackgroundColor lightgreen
    }
    .rose {

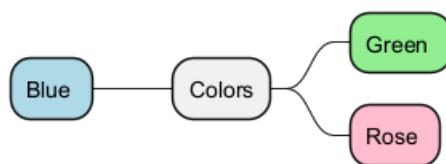
```



```

    BackgroundColor #FFBBCC
}
.your_style_name {
    BackgroundColor lightblue
}
}
</style>
+ Colors
++ Green <><green>>
++ Rose <><rose>>
-- Blue <><your_style_name>>
@endmindmap

```

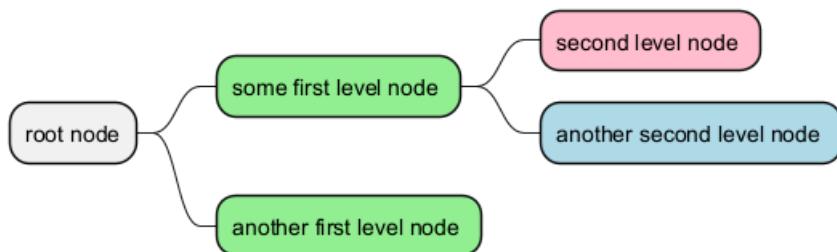


- Нотация в Markdown

```

@startmindmap
<style>
mindmapDiagram {
    .green {
        BackgroundColor lightgreen
    }
    .rose {
        BackgroundColor #FFBBCC
    }
    .your_style_name {
        BackgroundColor lightblue
    }
}
</style>
* root node
* some first level node <><green>>
    * second level node <><rose>>
    * another second level node <><your_style_name>>
* another first level node <><green>>
@endmindmap

```



- Применения стиля к одной ветви

```

@startmindmap
<style>
mindmapDiagram {
    .myStyle * {

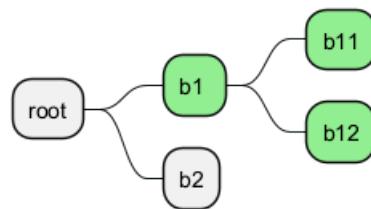
```



```

    BackgroundColor lightgreen
}
}
</style>
+ root
++ b1 <<myStyle>>
+++ b11
+++ b12
++ b2
@endmindmap

```



[Ref. GA-920]

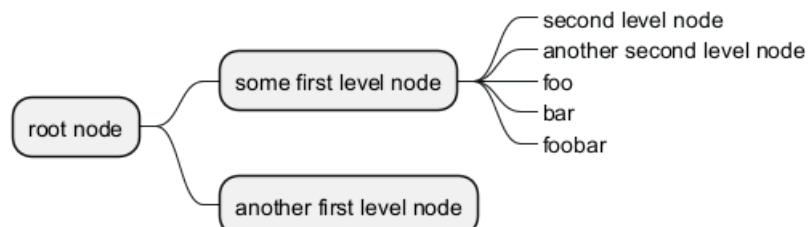
17.7 Оформление без рамки

Вы можете убрать рамку вокруг блока, используя символ подчёркивания.

```

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

```

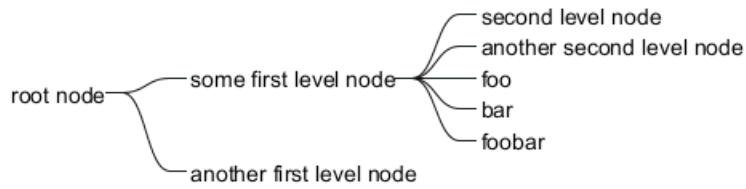


```

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

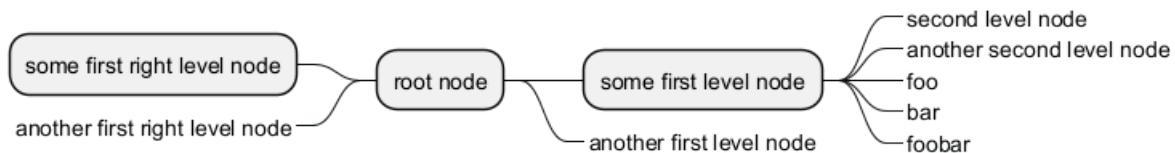
```





```

@startmindmap
+ root node
++ some first level node
+++ _ second level node
+++ _ another second level node
+++ _ foo
+++ _ bar
+++ _ foobar
++ _ another first level node
-- some first right level node
-- _ another first right level node
@endmindmap
  
```



17.8 Изменение направления диаграммы

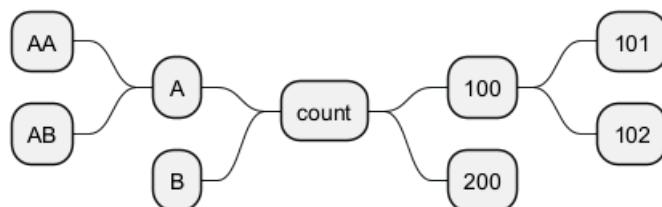
Можно использовать обе стороны диаграммы

```

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

left side

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



17.9 Изменение направления развёртки всей диаграммы

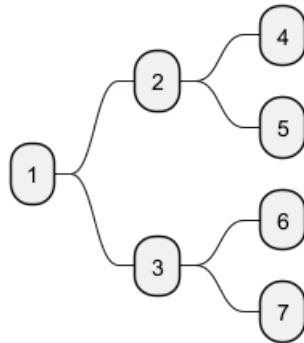
Команды чтобы изменить направление развёртки всей диаграммы:



- left to right direction () (*по умолчанию*)
- top to bottom direction ()
- right to left direction ()
- bottom to top direction () (*не реализовано/используйте обходные варианты*)

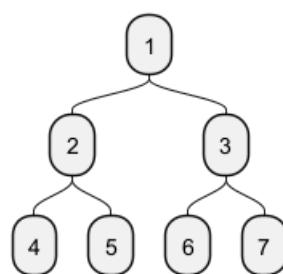
17.9.1 Слева направо (*настройка по умолчанию*)

```
@startmindmap
* 1
** 2
*** 4
*** 5
** 3
*** 6
*** 7
@endmindmap
```



17.9.2 Сверху вниз

```
@startmindmap
top to bottom direction
* 1
** 2
*** 4
*** 5
** 3
*** 6
*** 7
@endmindmap
```

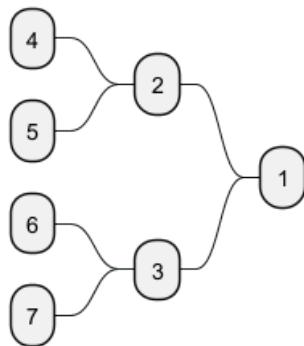


17.9.3 Справа налево

```
@startmindmap
right to left direction
```

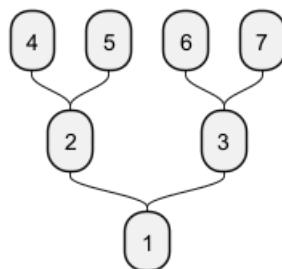


```
* 1
** 2
*** 4
*** 5
** 3
*** 6
*** 7
@endmindmap
```



17.9.4 Снизу вверх

```
@startmindmap
top to bottom direction
left side
* 1
** 2
*** 4
*** 5
** 3
*** 6
*** 7
@endmindmap
```



[Cм. QH-1413]

17.10 Полный пример

```
@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
```

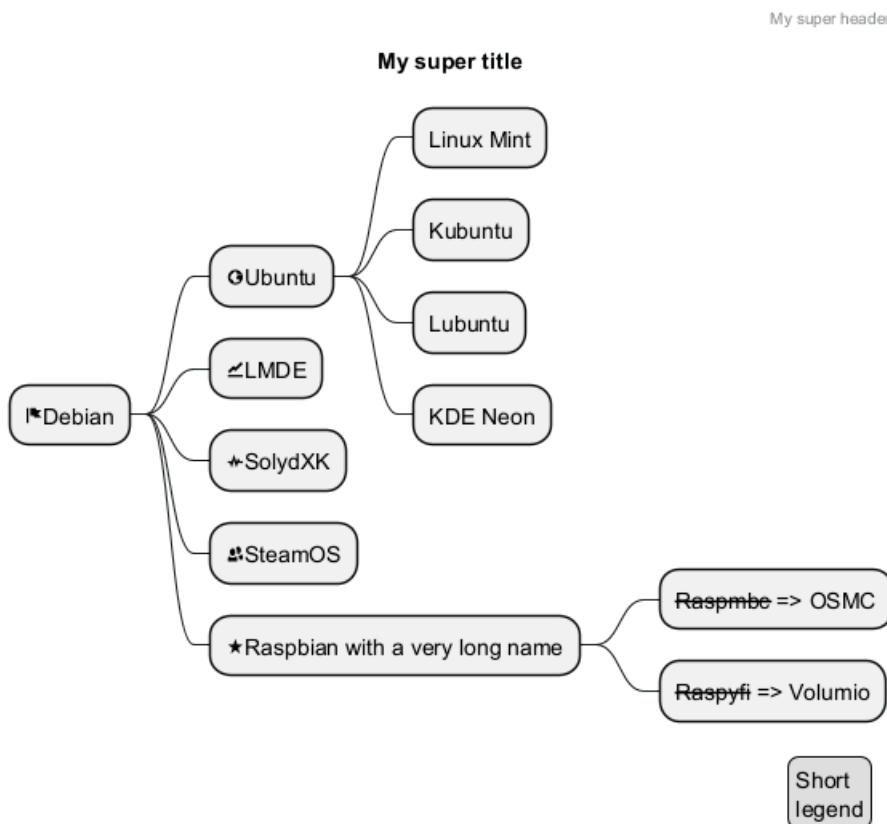


figure 1
My super footer

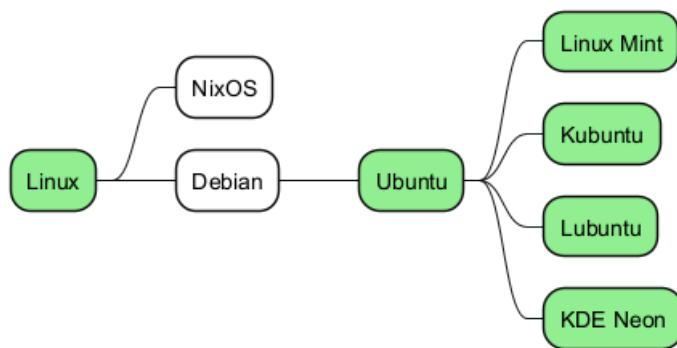
17.11 Стили диаграммы

17.11.1 блоки, глубина

```
@startmindmap
<style>
mindmapDiagram {
  node {
    BackgroundColor lightGreen
```



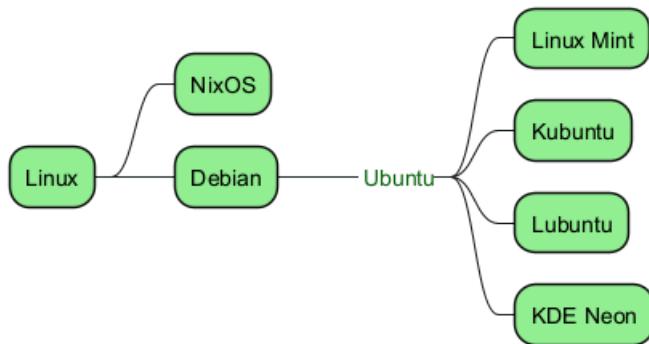
```
}
:depth(1) {
    BackGroundColor white
}
}
</style>
* Linux
** NixOS
** Debian
*** Ubuntu
**** Linux Mint
**** Kubuntu
**** Lubuntu
**** KDE Neon
@endmindmap
```



17.11.2 без рамки

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





17.12 Перенос строк

Используя команду `MaximumWidth` вы можете управлять автоматическим переносом строки. Единица измерения в команде – пиксель.

```
@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>

*      =)
**           ,
***          ,
  
```



```
**** , \n .
**** , PlantUML , , ,  

@endmindmap
```



17.13 Использования языка разметки Creole в диаграммах связей

Вы можете использовать Creole or HTML Creole в оформлении диаграмм:

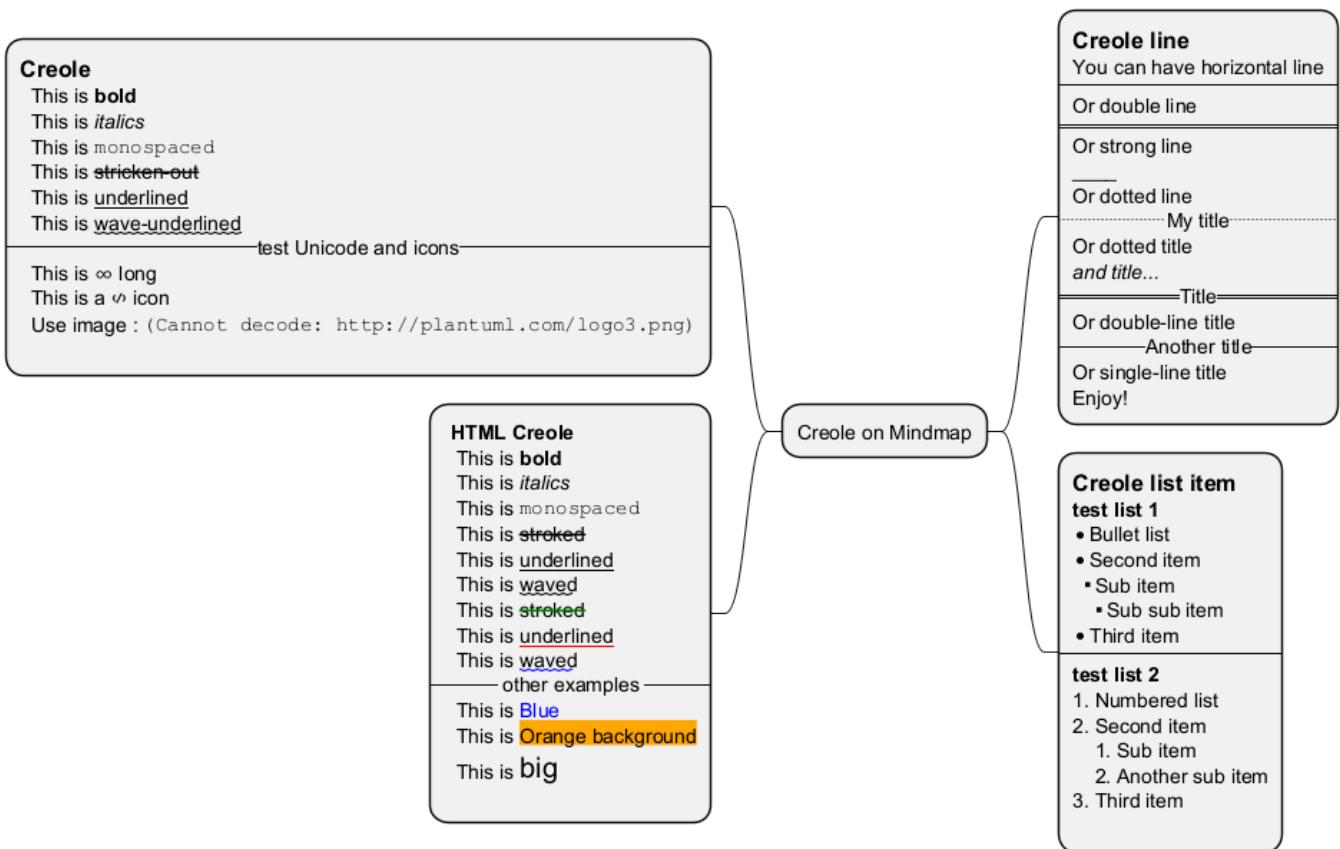
```
@startmindmap
* Creole on Mindmap
left side
**==Creole
This is **bold**
This is //italics//
This is "monospaced"
This is --stricken-out--
This is __underlined__
This is ~~wave-underlined~~
--test Unicode and icons--
This is <U+221E> long
This is a <&code> icon
Use image : <img: http://plantuml.com/logo3.png>
;

**: <b>HTML Creole
This is <b>bold</b>
This is <i>italics</i>
This is <font:monospaced>monospaced</font>
This is <s>stroked</s>
This is <u>underlined</u>
This is <w>waved</w>
This is <s:green>stroked</s>
This is <u:red>underlined</u>
This is <w:#0000FF>waved</w>
-- other examples --
This is <color:blue>Blue</color>
This is <back:orange>Orange background</back>
This is <size:20>big</size>
;
right side
**==Creole line
You can have horizontal line
----
```



```
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;
**==Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
;
@endmindmap
```





[Cm. QA-17838]



18 Структура разбиения работ (WBS)

Диаграмма структуры разбиения работ (WBS) - это ключевой **инструмент управления проектами**, позволяющий разбить проект на более мелкие и **управляемые компоненты** или задачи. По сути, это **иерархическая декомпозиция** общего объема работ, которые должны быть выполнены командой проекта для достижения целей проекта и создания необходимых результатов.

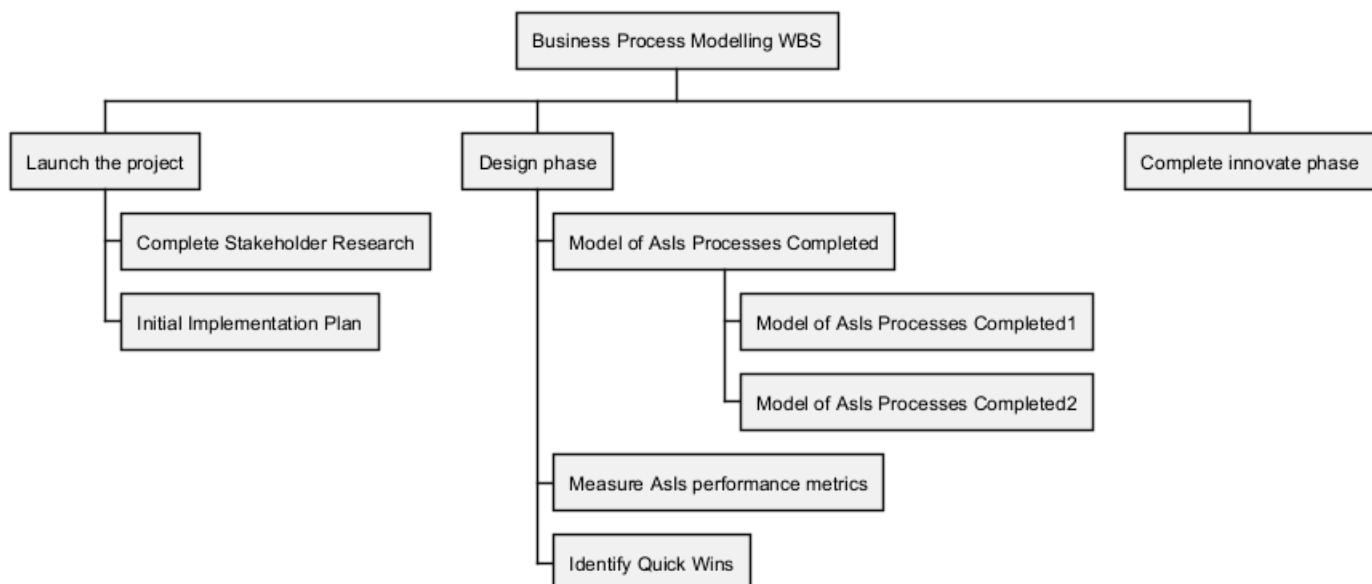
PlantUML может быть особенно полезен для создания **диаграмм WBS**. Благодаря **текстовому принципу построения диаграмм** создание и обновление WBS не сложнее редактирования текстового документа, что особенно удобно для управления изменениями в течение жизненного цикла проекта. Такой подход позволяет легко интегрировать **систему с системами контроля версий**, что обеспечивает отслеживание всех изменений и сохранение истории развития WBS.

Кроме того, совместимость PlantUML с различными инструментами повышает ее полезность в **среде совместной работы**. Команды могут легко интегрировать свои WBS-диаграммы в более широкую проектную документацию и системы управления. Простота синтаксиса PlantUML позволяет быстро вносить корректировки, что очень важно в **динамичных проектных средах**, где объем и задачи могут часто меняться. Таким образом, использование PlantUML для построения WBS-диаграмм сочетает в себе наглядность **визуальной разбивки** с гибкостью и управляемостью **текстовой системы**, что делает его ценным активом в **эффективном управлении проектами**.

18.1 Синтаксис OrgMode

Этот синтаксис совместим с OrgMode

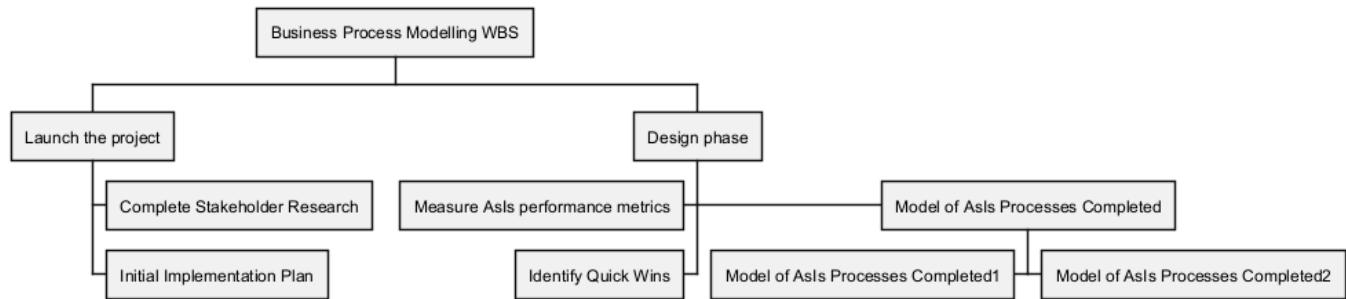
```
@startwbs
* Business Process Modelling WBS
** Launch the project
*** Complete Stakeholder Research
*** Initial Implementation Plan
** Design phase
*** Model of AsIs Processes Completed
**** Model of AsIs Processes Completed1
**** Model of AsIs Processes Completed2
*** Measure AsIs performance metrics
*** Identify Quick Wins
** Complete innovate phase
@endwbs
```



18.2 Change direction

You can change direction using < and >

```
@startwbs
* Business Process Modelling WBS
** Launch the project
*** Complete Stakeholder Research
*** Initial Implementation Plan
** Design phase
*** Model of AsIs Processes Completed
****< Model of AsIs Processes Completed1
****> Model of AsIs Processes Completed2
***< Measure AsIs performance metrics
***< Identify Quick Wins
@endwbs
```

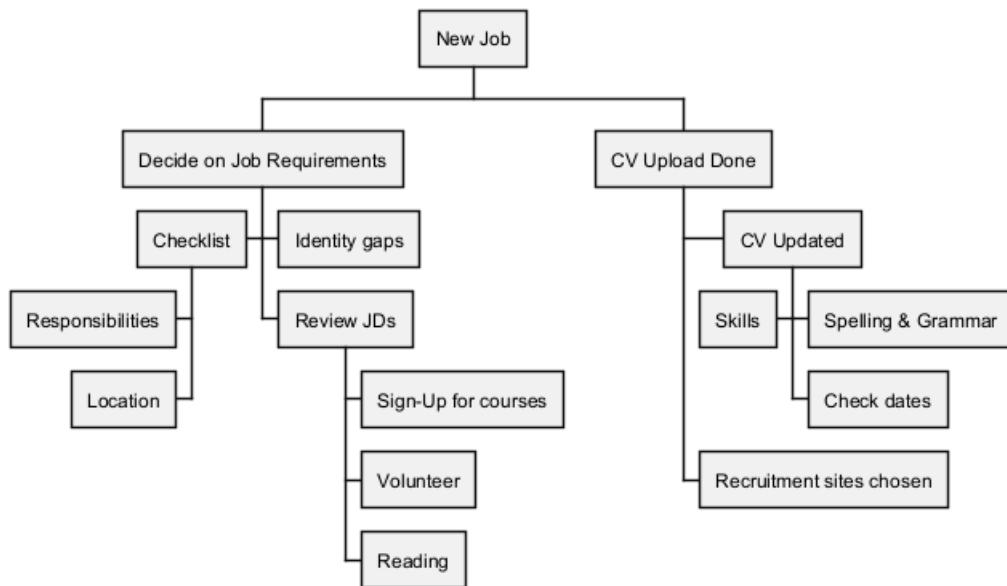


18.3 Arithmetic notation

You can use the following notation to choose diagram side.

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





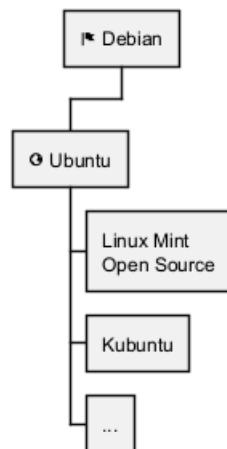
18.4 Multilines

You can use : and ; to have multilines box, as on MindMap.

```

@startwbs
* <&flag> Debian
** <&globe> Ubuntu
*** :Linux Mint
Open Source;

*** Kubuntu
*** ...
@endwbs
  
```



[Ref. QA-13945]

18.5 Removing box

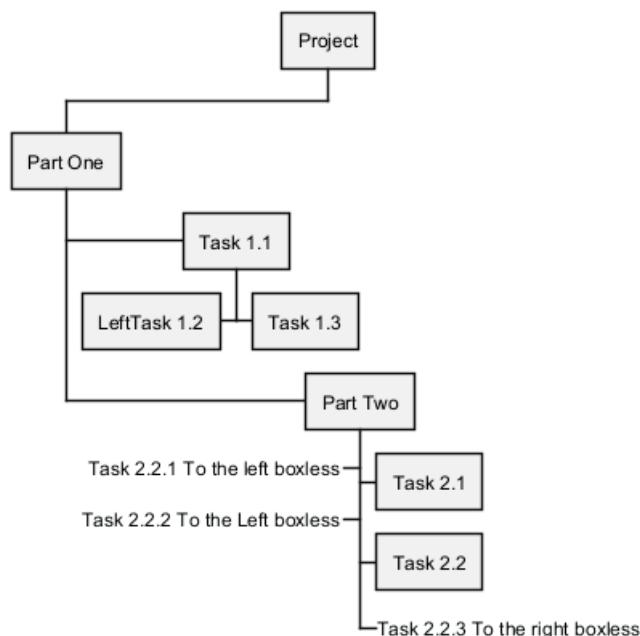
You can use underscore _ to remove box drawing.



18.5.1 Boxless on Arithmetic notation

18.5.2 Several boxless node

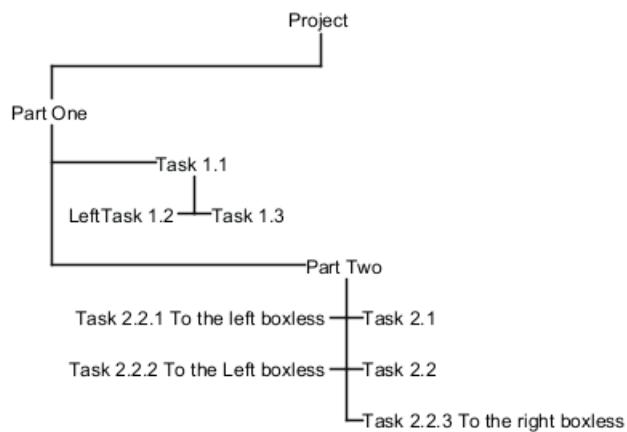
```
@startwbs
+ Project
+ Part One
+ Task 1.1
- LeftTask 1.2
+ Task 1.3
+ Part Two
+ Task 2.1
+ Task 2.2
-_ Task 2.2.1 To the left boxless
-_ Task 2.2.2 To the Left boxless
+_ Task 2.2.3 To the right boxless
@endwbs
```



18.5.3 All boxless node

```
@startwbs
+_ Project
+_ Part One
+_ Task 1.1
-_ LeftTask 1.2
+_ Task 1.3
+_ Part Two
+_ Task 2.1
+_ Task 2.2
-_ Task 2.2.1 To the left boxless
-_ Task 2.2.2 To the Left boxless
+_ Task 2.2.3 To the right boxless
@endwbs
```

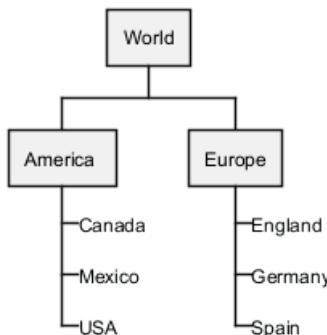




18.5.4 Boxless on OrgMode syntax

18.5.5 Several boxless node

```
@startwbs
* World
** America
*** _ Canada
*** _ Mexico
*** _ USA
** Europe
*** _ England
*** _ Germany
*** _ Spain
@endwbs
```

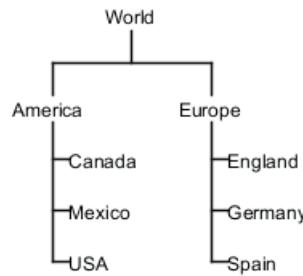


[Ref. QA-13297]

18.5.6 All boxless node

```
@startwbs
*_ World
**_ America
***_ Canada
***_ Mexico
***_ USA
**_ Europe
***_ England
***_ Germany
***_ Spain
@endwbs
```





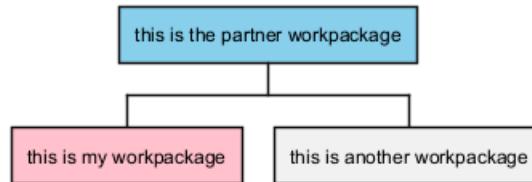
[Ref. QA-13355]

18.6 Colors (with inline or style color)

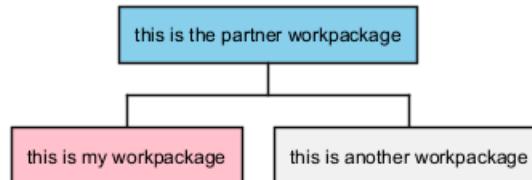
It is possible to change node color:

- with inline color

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



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



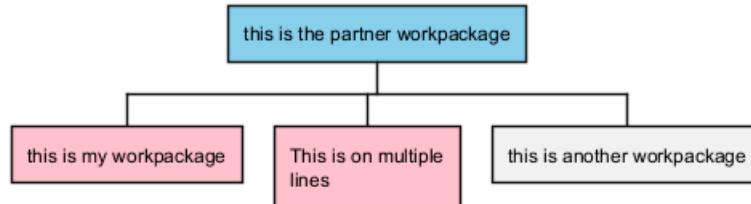
[Ref. QA-12374, only from v1.2020.20]

- with style color

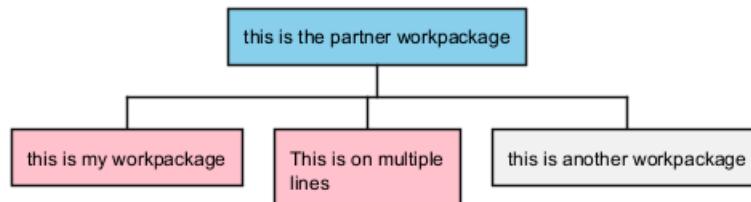
```
@startwbs
<style>
wbsDiagram {
    .pink {
        BackgroundColor pink
    }
    .your_style_name {
        BackgroundColor SkyBlue
    }
}
</style>
* this is the partner workpackage <<your_style_name>>
** this is my workpackage <<pink>>
```



```
**:This is on multiple
lines; <<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 on multiple
lines; <<pink>>
++ this is another workpackage
@endwbs
```



18.7 Using style

It is possible to change diagram style.

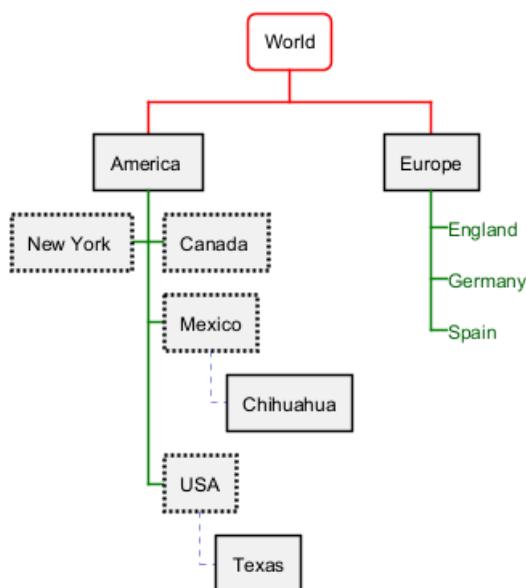
```
@startwbs
<style>
wbsDiagram {
    // all lines (meaning connector and borders, there are no other lines in WBS) are black by default
    Linecolor black
    arrow {
        // note that connector are actually "arrow" even if they don't look like as arrow
        // This is to be consistent with other UML diagrams. Not 100% sure that it's a good idea
        // So now connector are green
        LineColor green
    }
    :depth(0) {
        // will target root node
        BackgroundColor White
        RoundCorner 10
        LineColor red
        // Because we are targetting depth(0) for everything, border and connector for level 0 will be
```



```

}
arrow {
:depth(2) {
// Targetting only connector between Mexico-Chihuahua and USA-Texas
LineColor blue
LineStyle 4
LineThickness .5
}
}
node {
:depth(2) {
LineStyle 2
LineThickness 2.5
}
}
boxless {
// will target boxless node with '_'
FontColor darkgreen
}
}
</style>
* World
** America
*** Canada
*** Mexico
**** Chihuahua
*** USA
**** Texas
***< New York
** Europe
***_ England
***_ Germany
***_ Spain
@endwbs

```



18.8 Word Wrap

Using `MaximumWidth` setting you can control automatic word wrap. Unit used is pixel.



```
@startwbs
```

```
<style>
node {
    Padding 12
    Margin 3
    HorizontalAlignment center
    LineColor blue
    LineThickness 3.0
    BackgroundColor gold
    RoundCorner 40
    MaximumWidth 100
}

rootNode {
    LineStyle 8.0;3.0
    LineColor red
    BackgroundColor white
    LineThickness 1.0
    RoundCorner 0
    Shadowing 0.0
}

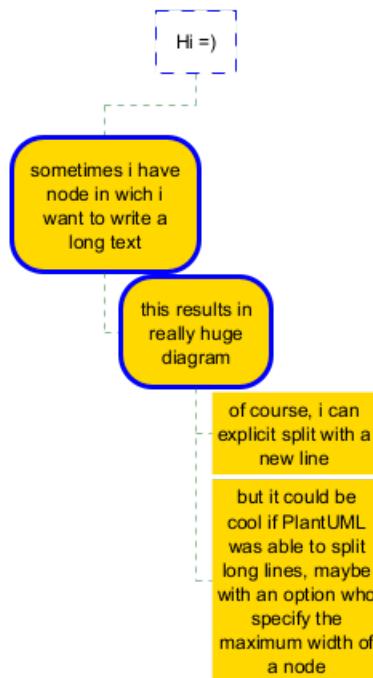
leafNode {
    LineColor gold
    RoundCorner 0
    Padding 3
}

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

* Hi =)
** sometimes i have node in which i want to write a long text
*** this results in really huge diagram
**** of course, i can explicitly split with a\nnew line
**** but it could be cool if PlantUML was able to split long lines, maybe with an option who specify
```

```
@endwbs
```



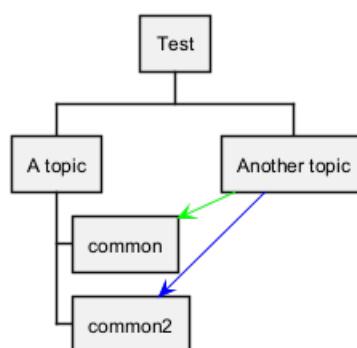


18.9 Add arrows between WBS elements

You can add arrows between WBS elements.

Using alias with as:

```
@startwbs
<style>
.foo {
    LineColor #00FF00;
}
</style>
* Test
** A topic
*** "common" as c1
*** "common2" as c2
** "Another topic" as t2
t2 -> c1 <<foo>>
t2 ..> c2 #blue
@endwbs
```

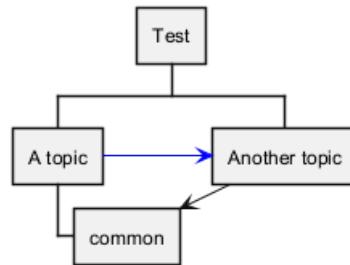


Using alias in parentheses:

```
@startwbs
* Test
```



```
**(b) A topic
***c1 common
**(t2) Another topic
t2 --> c1
b -> t2 #blue
@endwbs
```



[Ref. QA-16251]

18.10 Creole on WBS diagram

You can use Creole or HTML Creole on WBS:

```
@startwbs
* Creole on WBS
**==Creole
This is **bold**
This is //italics//
This is ""monospaced"""
This is --stricken-out--
This is __underlined__
This is ~~wave-underlined~~
--test Unicode and icons--
This is <U+221E> long
This is a <&code> icon
Use image : <img:https://plantuml.com/logo3.png>
;
**: <b>HTML Creole
This is <b>bold</b>
This is <i>italics</i>
This is <font:monospaced>monospaced</font>
This is <s>stroked</s>
This is <u>underlined</u>
This is <w>waved</w>
This is <s:green>stroked</s>
This is <u:red>underlined</u>
This is <w:#0000FF>waved</w>
-- other examples --
This is <color:blue>Blue</color>
This is <back:orange>Orange background</back>
This is <size:20>big</size>
;
**==Creole line
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
```

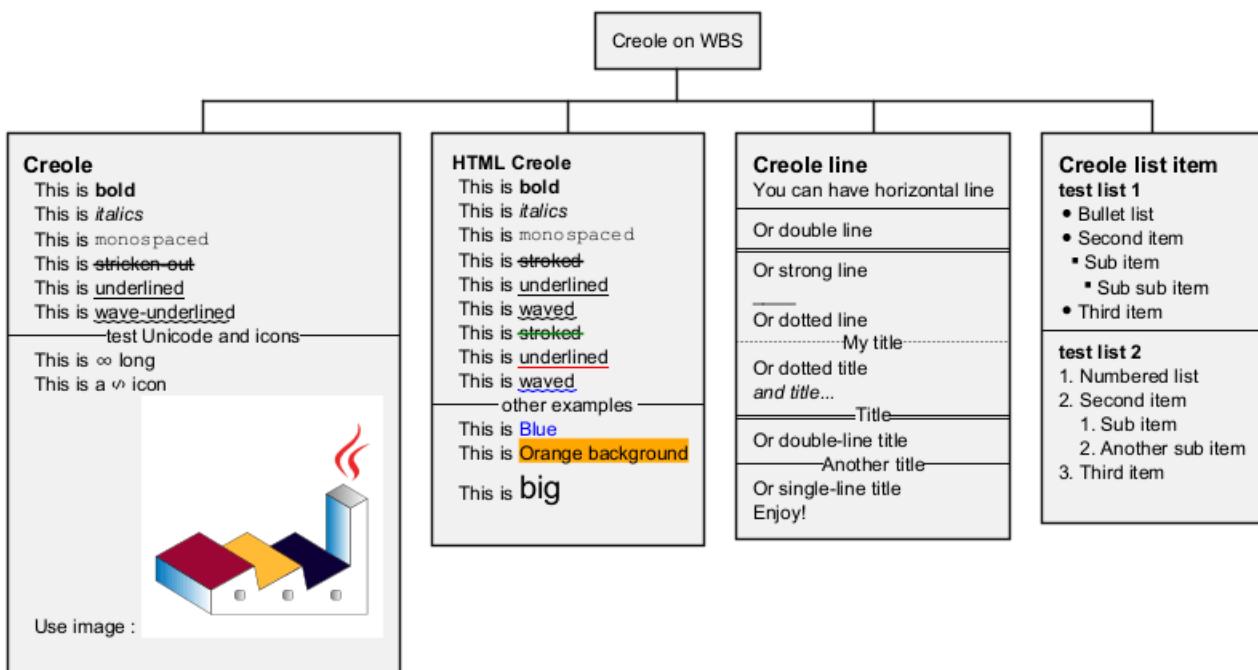


```

..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;

**==Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
;
@endwbs

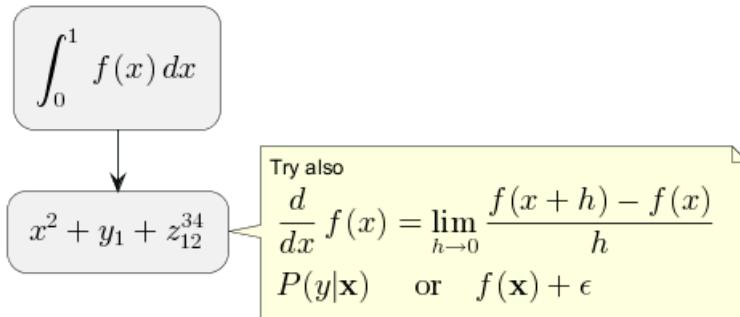
```



19 Maths

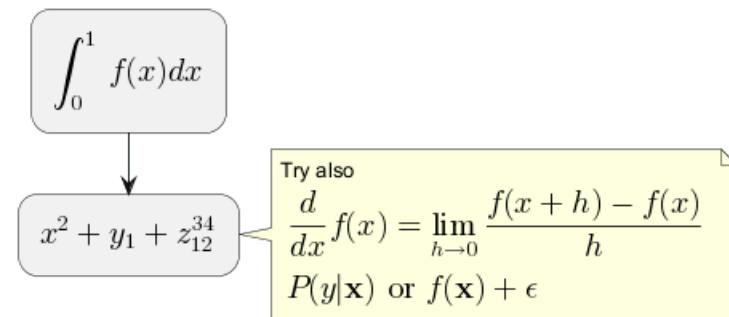
Within PlantUML, you can use AsciiMath notation:

```
@startuml
:<math>\int_0^1 f(x) dx</math>;
:<math>x^2+y_1+z_{12}^{34}</math>;
note right
Try also
<math>d/dx f(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}</math>
<math>P(y|x)</math> or <math>f(\mathbf{x}) + \epsilon</math>
end note
@enduml
```



or JLaTeXMath notation:

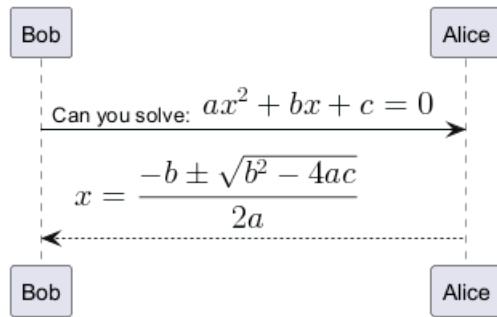
```
@startuml
:<math>\int_0^1 f(x) dx</math>;
:<math>x^2+y_1+z_{12}^{34}</math>;
note right
Try also
<math>\frac{d}{dx} f(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}</math>
<math>P(y|\mathbf{x})</math> or <math>f(\mathbf{x}) + \epsilon</math>
end note
@enduml
```



Here is another example:

```
@startuml
Bob -> Alice : Can you solve: <math>ax^2+bx+c=0</math>
Alice --> Bob: <math>x = (-b \pm \sqrt{b^2-4ac})/(2a)</math>
@enduml
```





19.1 Standalone diagram

You can also use @startmath/@endmath to create standalone AsciiMath formula.

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

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

Or use @startlatex/@endlatex to create standalone JLaTeXMath formula.

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

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

19.2 How is this working?

To draw those formulas, PlantUML uses two open source projects:

- AsciiMath that converts AsciiMath notation to LaTeX expression;
- JLatexMath that displays mathematical formulas written in LaTeX. JLaTeXMath is the best Java library to display LaTeX code.

ASCIIIMathTeXImg.js is small enough to be integrated into PlantUML standard distribution.

And since 2021 (V1.2021.8), ASCIIIMathTeXImg.js was ported on JAVA in PlantUML with ASCIIIMathTeXImg.java and since 2024 (V1.2024.5) there was some more corrections and improvements (*with the The-Lum/ASCIIIMathTeXImg Project*).

Since JLatexMath is bigger, you have to download it separately, then unzip the 4 jar files (*batik-all-1.7.jar*, *jlatexmath-minimal-1.0.3.jar*, *jlm_cyrillic.jar* and *jlm_greek.jar*) in the same folder as PlantUML.jar.



20 Entity Relationship Diagram

Основана на нотации Information Engineering.

Является расширением существующей диаграммы классов. Это расширение добавляет:

- Дополнительные отношения для нотации Information Engineering.
- Псевдоним `entity`, который сопоставляется с диаграммой классов `class`.
- Дополнительный модификатор видимости * для идентификации обязательных атрибутов.

В остальном синтаксис построения диаграмм такой же, как и для диаграмм классов. Все остальные возможности диаграмм классов также поддерживаются.

See also Chen [Entity Relationship Diagrams](er-diagram).

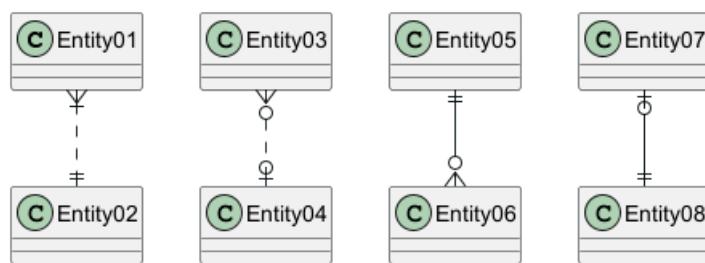
[Ref. [GH-31](<https://github.com/plantuml/plantuml/pull/31>)]

20.1 Information Engineering Relations

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

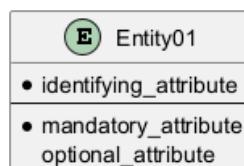
Examples:

```
@startuml
Entity01 }|...|| Entity02
Entity03 }o..o| Entity04
Entity05 ||--o{ Entity06
Entity07 |o--|| Entity08
@enduml
```



20.2 Entities

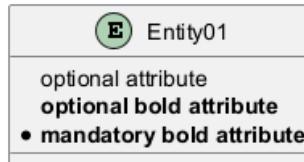
```
@startuml
entity Entity01 {
    * identifying_attribute
    --
    * mandatory_attribute
    optional_attribute
}
@enduml
```



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 Полный пример

```
@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
}

entity "Entity04" as e04 {
    *e4_id : number <<generated>>
    --
    e1_id : number <<FK>>
    other_details : text
}

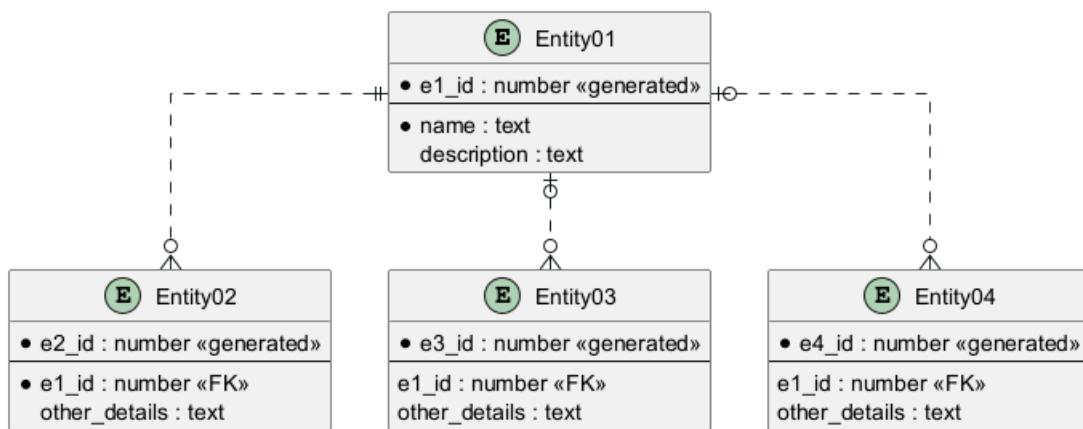
e01 ||..of e02
```



```
e01 |o..o{ e03
```

```
e01 |o..o{ e04
```

```
@enduml
```



В настоящее время вороньи лапки выглядят не очень хорошо, когда отношения нарисованы под углом к сущности. Этого можно избежать, используя параметр `linetype ortho` `skinparam`.



21 Общие команды в PlantUML

Откройте для себя основные команды, универсальные для всех типов диаграмм в PlantUML. Эти команды позволяют придать диаграммам многогранность и индивидуальность. Ниже мы разделим эти общие команды на три основные категории:

21.0.1 Глобальные элементы

- Комментарии:** Добавляют примечания или пояснения в сценарий диаграммы для передачи дополнительной информации или напоминания о необходимости дальнейших изменений.
- Примечания:** Включение дополнительной информации непосредственно в диаграмму для облегчения понимания или выделения важных аспектов.
- Управление размерами:** Настройте размеры различных элементов в соответствии с вашими предпочтениями, чтобы диаграмма получилась сбалансированной и пропорциональной.
- Заголовок и подписи:** Определите подходящий заголовок и добавьте подписи, чтобы прояснить контекст или аннотировать определенные части диаграммы.

21.0.2 Описание синтаксиса Creole

Используйте возможности синтаксиса Creole для дальнейшего форматирования содержимого любого элемента диаграммы. Этот стиль вики-разметки позволяет:

- Форматирование текста:** Настраивать внешний вид текста с помощью различных стилей и выравниваний.
- Списки:** Создавать упорядоченные и неупорядоченные списки для удобного представления информации.
- Ссылки:** Встраивать гиперссылки для быстрого перехода к нужным ресурсам.

21.0.3 Команда управления стилями

С помощью команды `style` можно полностью контролировать стиль представления элементов диаграммы. С ее помощью можно:

- Определить стили:** Задать единые стили для элементов, чтобы сохранить целостность визуальной темы.
- Настроить цвета:** Выбор определенных цветов для различных элементов для повышения визуальной привлекательности и создания четкой классификации.

Используйте эти команды для создания функциональных и эстетически привлекательных диаграмм, настраивая каждый элемент в соответствии с вашими требованиями.

21.1 Комментарии

21.1.1 Простой комментарий

Все записи начинающиеся с "одинарной кавычки" - это комментарии.

```
@startuml
'
@enduml
```

21.1.2 Block comment

Block comment use C-style comments except that instead of * you use an apostrophe ', then you can also put comments on several lines using /' to start and '/ to end.

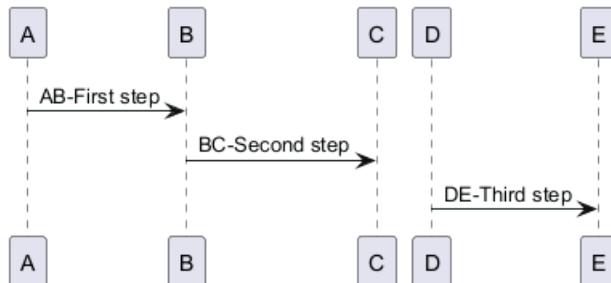
```
@startuml
'/>
many lines comments
here
'/
@enduml
```



[Ref. QA-1353]

Then you can also put block comment on the same line, as:

```
@startuml
/* case 1 */ A -> B : AB-First step
                    B -> C : BC-Second step
/* case 2 */ D -> E : DE-Third step
@enduml
```



[Ref. QA-3906 and QA-3910]

[Ref. GH-214]

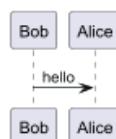
21.2 Zoom

You can use the `scale` command to zoom the generated image.

You can use either *a number* or *a fraction* to define the scale factor. You can also specify either `width` or `height` (*in pixel*). And you can also give both `width` and `height`: the image is scaled to fit inside the specified dimension.

- `scale 1.5`
- `scale 2/3`
- `scale 200 width`
- `scale 200 height`
- `scale 200*100`
- `scale max 300*200`
- `scale max 1024 width`
- `scale max 800 height`

```
@startuml
scale 180*90
Bob->Alice : hello
@enduml
```



21.3 Title

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

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

```
@startuml
skinparam titleBorderRoundCorner 15
skinparam titleBorderThickness 2
```



```

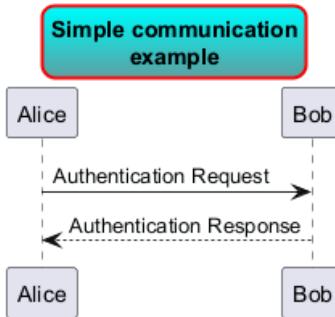
skinparam titleBorderColor red
skinparam titleBackgroundColor Aqua-CadetBlue

title Simple communication\nexample

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

@enduml

```



You can use creole formatting in the title.

You can also define title on several lines using title and end_title keywords.

```

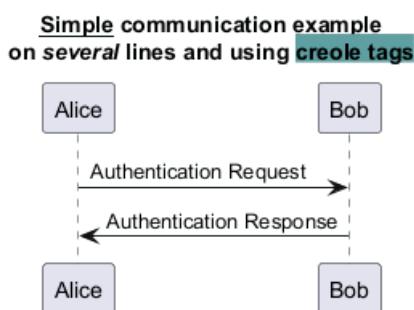
@startuml

title
<u>Simple</u> communication example
on <i>several</i> lines and using <back:cadetblue>creole tags</back>
end title

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

@enduml

```



21.4 Caption

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

```

@startuml

caption figure 1
Alice -> Bob: Hello

@enduml

```



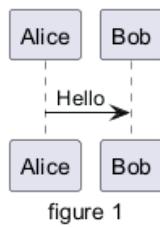


figure 1

21.5 Footer and header

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

You can optionally specify if you want a `center`, `left` or `right` footer/header, by adding a keyword.

As with title, it is possible to define a header or a footer on several lines.

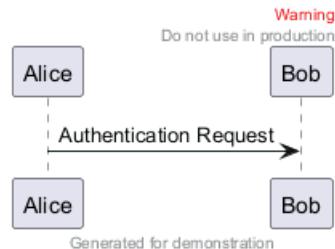
It is also possible to put some HTML into the header or footer.

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

```
header
<font color=red>Warning:</font>
Do not use in production.
endheader
```

```
center footer Generated for demonstration
```

```
@enduml
```



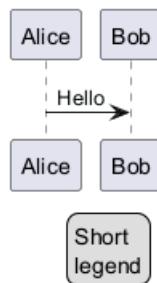
21.6 Legend the diagram

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

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

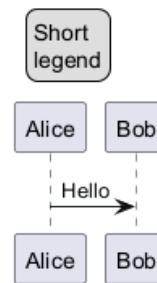
```
@startuml
Alice -> Bob : Hello
legend right
    Short
    legend
endlegend
@enduml
```





```

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



21.7 Appendix: Examples on all diagram

21.7.1 Activity

```

@startuml
header some header

footer some footer

title My title

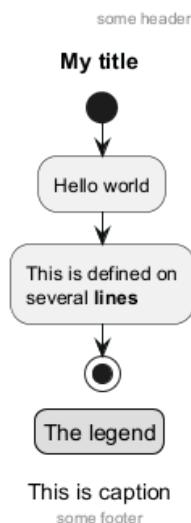
caption This is caption

legend
The legend
end legend

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

@enduml
  
```





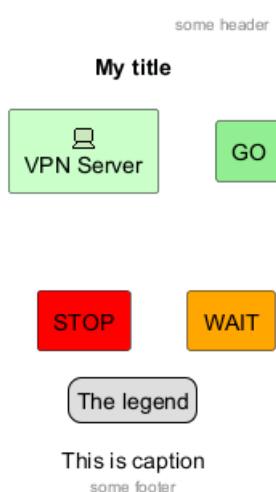
21.7.2 Archimate

```
@startuml
header some header
footer some footer
title My title
caption This is caption
legend
The legend
end legend

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange

@enduml
```



21.7.3 Class

```
@startuml
header some header

footer some footer

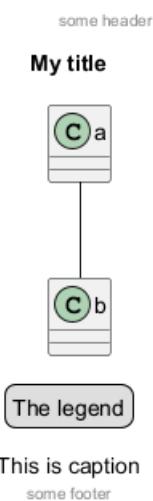
title My title

caption This is caption

legend
The legend
end legend
```

a -- b

@enduml



This is caption
some footer

21.7.4 Component, Deployment, Use-Case

```
@startuml
header some header

footer some footer

title My title

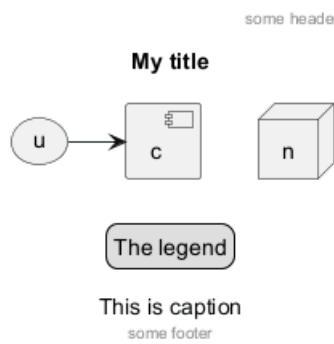
caption This is caption
```

```
legend
The legend
end legend
```

```
node n
(u) -> [c]
```

@enduml





21.7.5 Gantt project planning

```

@startgantt
header some header
footer some footer
title My title
caption This is caption
legend
The legend
end legend

```

[t] lasts 5 days

@endgantt

some header

My title



The legend

This is caption

some footer

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

21.7.6 Object

```

@startuml
header some header
footer some footer
title My title
caption This is caption
legend
The legend
end legend

```

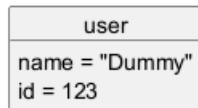


```
object user {
    name = "Dummy"
    id = 123
}
```

```
@enduml
```

some header

My title



The legend

This is caption

some footer

21.7.7 MindMap

```
@startmindmap
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

```
legend
```

```
The legend
```

```
end legend
```

```
* r
```

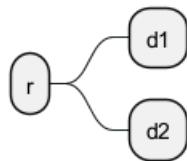
```
** d1
```

```
** d2
```

```
@endmindmap
```

some header

My title



The legend

This is caption

some footer

21.7.8 Network (nwdiag)

```
@startuml
header some header
```



```

footer some footer

title My title

caption This is caption

```

```

legend
The legend
end legend

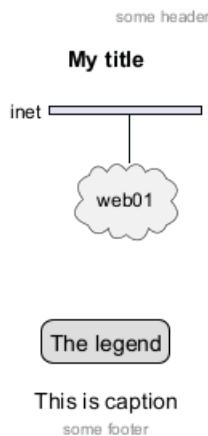
```

```

nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}

```

```
@enduml
```



21.7.9 Sequence

```

@startuml
header some header

footer some footer

title My title

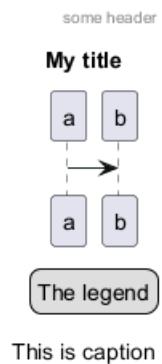
caption This is caption

legend
The legend
end legend

a->b
@enduml

```





This is caption
 some footer

21.7.10 State

```
@startuml
header some header

footer some footer

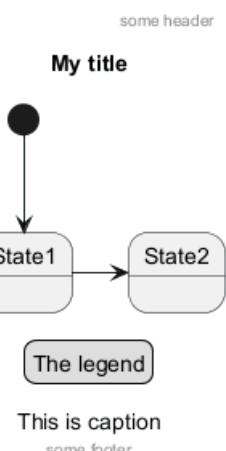
title My title

caption This is caption

legend
The legend
end legend

[*] --> State1
State1 -> State2

@enduml
```



This is caption
 some footer

21.7.11 Timing

```
@startuml
header some header

footer some footer

title My title

caption This is caption
```



```

legend
The legend
end legend

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

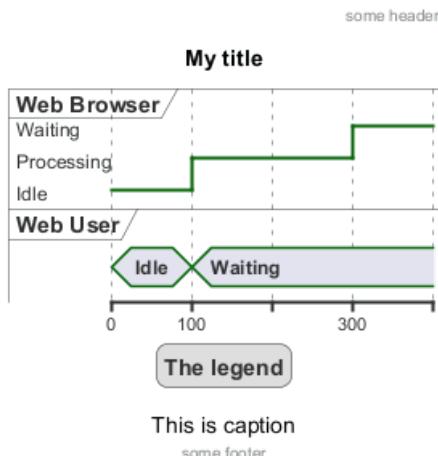
@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting

@enduml

```



21.7.12 Work Breakdown Structure (WBS)

```

@startwbs
header some header

footer some footer

title My title

caption This is caption

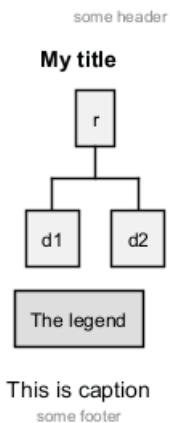
legend
The legend
end legend

* r
** d1
** d2

@endwbs

```





TODO: DONE [*Corrected on V1.2020.17*]

21.7.13 Wireframe (SALT)

```

@startsalt
header some header

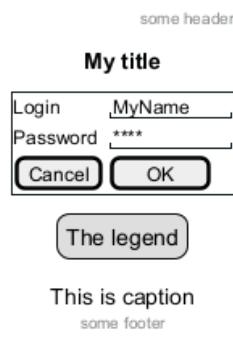
footer some footer

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

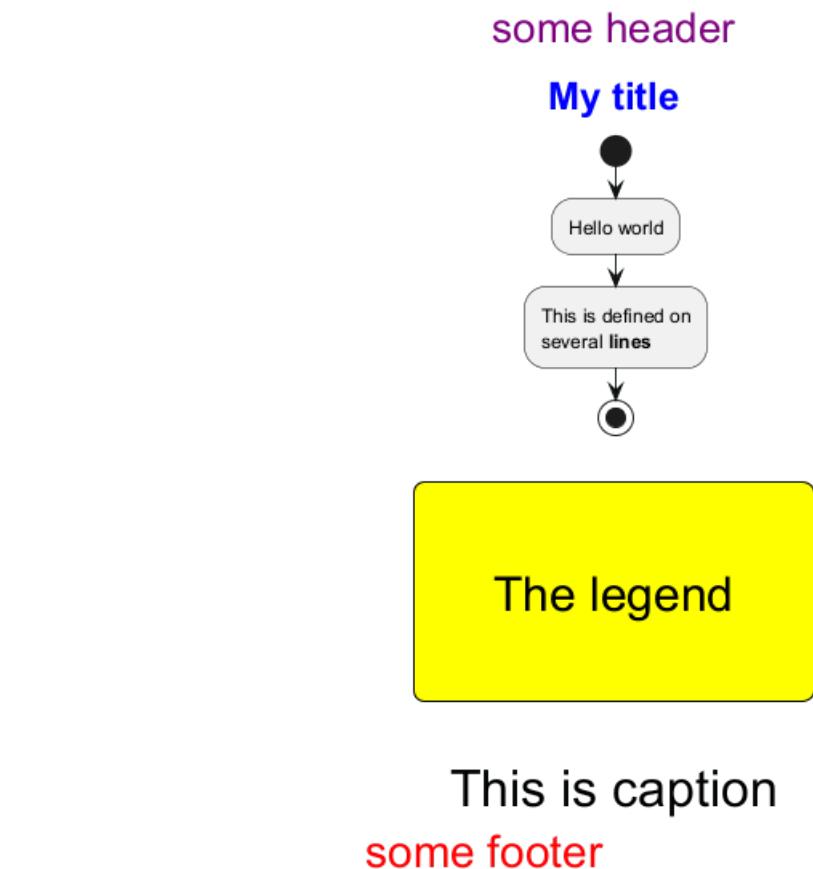
caption This is caption

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

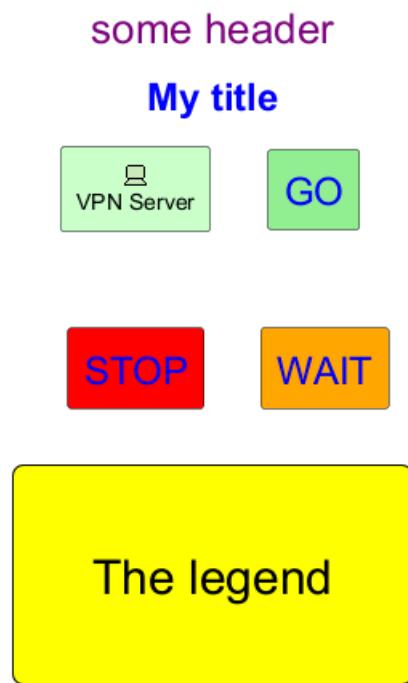
legend
The legend
end legend

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange

@enduml

```



This is caption
some footer

21.8.3 Class

```

@startuml
<style>
title {
    HorizontalAlignment right
    FontSize 24

```



```
FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

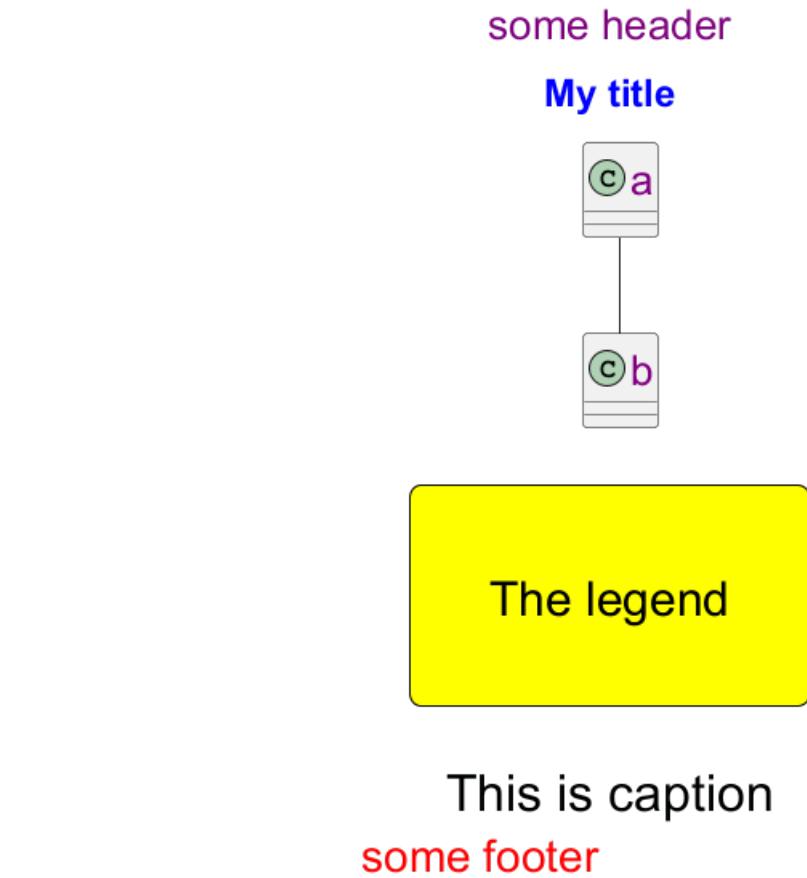
caption This is caption

legend
The legend
end legend

a -- b

@enduml
```





21.8.4 Component, Deployment, Use-Case

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

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

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



```

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

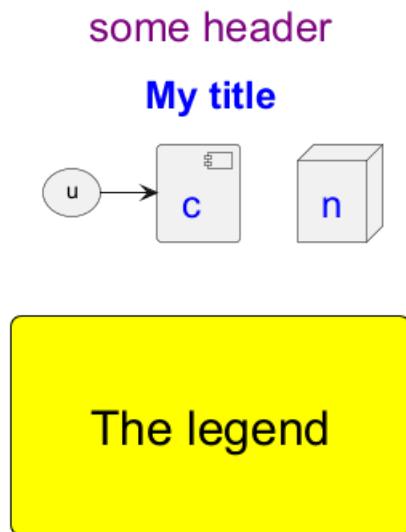
caption This is caption

legend
The legend
end legend

node n
(u) -> [c]

@enduml

```



This is caption
some footer

21.8.5 Gantt project planning

```

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

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple

```



```

}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

[t] lasts 5 days

@endgantt

```



This is caption
some footer



21.8.6 Object

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

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

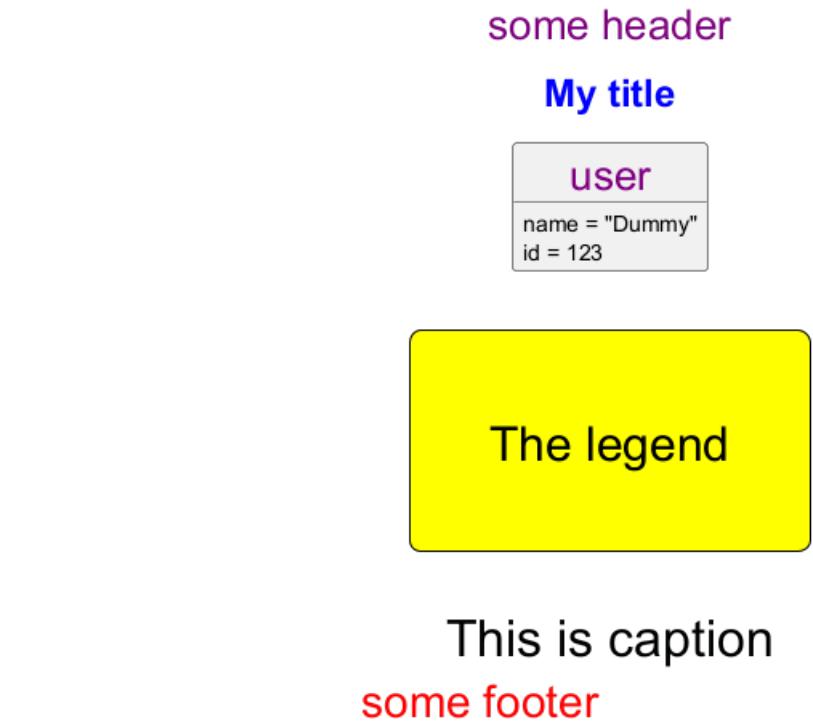
caption This is caption

legend
The legend
end legend

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

@enduml
```





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

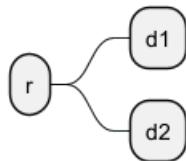
* r
** d1
** d2

@endmindmap

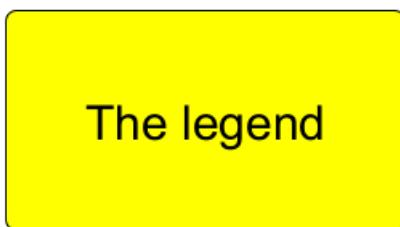
```

some header

My title



The legend



This is caption

some footer

21.8.8 Network (nwdiag)

```

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

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

```



```
footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

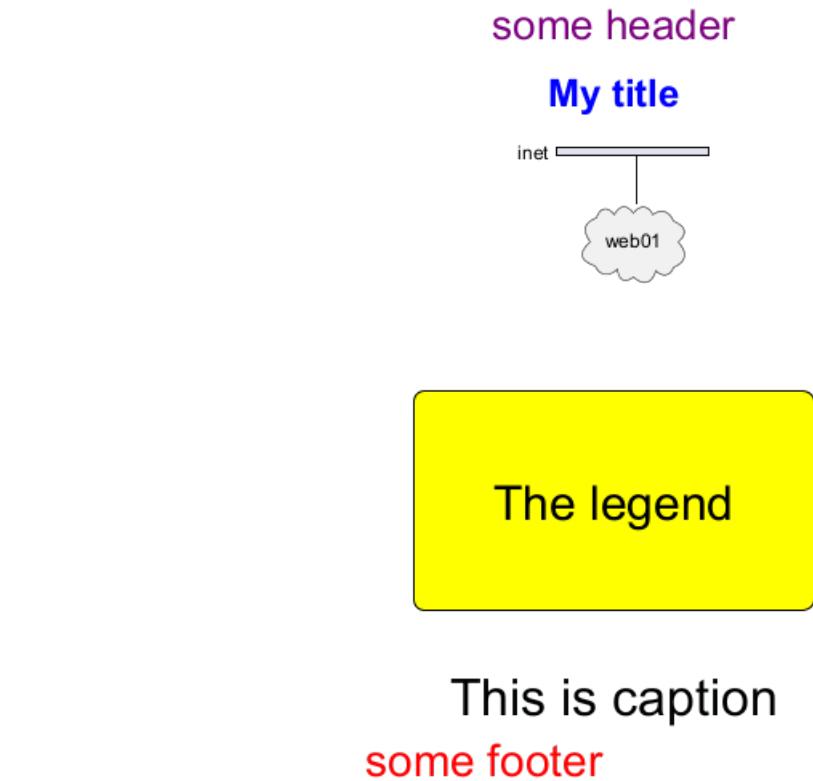
caption This is caption

legend
The legend
end legend

nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}

@enduml
```





21.8.9 Sequence

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

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
```



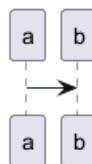
```

header some header
footer some footer
title My title
caption This is caption
legend
The legend
end legend
a->b
@enduml

```

some header

My title



The legend

This is caption
some footer

21.8.10 State

```

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

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
}

```



```
FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

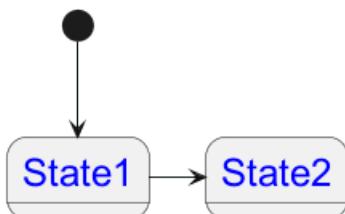
legend
The legend
end legend

[*] --> State1
State1 -> State2

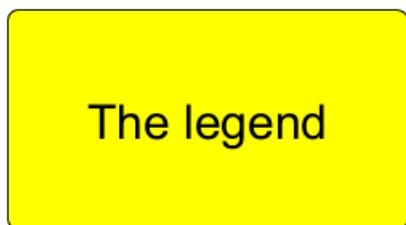
@enduml
```

some header

My title



The legend



This is caption

some footer



21.8.11 Timing

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

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

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

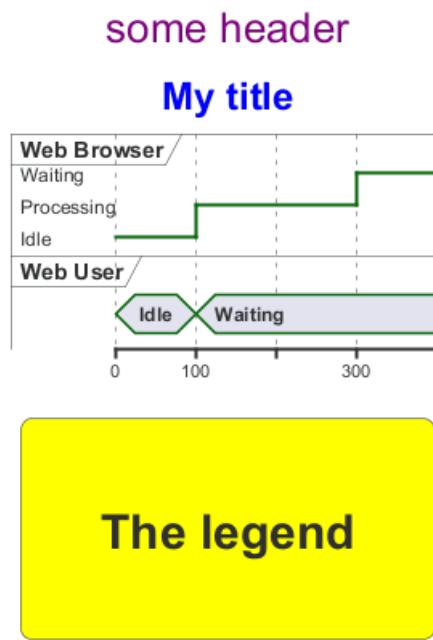
@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting
```



```
@enduml
```



This is caption
some footer

21.8.12 Work Breakdown Structure (WBS)

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

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

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



```

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

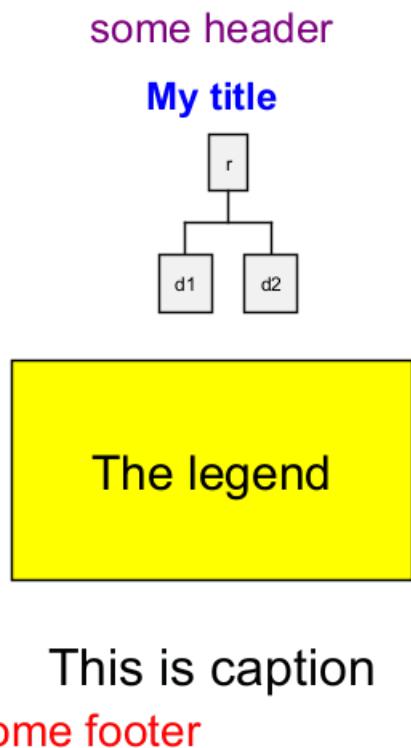
caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endwbs

```



This is caption
some footer

21.8.13 Wireframe (SALT)

TODO:FIXME Fix all (**title**, **caption**, **legend**, **header**, **footer**) for salt. **TODO:**FIXME

```

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

```



```

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
@startsalt
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

{+
    Login | "MyName"
    Password | "****"
    [Cancel] | [ OK ]
}
@endsalt
some header

```



The legend

This is caption

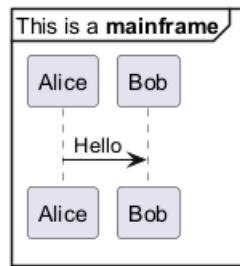
some footer

21.9 Mainframe

```
@startuml
mainframe This is a **mainframe**
```



```
Alice->Bob : Hello
@enduml
```



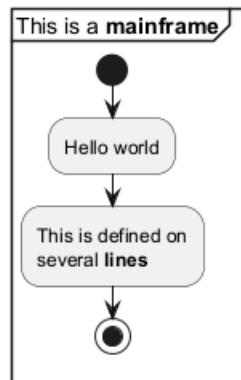
[Ref. QA-4019 and Issue#148]

21.10 Appendix: Examples of Mainframe on all diagram

21.10.1 Activity

```
@startuml
mainframe This is a **mainframe**

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

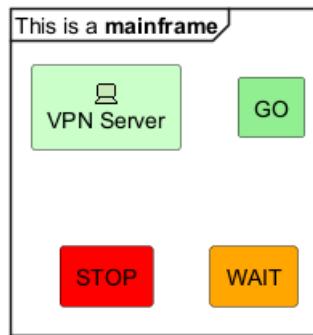


21.10.2 Archimate

```
@startuml
mainframe This is a **mainframe**

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>
rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
@enduml
```



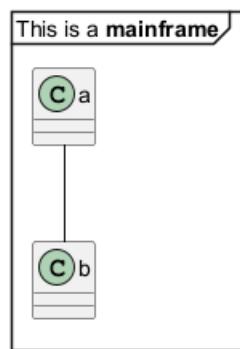


TODO: FIXME □ Cropped on the top and on the left **TODO:** FIXME

21.10.3 Class

```
@startuml
mainframe This is a **mainframe**

a -- b
@enduml
```

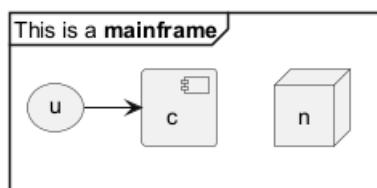


TODO: FIXME □ Cropped on the top and on the left **TODO:** FIXME

21.10.4 Component, Deployment, Use-Case

```
@startuml
mainframe This is a **mainframe**

node n
(u) -> [c]
@enduml
```



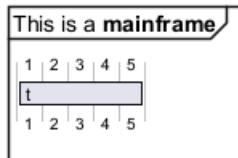
TODO: FIXME □ Cropped on the top and on the left **TODO:** FIXME

21.10.5 Gantt project planning

```
@startgantt
mainframe This is a **mainframe**

[t] lasts 5 days
@endgantt
```



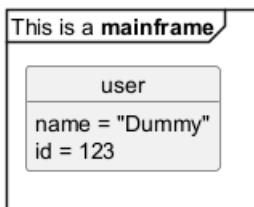


TODO: FIXME □ Cropped on the top and on the left **TODO:** FIXME

21.10.6 Object

```
@startuml
mainframe This is a **mainframe**

object user {
    name = "Dummy"
    id = 123
}
@enduml
```

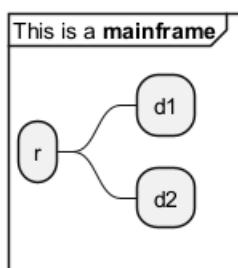


TODO: FIXME □ Cropped on the top! **TODO:** FIXME

21.10.7 MindMap

```
@startmindmap
mainframe This is a **mainframe**

* r
** d1
** d2
@endmindmap
```

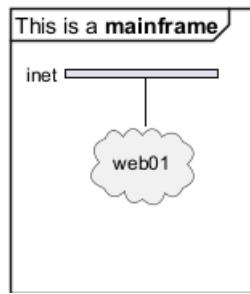


21.10.8 Network (nwdiag)

```
@startuml
mainframe This is a **mainframe**

nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}
@enduml
```





TODO: FIXME □ Cropped on the top! **TODO:** FIXME

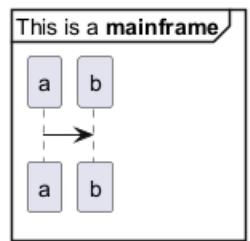
21.10.9 Sequence

```

@startuml
mainframe This is a **mainframe**

a->b
@enduml

```



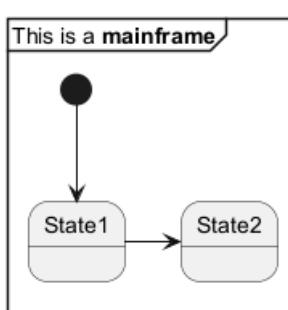
21.10.10 State

```

@startuml
mainframe This is a **mainframe**

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

```



TODO: FIXME □ Cropped on the top and on the left **TODO:** FIXME

21.10.11 Timing

```

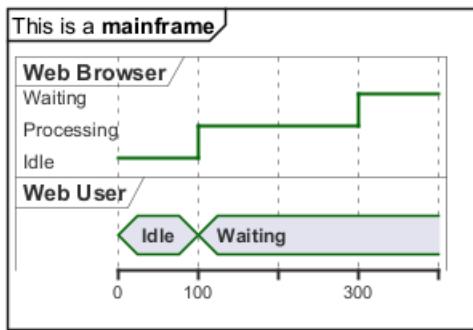
@startuml
mainframe This is a **mainframe**

robust "Web Browser" as WB
concise "Web User" as WU
@0
WU is Idle

```

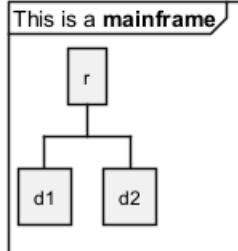


```
WB is Idle
@100
WU is Waiting
WB is Processing
@300
WB is Waiting
@enduml
```



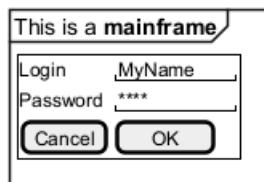
21.10.12 Work Breakdown Structure (WBS)

```
@startwbs
mainframe This is a **mainframe**
* r
** d1
** d2
@endwbs
```



21.10.13 Wireframe (SALT)

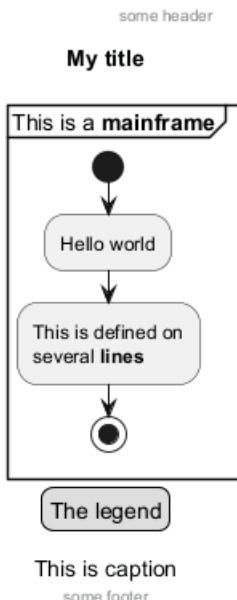
```
@startsalt
mainframe This is a **mainframe**
{+
    Login | "MyName"
    Password | "****"
    [Cancel] | [ OK ]
}
@endsalt
```



21.11 Appendix: Examples of title, header, footer, caption, legend and mainframe on all diagram

21.11.1 Activity

```
@startuml  
mainframe This is a **mainframe**  
header some header  
  
footer some footer  
  
title My title  
  
caption This is caption  
  
legend  
The legend  
end legend  
  
start  
:Hello world;  
:This is defined on  
several **lines**;  
stop  
  
@enduml
```



21.11.2 Archimate

```
@startuml  
mainframe This is a **mainframe**  
header some header  
  
footer some footer  
  
title My title  
  
caption This is caption
```



legend

The legend

end legend

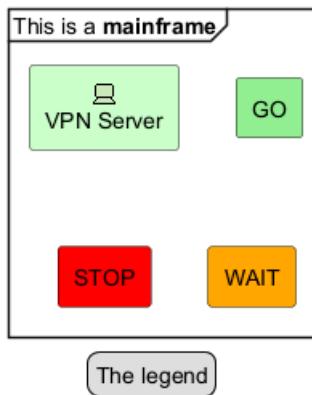
```
archimate #Technology "VPN Server" as vpnServerA <<technology-device>>
```

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

```
@enduml
```

some header

My title



This is caption

some footer

21.11.3 Class

```
@startuml
```

```
mainframe This is a **mainframe**  
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

legend

The legend

end legend

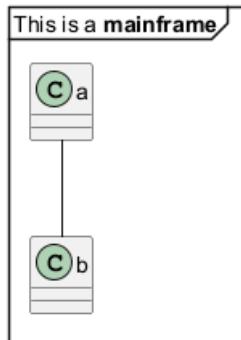
```
a -- b
```

```
@enduml
```



some header

My title



This is caption

some footer

21.11.4 Component, Deployment, Use-Case

```
@startuml  
mainframe This is a **mainframe**  
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

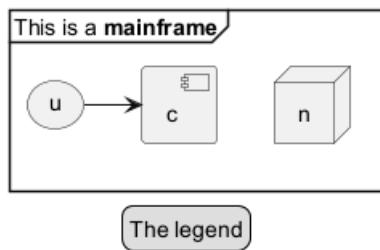
```
legend  
The legend  
end legend
```

```
node n  
(u) -> [c]
```

```
@enduml
```

some header

My title



This is caption

some footer

21.11.5 Gantt project planning

```
@startgantt  
mainframe This is a **mainframe**  
header some header
```



```
footer some footer  
title My title  
caption This is caption
```

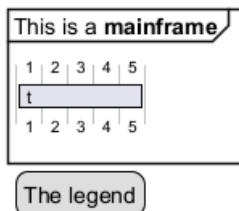
```
legend  
The legend  
end legend
```

```
[t] lasts 5 days
```

```
@endgantt
```

some header

My title



This is caption

some footer

21.11.6 Object

```
@startuml  
mainframe This is a **mainframe**  
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

```
legend  
The legend  
end legend
```

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

```
@enduml
```





21.11.7 MindMap

```
@startmindmap
mainframe This is a **mainframe**
header some header

footer some footer

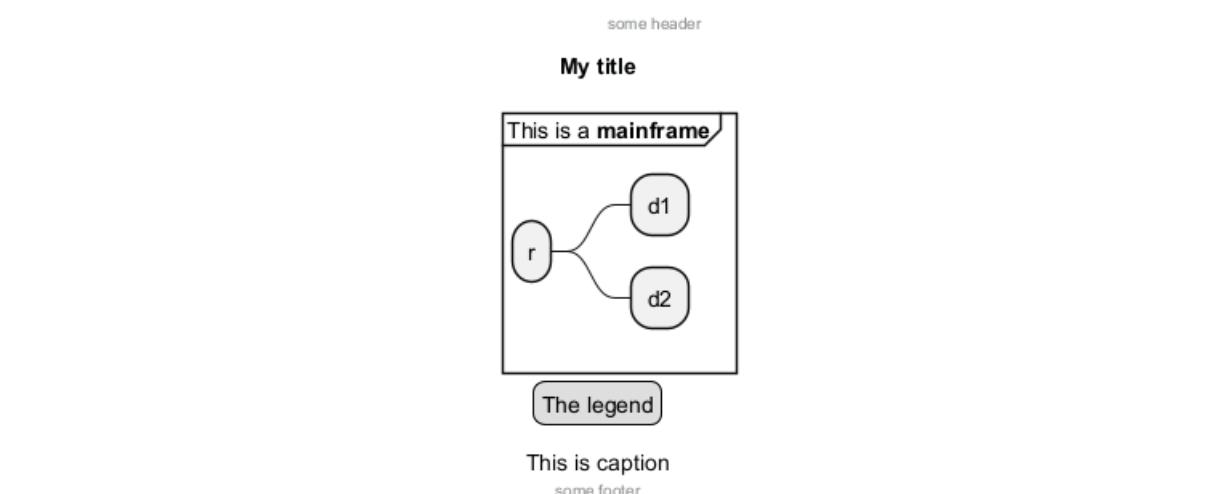
title My title

caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endmindmap
```

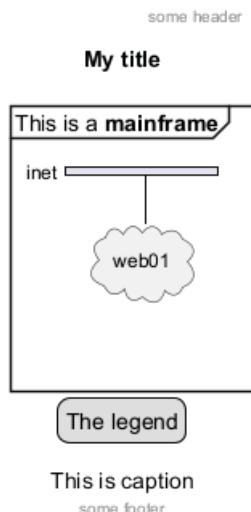


21.11.8 Network (nwdiag)

```
@startuml
mainframe This is a **mainframe**
header some header
```



```
footer some footer  
  
title My title  
  
caption This is caption  
  
legend  
The legend  
end legend  
  
nwdiag {  
    network inet {  
        web01 [shape = cloud]  
    }  
}  
  
@enduml
```



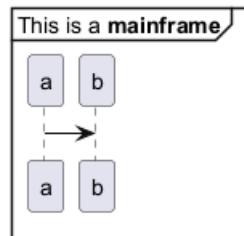
21.11.9 Sequence

```
@startuml  
mainframe This is a **mainframe**  
header some header  
  
footer some footer  
  
title My title  
  
caption This is caption  
  
legend  
The legend  
end legend  
  
a->b  
@enduml
```



some header

My title



The legend

This is caption

some footer

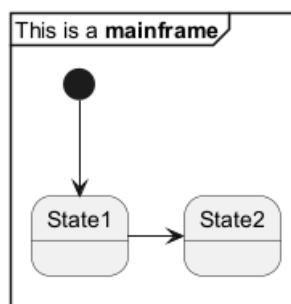
21.11.10 State

```
@startuml  
mainframe This is a **mainframe**  
header some header  
  
footer some footer  
  
title My title  
  
caption This is caption  
  
legend  
The legend  
end legend  
  
[*] --> State1  
State1 -> State2
```

@enduml

some header

My title



The legend

This is caption

some footer

21.11.11 Timing

```
@startuml  
mainframe This is a **mainframe**
```



```
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

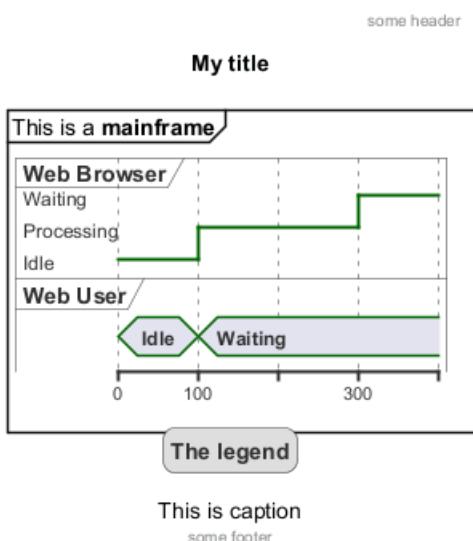
robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting

@enduml
```



21.11.12 Work Breakdown Structure (WBS)

```
@startwbs
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption
```



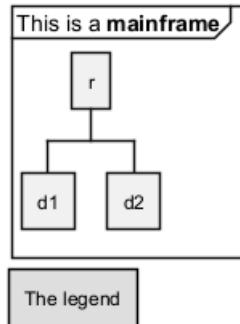
```
legend  
The legend  
end legend
```

```
* r  
** d1  
** d2
```

```
@endwbs
```

some header

My title



The legend

This is caption

some footer

21.11.13 Wireframe (SALT)

```
@startsalt  
mainframe This is a **mainframe**  
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

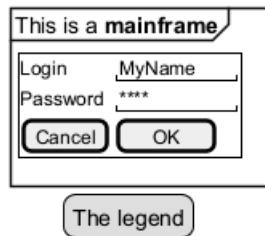
```
legend  
The legend  
end legend
```

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



some header

My title



This is caption

some footer



22 Creole синтаксис

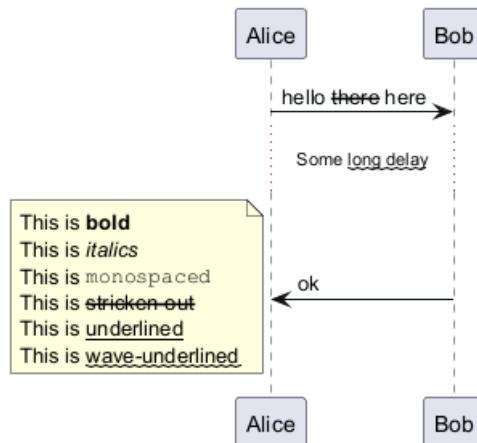
Creole - это облегчённый язык разметки, используемый в различных вики.

Чтобы иметь стандартизованный способ стилизации текста, в PlantUML интегрирован легковесный движок Creole и все диаграммы в PlantUML поддерживают этот синтаксис.

Также сохраняется совместимость с синтаксисом HTML.

22.1 Emphasized text

```
@startuml
Alice -> Bob : hello --there-- here
... Some ~~long delay~~ ...
Bob -> Alice : ok
note left
    This is **bold**
    This is //italics//
    This is ""monospaced"""
    This is --stricken-out--
    This is __underlined__
    This is ~~wave-underlined~~
end note
@enduml
```



22.2 Lists

You can use numbered and bulleted lists in node text, notes, etc.

TODO: You cannot quite mix numbers and bullets in a list and its sublist.

```
@startuml
object demo {
    * Bullet list
    * Second item
}
note left
    * Bullet list
    * Second item
    ** Sub item
end note

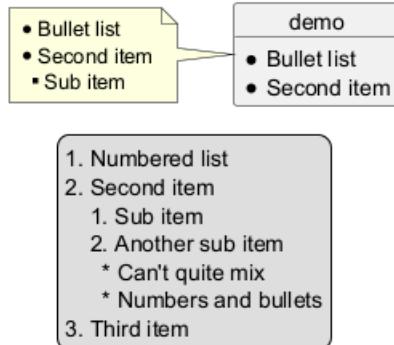
legend
# Numbered list
# Second item
## Sub item
```



```

## Another sub item
    * Can't quite mix
    * Numbers and bullets
# Third item
end legend
@enduml

```



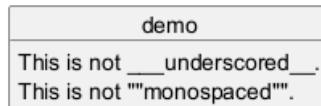
22.3 Escape character

You can use the tilde ~ to escape special creole characters.

```

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

```



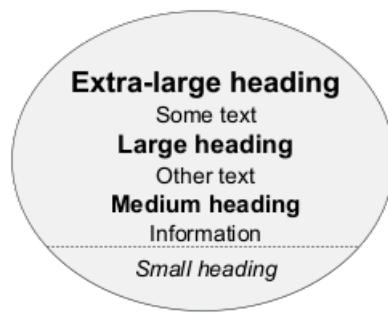
22.4 Headings

```

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

```

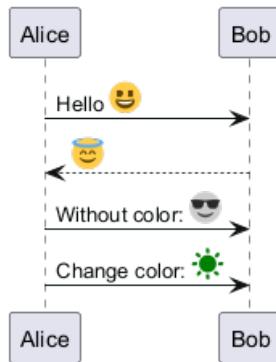




22.5 Emoji

All emojis from Twemoji (see EmojiTwo on Github) are available using the following syntax:

```
@startuml
Alice -> Bob : Hello <:1f600:>
return <:innocent:>
Alice -> Bob : Without color: <#0:sunglasses:>
Alice -> Bob : Change color: <#green:sunny:>
@enduml
```



Unlike Unicode Special characters that depend on installed fonts, the emoji are always available. Furthermore, emoji are already colored, but you can recolor them if you like (see examples above).

One can pick emoji from the emoji cheat sheet, the Unicode full-emoji-list, or the flat list emoji.txt in the plantuml source.

You can also use the following PlantUML command to list available emoji:

```
@startuml
emoji <block>
@enduml
```

As of 13 April 2023, you can select between 1174 emoji from the following Unicode blocks:

- Unicode block 26: 83 emoji
- Unicode block 27: 33 emoji
- Unicode block 1F3: 246 emoji
- Unicode block 1F4: 255 emoji
- Unicode block 1F5: 136 emoji
- Unicode block 1F6: 181 emoji
- Unicode block 1F9: 240 emoji



22.5.1 Unicode block 26

```
@startuml
emoji 26
@enduml
```

Emoji available on Unicode Block 26

(Blocks available: 26, 27, 1F3, 1F4, 1F5, 1F6, 1F9)

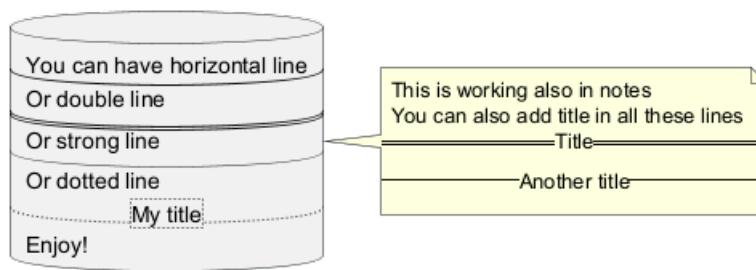
<:2600:> ☀️☀️ <:sunny:>	<:264d:> ♍️♍️ <:virgo:>	<:26aa:> ⚡⚡ <:white_circle:>
<:2601:> ☁️ ☁️ <:cloud:>	<:264e:> ♎️♎️ <:libra:>	<:26ab:> ●● <:black_circle:>
<:2602:> ☂️ ☂️ <:open_umbrella:>	<:264f:> ☽ ☽ <:scorpius:>	<:26b0:> ☠️ ☠️ <:coffin:>
<:2603:> 🎅⛄️ <:snowman_with_snow:>	<:2650:> ✸ ✸ <:sagittarius:>	<:26b1:> ☠️ ☠️ <:funeral_urn:>
<:2604:> 🌩️ 🌩️ <:comet:>	<:2651:> ☃ ☃ <:capricorn:>	<:26bd:> ⚽ ⚽ <:soccer:>
<:2605:> ☎️ ☎️ <:phone:>	<:2652:> ☷ ☷ <:aquarius:>	<:26be:> ⚾ ⚾ <:baseball:>
<:2611:> ✅ ✅ <:ballot_box_with_check:>	<:2653:> ☪ ☪ <:pisces:>	<:26c4:> 🎅⛄️ <:snowman:>
<:2614:> ☂️ ☂️ <:umbrella:>	<:265f:> ☜ ☜ <:chess_pawn:>	<:26c5:> ☦ ☦ <:partly_sunny:>
<:2615:> ☕ ☕ <:coffee:>	<:2660:> ♣ ♣ <:spades:>	<:26c8:> ☁️ ☀️ <:cloud_with_lightning_and_rain:>
<:2618:> ☘ ☘ <:shamrock:>	<:2663:> ♠ ♠ <:clubs:>	<:26ce:> ☤ ☤ <:ophichthus:>
<:261d:> ☝ ☝ <:point_up:>	<:2665:> ❤️ ❤️ <:hearts:>	<:26cf:> ↗ ↗ <:pick:>
<:2620:> ☠️ ☠️ <:skull_and_crossbones:>	<:2666:> ♦️ ♦️ <:diamonds:>	<:26d1:> ☠️ ☠️ <:rescue_worker_helmet:>
<:2622:> ☣ ☣ <:radioactive:>	<:2668:> ☳ ☳ <:hotsprings:>	<:26d3:> ☱ ☱ <:chains:>
<:2623:> ☣ ☣ <:biohazard:>	<:267b:> ☶ ☶ <:recycle:>	<:26d4:> ☞ ☞ <:no_entry:>
<:2626:> ☤ ☤ <:orthodox_cross:>	<:267e:> ☺ ☺ <:infinity:>	<:26e9:> ☮ ☮ <:shinto_shrine:>
<:262a:> ☢ ☢ <:star_and_crescent:>	<:267f:> ☤ ☤ <:wheelchair:>	<:26ea:> ☧ ☧ <:church:>
<:262e:> ☮ ☮ <:peace_symbol:>	<:2692:> ☨ ☨ <:hammer_and_pick:>	<:26f0:> ☪ ☪ <:mountain:>
<:262f:> ☯ ☯ <:yin_yang:>	<:2693:> ☳ ☳ <:anchor:>	<:26f1:> ☙ ☙ <:parasol_on_ground:>
<:2638:> ☮ ☮ <:wheel_of_dharma:>	<:2694:> ✸ ✸ <:crossed_swords:>	<:26f2:> ☬ ☬ <:fountain:>
<:2639:> ☹ ☹ <:frowning_face:>	<:2695:> ☷ ☷ <:medical_symbol:>	<:26f3:> ☪ ☪ <:golf:>
<:263a:> ☺ ☺ <:relaxed:>	<:2696:> ☰ ☰ <:balance_scale:>	<:26f4:> ☩ ☩ <:ferry:>
<:2640:> ☵ ☵ <:female_sign:>	<:2697:> ☪ ☪ <:alembic:>	<:26f5:> ☩ ☩ <:boat:>
<:2642:> ☶ ☶ <:male_sign:>	<:2699:> ☮ ☮ <:gear:>	<:26f7:> ☶ ☶ <:skier:>
<:2648:> ☊ ☊ <:aries:>	<:269b:> ☷ ☷ <:atom_symbol:>	<:26f8:> ☶ ☶ <:ice_skate:>
<:2649:> ☊ ☊ <:taurus:>	<:269c:> ☮ ☮ <:fleur_de_lis:>	<:26f9:> ☺ ☺ <:bouncing_ball_person:>
<:264a:> ☊ ☊ <:gemini:>	<:26a0:> ☠ ☠ <:warning:>	<:26fa:> ☠ ☠ <:tent:>
<:264b:> ☊ ☊ <:cancer:>	<:26a1:> ☢ ☢ <:zap:>	<:26fd:> ☩ ☩ <:fuel泵:>
<:264c:> ☊ ☊ <:leo:>	<:26a7:> ☭ ☭ <:transgender_symbol:>	

22.6 Horizontal lines

```
@startuml
database DB1 as "
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
Enjoy!
"
note right
This is working also in notes
You can also add title in all these lines
==Title==
--Another title--
end note

@enduml
```





22.7 Links

You can also use URL and links.

Simple links are define using two square brackets (or three square brackets for field or method on class diagram).

Example:

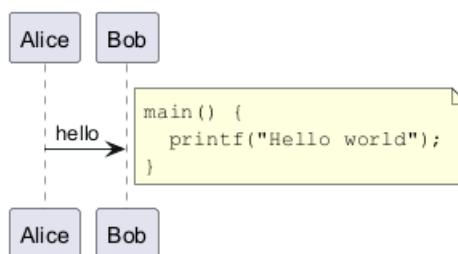
- [[http://plantuml.com]]
- [[http://plantuml.com This label is printed]]
- [[http://plantuml.com{Optional tooltip} This label is printed]]

URL can also be authenticated.

22.8 Code

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

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

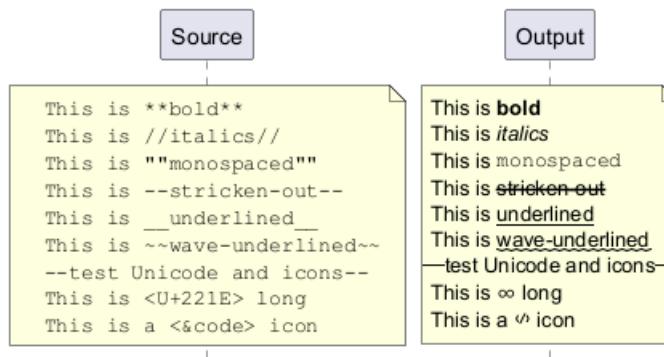


This is especially useful to illustrate some PlantUML code and the resulting rendering:

```
@startuml
hide footbox
note over Source
<code>
This is **bold**
This is //italics//
This is ""monospaced"""
This is --stricken-out--
This is __underlined__
This is ~~wave-underlined~~
```



```
--test Unicode and icons--
This is <U+221E> long
This is a <&code> icon
</code>
end note
/note over Output
This is **bold**
This is //italics//
This is ""monospaced"""
This is --stricken-out--
This is __underlined__
This is ~~wave-underlined~~
--test Unicode and icons--
This is <U+221E> long
This is a <&code> icon
end note
@enduml
```



22.9 Table

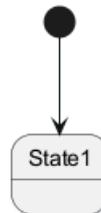
22.9.1 Create a table

It is possible to build table, with | separator.

```
@startuml
skinparam titleFontSize 14
title
Example of simple table
|= |= table |= header |
| a | table | row |
| b | table | row |
end title
[*] --> State1
@enduml
```

Example of simple table

table	header
a	row
b	row



22.9.2 Align fields using Table

You can use a table to align "fields" of class members. The example below (taken from buildingSmart Data Dictionary) shows for each member: icon, name, datatype and cardinality. Use the <#transparent,&#transparent> color specification so table cells have no foreground and background color.

(The example also shows the use of icons)

```
@startuml
hide empty members
hide circle

class "<:wrench:> Property" as Property {
<#transparent,#transparent>|<:link:>| id| iri| 1..1|
|<:spiral_notepad:>| name (bsdd:name)| string| 1..1|
|<:calendar:>| activationDateUtc| dateTime| 1..1|
|<:spiral_notepad:>| code| string| 1..1|
|<:spiral_notepad:>| connectedPropertyCode| string| 0..*|
|<:spiral_notepad:>| countryOfOrigin| string| 0..1|
|<:spiral_notepad:>| countryOfUse| string| 0..*|
|<:spiral_notepad:>| creatorLanguageCode| string| 0..1|
|<:spiral_notepad:>| dataType| string| 0..1|
|<:calendar:>| deActivationDateUtc| dateTime| 0..1|
|<:spiral_notepad:>| definition| string| 0..1|
|<:spiral_notepad:>| deprecationExplanation| string| 0..1|
|<:spiral_notepad:>| description| string| 0..1|
|<:spiral_notepad:>| dimension| string| 0..1|
|<:1234:>| dimensionAmountOfSubstance| int| 0..1|
|<:1234:>| dimensionElectricCurrent| int| 0..1|
|<:1234:>| dimensionLength| int| 0..1|
|<:1234:>| dimensionLuminousIntensity| int| 0..1|
|<:1234:>| dimensionMass| int| 0..1|
|<:1234:>| dimensionThermodynamicTemperature| int| 0..1|
|<:1234:>| dimensionTime| int| 0..1|
|<:spiral_notepad:>| documentReference| string| 0..1|
|<:spiral_notepad:>| dynamicParameterPropertyCodes| string| 0..*|
|<:spiral_notepad:>| example| string| 0..1|
|<:ballot_box_with_check:>| isDynamic| boolean| 1..1|
|<:eight_spoked_asterisk:>| maxExclusive| decimal| 0..1|
|<:eight_spoked_asterisk:>| maxInclusive| decimal| 0..1|
|<:spiral_notepad:>| methodOfMeasurement| string| 0..1|
|<:eight_spoked_asterisk:>| minExclusive| decimal| 0..1|
|<:eight_spoked_asterisk:>| minInclusive| decimal| 0..1|
|<:spiral_notepad:>| name| string| 1..1|
|<:spiral_notepad:>| pattern| string| 0..1|
|<:spiral_notepad:>| physicalQuantity| string| 0..1|
|<:book:>| propertyValueKind| PropertyValueKind| 0..1|
|<:spiral_notepad:>| replacedObjectCodes| string| 0..*|
|<:spiral_notepad:>| replacingObjectCodes| string| 0..*|
|<:calendar:>| revisionDateUtc| dateTime| 0..1|
|<:1234:>| revisionNumber| int| 0..1|
|<:spiral_notepad:>| status| string| 1..1|
|<:spiral_notepad:>| subdivisionsOfUse| string| 0..*|
|<:spiral_notepad:>| textFormat| string| 0..1|
|<:spiral_notepad:>| uid| string| 0..1|
|<:spiral_notepad:>| unit| string| 0..*|
|<:calendar:>| versionDateUtc| dateTime| 0..1|
|<:1234:>| versionNumber| int| 0..1|
|<:link:>| visualRepresentationUri| iri| 0..1|
}
```



@enduml

Property		
id	iri	1..1
name (bsdd:name)	string	1..1
activationDateUtc	dateTime	1..1
code	string	1..1
connectedPropertyCode	string	0..*
countryOfOrigin	string	0..1
countryOfUse	string	0..*
creatorLanguageCode	string	0..1
dataType	string	0..1
deActivationDateUtc	dateTime	0..1
definition	string	0..1
deprecationExplanation	string	0..1
description	string	0..1
dimension	string	0..1
dimensionAmountOfSubstance	int	0..1
dimensionElectricCurrent	int	0..1
dimensionLength	int	0..1
dimensionLuminousIntensity	int	0..1
dimensionMass	int	0..1
dimensionThermodynamicTemperature	int	0..1
dimensionTime	int	0..1
documentReference	string	0..1
dynamicParameterPropertyCodes	string	0..*
example	string	0..1
<input checked="" type="checkbox"/> isDynamic	boolean	1..1
* maxExclusive	decimal	0..1
* maxInclusive	decimal	0..1
* methodOfMeasurement	string	0..1
* minExclusive	decimal	0..1
* minInclusive	decimal	0..1
name	string	1..1
pattern	string	0..1
physicalQuantity	string	0..1
propertyValueKind	PropertyValueKind	0..1
replacedObjectCodes	string	0..*
replacingObjectCodes	string	0..*
revisionDateUtc	dateTime	0..1
revisionNumber	int	0..1
status	string	1..1
subdivisionsOfUse	string	0..*
textFormat	string	0..1
uid	string	0..1
unit	string	0..*
versionDateUtc	dateTime	0..1
versionNumber	int	0..1
visualRepresentationUri	iri	0..1

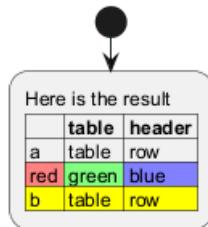
You can also try to use tabs \t and skinparam tabSize n to align fields, but this doesn't work so well: [Ref QA-3820]



22.9.3 Add color on rows or cells

You can specify background colors of rows and cells:

```
@startuml
start
:Here is the result
|= |= table |= header |
| a | table | row |
|<#FF8080> red |<#80FF80> green |<#8080FF> blue |
<#yellow>| b | table | row |
@enduml
```



22.9.4 Add color on border and text

You can also specify colors of text and borders.

```
@startuml
title
<#lightblue,#red>|= Step |= Date |= Name |= Status |= Link |
<#lightgreen>| 1.1 | TBD | plantuml news |<#Navy><color:OrangeRed><b> Unknown | [[https://plantuml.com]]
end title
@enduml
```

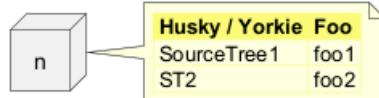
Step	Date	Name	Status	Link
1.1	TBD	plantuml news	Unknown	plantuml news

[Ref. QA-7184]

22.9.5 No border or same color as the background

You can also set the border color to the same color as the background.

```
@startuml
node n
note right of n
<#FBFB77,&#FBFB77>|= Husky / Yorkie |= Foo |
| SourceTree1 | foo1 |
| ST2 | foo2 |
end note
@enduml
```



[Ref. QA-12448]

22.9.6 Bold header or not

= as the first char of a cell indicates whether to make it bold (usually used for headers), or not.



```

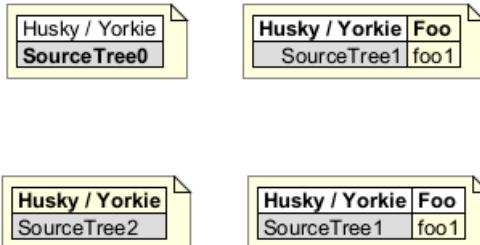
@startuml
note as deepCSS0
|<#white> Husky / Yorkie |
|= <#gainsboro> SourceTree0 |
endnote

note as deepCSS1
|= <#white> Husky / Yorkie |= Foo |
|<#gainsboro><r> SourceTree1 | foo1 |
endnote

note as deepCSS2
|= Husky / Yorkie |
|<#gainsboro> SourceTree2 |
endnote

note as deepCSS3
<#white>|= Husky / Yorkie |= Foo |
|<#gainsboro> SourceTree1 | foo1 |
endnote
@enduml

```



[Ref. QA-10923]

22.10 Tree

You can use | _ characters to build a tree.

On common commands, like title:

```

@startuml
skinparam titleFontSize 14
title
    Example of Tree
    |_ First line
    |_ **Bom (Model)**
    |_ prop1
    |_ prop2
    |_ prop3
    |_ Last line
end title
[*] --> State1
@enduml

```

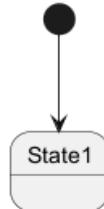


Example of Tree

```

└─ First line
  └─ Bom (Model)
    └─ prop1
    └─ prop2
    └─ prop3
  └─ Last line

```



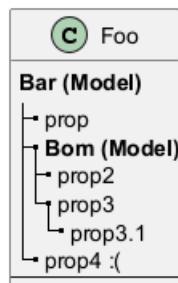
On Class diagram.

(Please note how we have to use an empty second compartment, else the parentheses in **(Model)** cause that text to be moved to a separate first compartment):

```

@startuml
class Foo {
**Bar (Model)**
|_ prop
|_ **Bom (Model)**
|_ prop2
|_ prop3
|_ prop3.1
|_ prop4 :(
-- 
}
@enduml

```



[Ref. QA-3448]

On Component or Deployment diagrams:

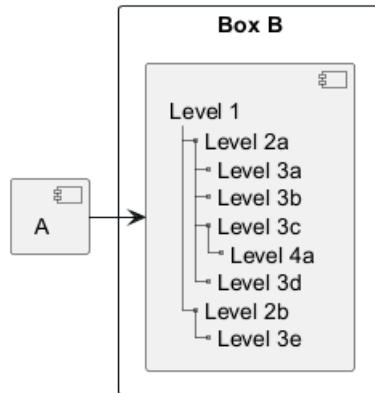
```

@startuml
[A] as A
rectangle "Box B" {
    component B [
        Level 1
        |_ Level 2a
        |_ Level 3a
        |_ Level 3b
        |_ Level 3c
        |_ Level 4a
        |_ Level 3d
        |_ Level 2b
        |_ Level 3e
    ]
}

```



```
]
}
A -> B
@enduml
```

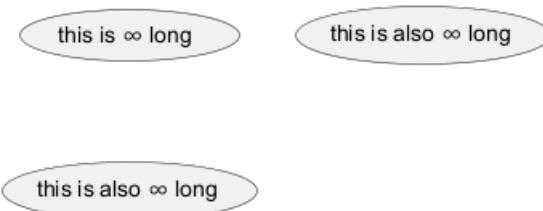


[Ref. QA-11365]

22.11 Special characters

It's possible to use any unicode character, either directly or with syntax &#nnnnnn; (decimal) or <U+XXXXXX> (hex):

```
@startuml
usecase direct as "this is ☺ long"
usecase ampHash as "this is also ☺ long"
usecase angleBrackets as "this is also <U+221E> long"
@enduml
```



Please note that not all Unicode chars appear correctly, depending on installed fonts.

- You can use the listfonts command with a test string of your desired characters, to see which fonts may include them.
- For characters that are emoji, it's better to use the Emoji notation that doesn't depend on installed fonts, and the emoji are colored.
- The PlantUML server has the "Noto Emoji" font that has most emoji. If you want to render diagrams on your local system, you should check which fonts you have.
- Unfortunately "Noto Emoji" lacks normal chars, so you need to switch fonts, eg

```
@startuml
rectangle "<font:Noto Emoji><U+1F3F7></font> label"
rectangle "<font:Noto Emoji><U+1F527></font> wrench"
rectangle "<font:Noto Emoji><U+1F6E0></font> hammer_and_wrench"
@enduml
```





See Issue 72 for more details.

22.12 Legacy HTML

You can mix Creole with the following HTML tags:

- for bold text
- <u> or <u:#AAAAAA> or <u: [[color|colorName]]> for underline
- <i> for italic
- <s> or <s:#AAAAAA> or <s: [[color|colorName]]> for strike text
- <w> or <w:#AAAAAA> or <w: [[color|colorName]]> for wave underline text
- <plain> for plain text
- <color:#AAAAAA> or <color: [[color|colorName]]>
- <back:#AAAAAA> or <back: [[color|colorName]]> for background color
- <size:nn> to change font size
- <img:file> : the file must be accessible by the filesystem
- <img:https://plantuml.com/logo3.png> : the URL must be available from the Internet
- {scale:nn} to change image size, eg <img:file.png{scale=0.3}>

```
@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:https://plantuml.com/logo3.png>
;
@enduml
```



22.12.1 Common HTML element

```

@startuml
hide footbox
note over Source
<code>
This is <b>bold</b>
This is <i>italics</i>
This is <font:monospaced>monospaced</font>
This is <s>stroked</s>
This is <u>underlined</u>
This is <w>waved</w>
This is <s:green>stroked</s>
This is <u:red>underlined</u>
This is <w:#0000FF>waved</w>
This is <b>a bold text containing <plain>plain text</plain> inside</b>
-- other examples --
This is <color:blue>Blue</color>
This is <back:orange>Orange background</back>
This is <size:20>big</size>
</code>
end note
/note over Output
This is <b>bold</b>
This is <i>italics</i>
This is <font:monospaced>monospaced</font>
This is <s>stroked</s>
This is <u>underlined</u>
This is <w>waved</w>
This is <s:green>stroked</s>
This is <u:red>underlined</u>
This is <w:#0000FF>waved</w>
This is <b>a bold text containing <plain>plain text</plain> inside</b>
-- other examples --
This is <color:blue>Blue</color>
This is <back:orange>Orange background</back>
This is <size:20>big</size>
end note
@enduml

```



[Ref. QA-5254 for plain]

22.12.2 Subscript and Superscript element [sub, sup]

```
@startuml
:
This is the "caffeine" molecule: C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>
</code>
This is the "caffeine" molecule: C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>
-----
<code>
This is the Pythagorean theorem: a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup>
</code>
This is the Pythagorean theorem: a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup>;
@enduml
```

This is the "caffeine" molecule: C₈H₁₀N₄O₂
 This is the "caffeine" molecule: C₈H₁₀N₄O₂

This is the Pythagorean theorem: a² + b² = c²
 This is the Pythagorean theorem: a² + b² = c²

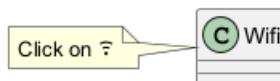
22.13 OpenIconic

OpenIconic is a very nice open-source icon set. Those icons are integrated in the creole parser, so you can use them out-of-the-box.

Use the following syntax: <&ICON_NAME>.

```
@startuml
title: <size:20><&heart>Use of OpenIconic<&heart></size>
class Wifi
note left
  Click on <&wifi>
end note
@enduml
```

♥Use of OpenIconic♥



The complete list is available with the following special command:

```
@startuml
listopeniconic
@enduml
```



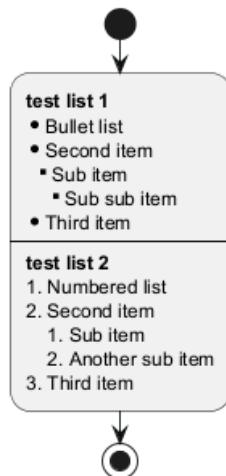
List Open Iconic	◆ bell	◆ cloud	≡ excerpt	≡ justify-right	♪ musical-note	★ star
Credit to https://useiconic.com/open	✿ bluetooth	☁ cloudy	⊖ expand-down	☛ key	☛ paperclip	☀ sun
	▣ bold	▷ code	◁ expand-left	▷ laptop	☛ pencil	▢ tablet
	↑ bolt	✿ cog	▷ expand-right	▷ layers	☛ people	☛ tag
- account-login	▣ book	▷ collapse-down	▷ expand-up	✿ lightbulb	☛ person	☛ tags
- account-logout	▣ bookmark	▷ collapse-left	▷ external-link	?? link-broken	☛ phone	@@ target
↷ action-redo	▣ box	▷ collapse-right	☛ eye	☛ link-intact	☛ pie-chart	☛ task
↶ action-undo	▣ briefcase	▷ collapse-up	☛ eyedropper	≡ list-rich	† pin	▣ terminal
≡ align-center	£ british-pound	✖ command	☛ file	≡ list	● play-circle	TEXT
≡ align-left	▣ browser	■ comment-square	☛ fire	↗ location	+ plus	thumb-down
≡ align-right	✗ brush	☛ compass	☛ flag	☛ lock-locked	○ power-standby	thumb-up
⌚ aperture	✿ bug	● contrast	☛ flash	☛ lock-unlocked	☛ print	⌚ timer
↓ arrow-bottom	▼ bulhorn	≡ copywriting	■ folder	☛ loop-circular	≡ project	☛ transfer
⌚ arrow-circle-bottom	▣ calculator	■ credit-card	☛ fork	☛ loop-square	★ pulse	☛ trash
⌚ arrow-circle-left	▣ calendar	☛ crop	☛ fullscreen-enter	☛ loop	☛ puzzle-piece	underline
⌚ arrow-circle-right	▣ camera-slr	☛ dashboard	✖ fullscreen-exit	ⓧ magnifying-glass	? question-mark	vertical-align-bottom
⌚ arrow-circle-top	▼ caret-bottom	↑ data-transfer-download	ⓧ globe	ⓧ map-marker	✿ rain	vertical-align-center
← arrow-left	◀ caret-left	↑ data-transfer-upload	☛ graph	☛ map	✖ random	vertical-align-top
→ arrow-right	▶ caret-right	☛ delete	☰ grid-four-up	media-pause	☛ reload	☛ video
↓ arrow-thick-bottom	▲ caret-top	☛ dial	☰ grid-three-up	► media-play	☛ resize-both	volume-high
← arrow-thick-left	▼ cart	☛ document	☰ grid-two-up	● media-record	☛ resize-height	volume-low
→ arrow-thick-right	■ chat	✿ dollar	☛ hard-drive	◀ media-skip-backward	☛ resize-width	volume-off
↑ arrow-thick-top	✓ check	” double-quote-sans-left	☛ header	► media-skip-forward	☛ rss-alt	warning
↑ arrow-top	▼ chevron-bottom	” double-quote-sans-right	Ⓐ headphones	☛ media-step-backward	☛ rss	wifi
॥ audio-spectrum	◀ chevron-left	” double-quote-serif-left	♥ heart	☛ media-step-forward	☛ script	wrench
⊛ audio	▶ chevron-right	” double-quote-serif-right	⌂ home	☛ media-stop	☛ share-boxed	x
✿ badge	▲ chevron-top	✿ droplet	☛ image	✿ medical-cross	☛ share	¥ yen
⌚ ban	● circle-check	▲ eject	☛ inbox	≡ menu	☛ shield	zoom-in
▣ bar-chart	● circle-x	▲ elevator	∞ infinity	✿ microphone	☛ signal	zoom-out
✿ basket	▣ clipboard	… ellipses	☛ info	- minus	☛ signpost	
□ battery-empty	⌚ clock	☛ envelope-closed	☛ italic	☛ monitor	☛ sort-ascending	
■ battery-full	▲ cloud-download	☛ envelope-open	≡ justify-center	☛ moon	☛ sort-descending	
✿ beaker	✿ cloud-upload	€ euro	≡ justify-left	☛ move	☛ spreadsheet	

22.14 Appendix: Examples of "Creole List" on all diagrams

22.14.1 Activity

```
@startuml
start
:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item;
stop
@enduml
```





22.14.2 Class

TODO: FIXME □

- *Sub item*
- *Sub sub item*

TODO: FIXME

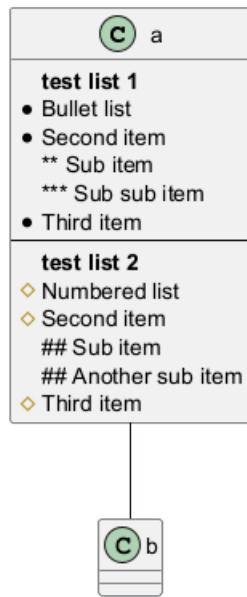
@startuml

```
class a {
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}
```

a -- b

@enduml





22.14.3 Component, Deployment, Use-Case

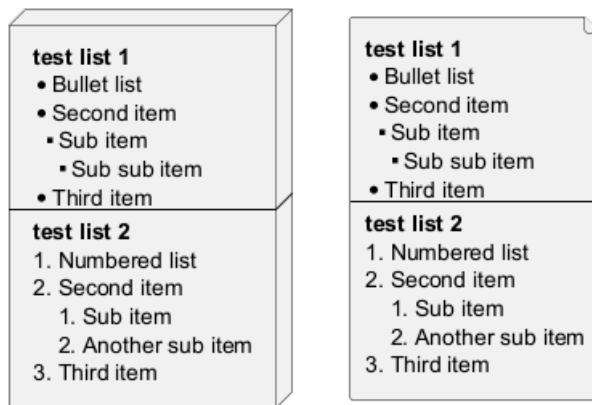
```

@startuml
node n [
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
]

file f as "
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
"
@enduml

```





TODO: DONE [Corrected in V1.2020.18]

22.14.4 Gantt project planning

N/A

22.14.5 Object

TODO: FIXME □

- *Sub item*
- *Sub sub item*

TODO: FIXME

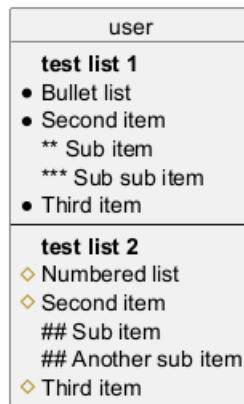
```

@startuml
object user {
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}

@enduml

```





22.14.6 MindMap

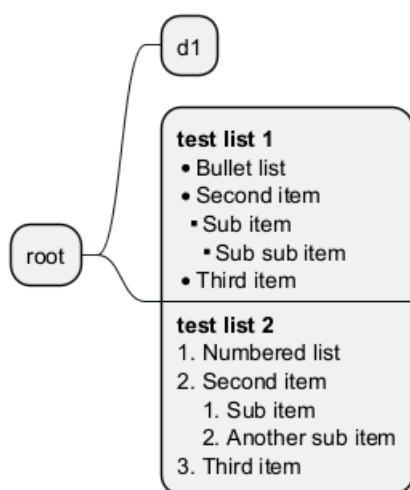
@startmindmap

```

* root
** d1
**:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item;

```

@endmindmap

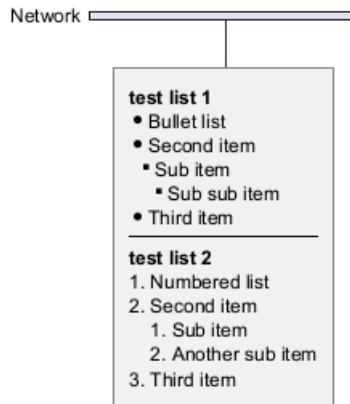


22.14.7 Network (nwdiag)

@startuml

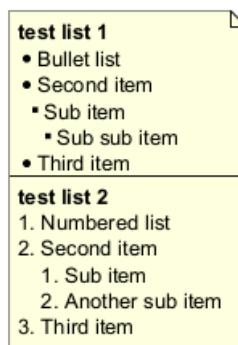


```
nwdiag {
    network Network {
        Server [description="**test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item"]
    }
    @enduml
```



22.14.8 Note

```
@startuml
note as n
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
end note
@enduml
```



22.14.9 Sequence

```
@startuml
<style>
participant {HorizontalAlignment left}
```



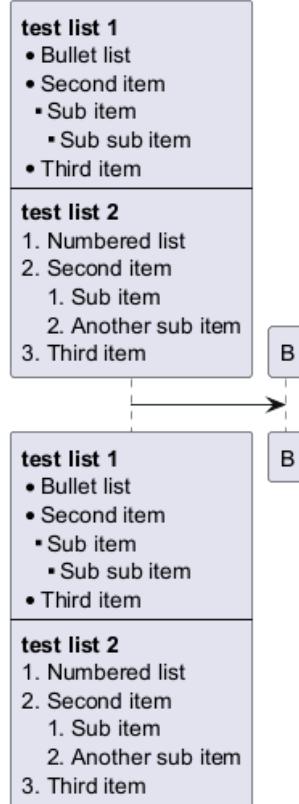
```

</style>
participant Participant [
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
]

participant B

Participant -> B
@enduml

```



[Ref. QA-15232]

22.14.10 State

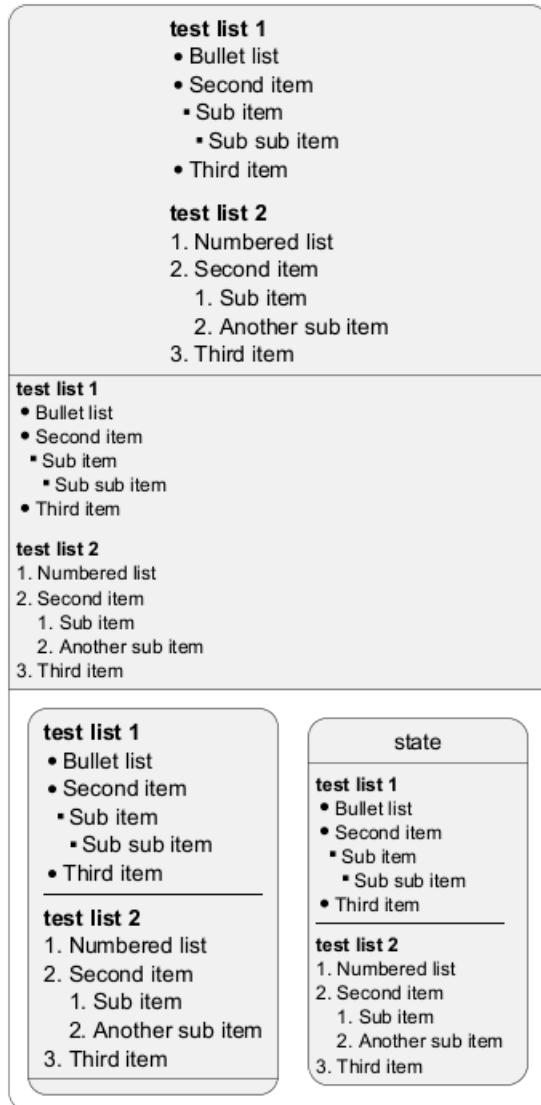
```

@startuml
<style>
stateDiagram {
title {HorizontalAlignment left}
}
</style>
state "**test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n--"

```



```
a: **test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n----\nstate "**test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n--\nstate : **test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n}\n\n@enduml
```



[Ref. QA-16978]

22.14.11 WBS

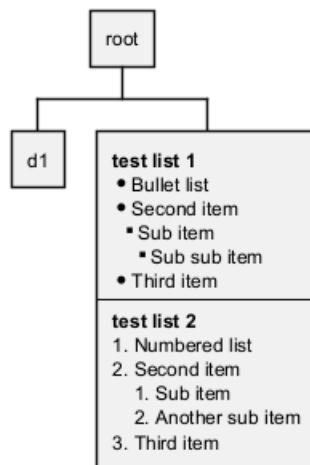
@startwbs

```
* root
** d1
***:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
-----
**test list 2**
# Numbered list
```



```
# Second item
## Sub item
## Another sub item
# Third item;

@endwbs
```



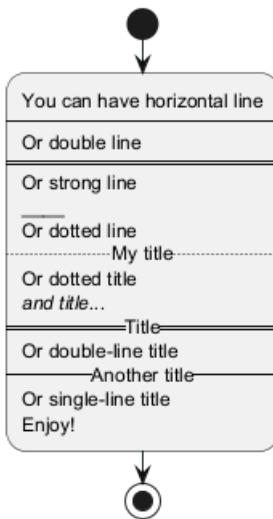
22.15 Appendix: Examples of "Creole horizontal lines" on all diagrams

22.15.1 Activity

TODO:FIXME strong line **----** **TODO:**FIXME

```
@startuml
start
:You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;
stop
@enduml
```





22.15.2 Class

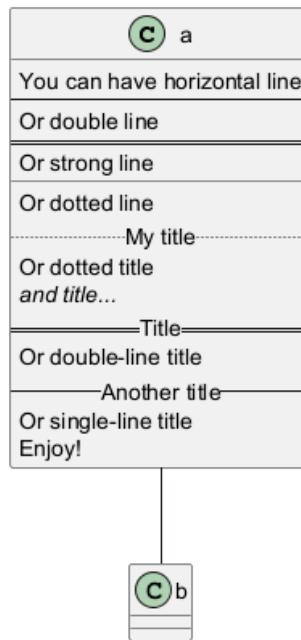
```
@startuml
```

```
class a {
    You can have horizontal line
    ----
    Or double line
    ====
    Or strong line
    ----
    Or dotted line
    ..My title..
    Or dotted title
    //and title... //
    ==Title==
    Or double-line title
    --Another title--
    Or single-line title
    Enjoy!
}
```

```
a -- b
```

```
@enduml
```





22.15.3 Component, Deployment, Use-Case

```
@startuml
node n [
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
]

file f as "
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
"

```

person p [

You can have horizontal line

-----

```
Or double line
=====
```

```
Or strong line
----
```

```
Or dotted line
```

```
..My title...
```

```
//and title... //
```

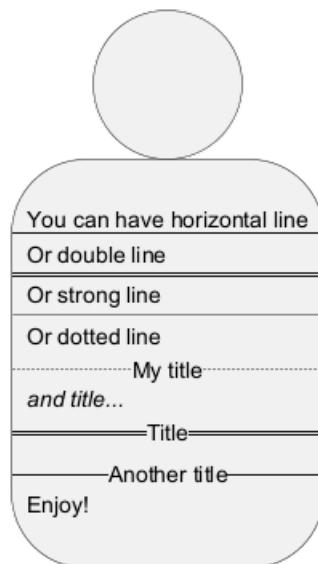
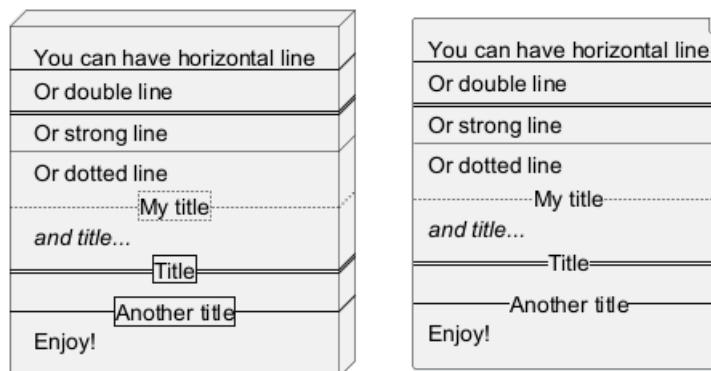
```
==Title==
```

```
--Another title--
```

```
Enjoy!
```

```
]
```

```
@enduml
```



22.15.4 Gantt project planning

N/A

22.15.5 Object

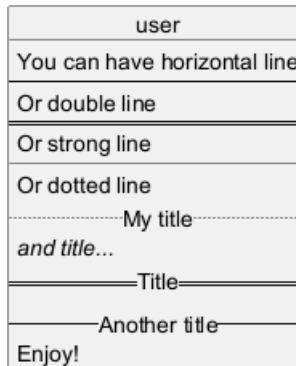
```
@startuml
object user {
You can have horizontal line
----
Or double line
=====
```



Or strong line

```
-----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
}
```

@enduml



TODO: DONE [Corrected on VI.2020.18]

22.15.6 MindMap

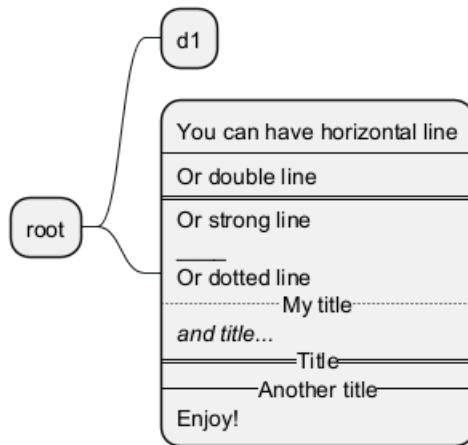
TODO:FIXME strong line ----- **TODO:**FIXME

@startmindmap

```
* root
** d1
**:You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!;
```

@endmindmap

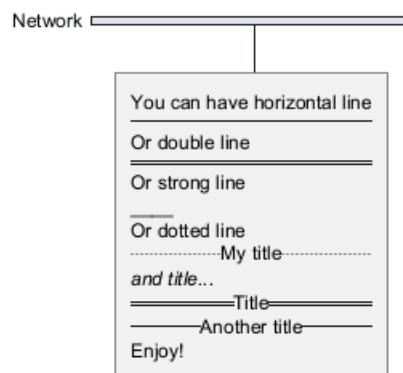




22.15.7 Network (nwdiag)

```

@startuml
nwdiag {
    network Network {
        Server [description="You can have horizontal line\n---\nOr double line\n====\nOr strong line\n-----\nOr dotted line\n..... My title .....\nand title...\n----- Title ----- \n----- Another title ----- \nEnjoy!"]
    }
@enduml
  
```

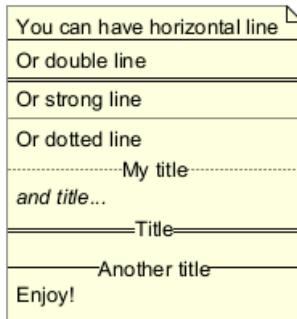


22.15.8 Note

```

@startuml
note as n
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
end note
@enduml
  
```





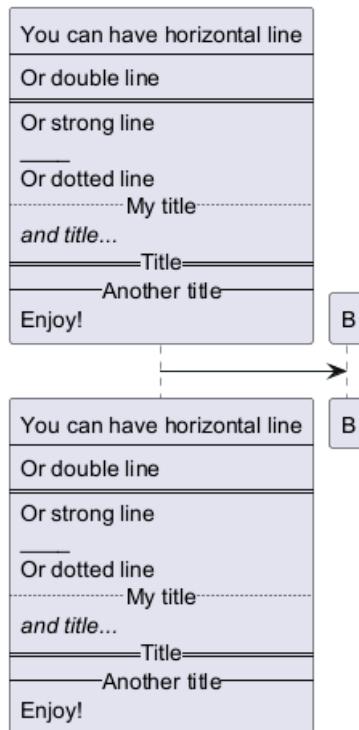
22.15.9 Sequence

```
@startuml
<style>
participant {HorizontalAlignment left}
</style>
participant Participant [
You can have horizontal line
-----
Or double line
=====
Or strong line
-----
Or dotted line
..My title...
//and title... //
==Title==
--Another title--
Enjoy!
]

participant B

Participant -> B
@enduml
```



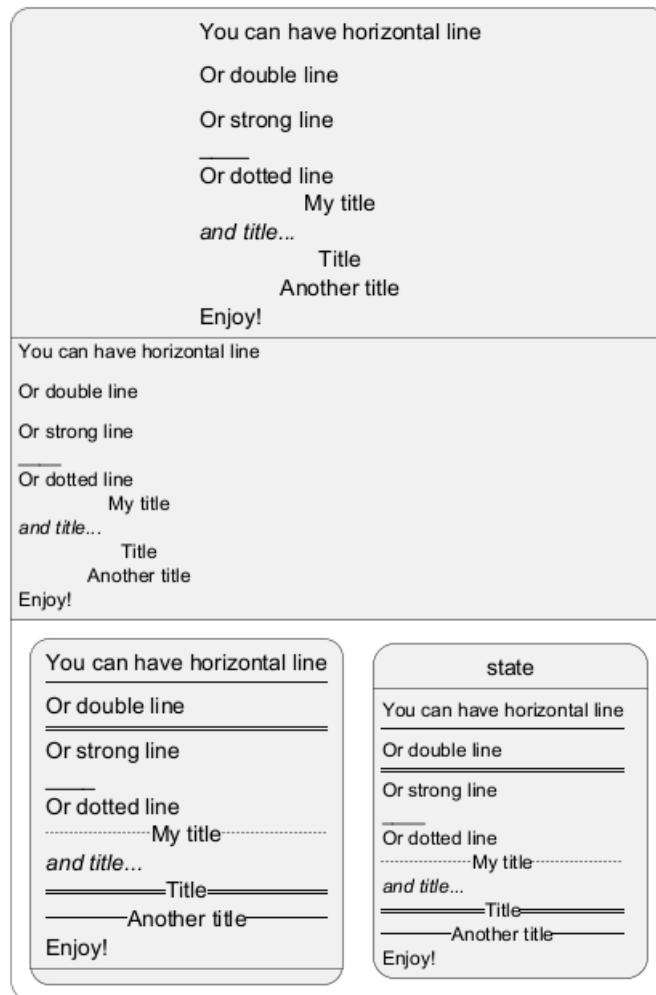


[Ref. QA-15232]

22.15.10 State

```
@startuml
<style>
stateDiagram {
title {HorizontalAlignment left}
}
</style>
state "You can have horizontal line\n----\nOr double line\n====\nOr strong line\n___\nOr dotted line"
a: You can have horizontal line\n----\nOr double line\n====\nOr strong line\n___\nOr dotted line\n.
state "You can have horizontal line\n----\nOr double line\n====\nOr strong line\n___\nOr dotted line"
state : You can have horizontal line\n----\nOr double line\n====\nOr strong line\n___\nOr dotted line
}
@enduml
```





[Ref. QA-16978, GH-1479]

22.15.11 WBS

TODO: FIXME strong line ---- TODO: FIXME

@startwbs

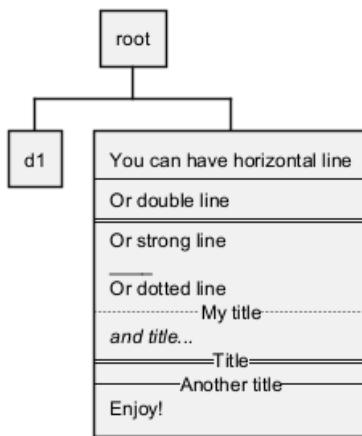
```

* root
** d1
**:You can have horizontal line
----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!;

```

@endwbs





22.16 Style equivalent (between Creole and HTML)

Style	Creole	Legacy HTML like
bold	This is **bold**	This is bold
<i>italics</i>	This is //italics//	This is <i>italics</i>
<u>monospaced</u>	This is ""monospaced""	This is <font:monospaced>monospaced
<u>stroked</u>	This is --stroked--	This is <s>stroked</s>
<u>underlined</u>	This is __underlined__	This is <u>underlined</u>
waved	This is ~~~	This is <w>waved</w>

```

@startmindmap
* Style equivalent\n(between Creole and HTML)
**:**Creole**
-----
<#silver>|= code|= output
| \n This is ""~**bold**"\n | \n This is **bold** |
| \n This is ""~//italics//"\n | \n This is //italics// |
| \n This is ""~"monospaced~"" "\n | \n This is ""monospaced"" |
| \n This is ""~~stroked~~"\n | \n This is --stroked-- |
| \n This is ""~__underlined__"\n | \n This is __underlined__ |
| \n This is ""~<U+007E><U+007E>waved<U+007E><U+007E>""\n | \n This is ~~waved~~ |
**:<b>Legacy HTML like
-----
<#silver>|= code|= output
| \n This is ""~<b>bold</b>""\n | \n This is <b>bold</b> |
| \n This is ""~<i>italics</i>""\n | \n This is <i>italics</i> |
| \n This is ""~<font:monospaced>monospaced</font>""\n | \n This is <font:monospaced>monospaced</font>
| \n This is ""~<s>stroked</s>""\n | \n This is <s>stroked</s> |
| \n This is ""~<u>underlined</u>""\n | \n This is <u>underlined</u> |
| \n This is ""~<w>waved</w>""\n | \n This is <w>waved</w> |

And color as a bonus...
<#silver>|= code|= output
| \n This is ""~<s: #color:green>""green""</color>"">stroked</s>""\n | \n This is <s:green>stroked</s> |
| \n This is ""~<u: #color:red>""red""</color>"">underlined</u>""\n | \n This is <u:red>underlined</u> |
| \n This is ""~<w: #color:#0000FF>""#0000FF""</color>"">waved</w>""\n | \n This is <w:#0000FF>waved</w>
@endmindmap

```



Creole	
code	output
This is **bold**	This is bold
This is //italics//	This is <i>italics</i>
This is ""monospaced""	This is monospaced
This is --stroked--	This is stroked
This is __underlined__	This is <u>underlined</u>
This is ~~waved~~	This is <u>waved</u>

Legacy HTML like	
code	output
This is bold	This is bold
This is <i>italics</i>	This is <i>italics</i>
This is monospaced	This is monospaced
This is <s>stroked</s>	This is stroked
This is <u>underlined</u>	This is <u>underlined</u>
This is <w>waved</w>	This is <u>waved</u>

And color as a bonus...	
code	output
This is <s :green>stroked</s>	This is stroked
This is <u :red>underlined</u>	This is <u>underlined</u>
This is <w :#0000FF>waved</w>	This is <u>waved</u>



23 Defining and using sprites

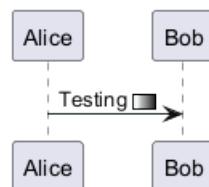
A *Sprite* is a small graphic element that can be used in diagrams.

In PlantUML, sprites are monochrome and can have either 4, 8 or 16 gray level.

To define a sprite, you have to use a hexadecimal digit between 0 and F per pixel.

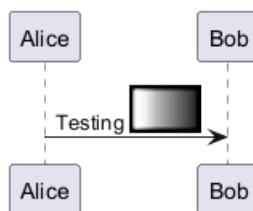
Then you can use the sprite using <\$XXX> where XXX is the name of the sprite.

```
@startuml
sprite $foo1 {
    FFFFFFFFFFFFFF
    F0123456789ABCF
    F0123456789ABCF
}
Alice -> Bob : Testing <$foo1>
@enduml
```



You can scale the sprite.

```
@startuml
sprite $foo1 {
    FFFFFFFFFFFFFF
    F0123456789ABCF
    F0123456789ABCF
}
Alice -> Bob : Testing <$foo1{scale=3}>
@enduml
```



23.1 Inline SVG sprite

You can also use inlined SVG for sprites.

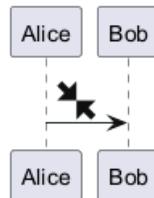


Only a tiny subset of SVG directives is possible, so you probably have to compress existing SVG files using <https://vecta.io/nano>. [Ref. GH-1066]

```
@startuml
sprite foo1 <svg width="8" height="8" viewBox="0 0 8 8">
<path d="M1 0l-1 1 1.5 1.5-1.5 1.5h4v-4l-1.5 1.5-1.5-1.5zm3 4v4l1.5-1.5 1.5 1.5 1-1-1.5-1.5 1.5-1.5h4z"/>
</svg>
```

Alice->Bob : <\$foo1*3>

@enduml

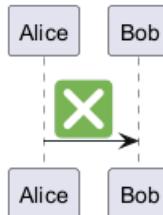


Another example:

```
@startuml
sprite foo1 <svg viewBox="0 0 36 36">
<path fill="#77B255" d="M36 32c0 2.209-1.791 4-4 4H4c-2.209 0-4-1.791-4-4V4c0-2.209 1.791-4 4-4h28c2
<path fill="#FFF" d="M21.529 18.00618.238-8.238c.977-.976.977-2.559 0-3.535-.977-.977-2.559-.977-3.535
</svg>
```

Alice->Bob : <\$foo1>

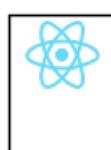
@enduml



You can also use rotation:

```
@startuml
sprite react <svg viewBox="0 0 230 230">
<circle cx="115" cy="115" r="20.5" fill="#61dafb"/>
<ellipse rx="110" ry="42" cx="115" cy="115" stroke="#61dafb" stroke-width="10" fill="none"/>
<ellipse rx="110" ry="42" cx="115" cy="115" stroke="#61dafb" stroke-width="10" fill="none" transform="rotate(45deg)"/>
<ellipse rx="110" ry="42" cx="115" cy="115" stroke="#61dafb" stroke-width="10" fill="none" transform="rotate(-45deg)"/>
</svg>

rectangle <$react{scale=0.2}>
@enduml
```



And you can use color:

```
@startuml
sprite react <svg viewBox="0 0 230 230">
<circle cx="115" cy="102" r="20.5" fill="#61dafb"/>
```

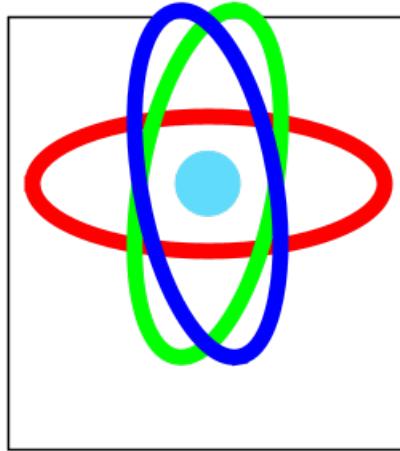


```

<ellipse rx="110" ry="42" cx="115" cy="102" stroke="#ff0000" stroke-width="10" fill="none"/>
<g transform="rotate(100 115 102)">
<ellipse rx="110" ry="42" cx="115" cy="102" stroke="#00ff00" stroke-width="10" fill="none"/>
</g>
<g transform="rotate(-100 115 102)">
<ellipse rx="110" ry="42" cx="115" cy="102" stroke="#0000ff" stroke-width="10" fill="none"/>
</g>
</svg>

rectangle <$react{scale=1}>
@enduml

```



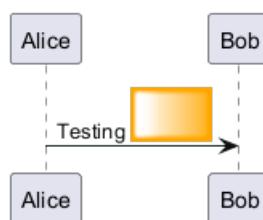
23.2 Changing colors

Although sprites are monochrome, it's possible to change their color.

```

@startuml
sprite $foo1 {
    FFFFFFFFFFFFFF
    F0123456789ABCF
    F0123456789ABCF
}
Alice -> Bob : Testing <$foo1,scale=3.4,color=orange>
@enduml

```



23.3 Encoding Sprite

To encode sprite, you can use the command line like:



```
java -jar plantuml.jar -encodesprite 16z foo.png
```

where `foo.png` is the image file you want to use (it will be converted to gray automatically).

After `-encodesprite`, you have to specify a format: 4, 8, 16, 4z, 8z or 16z.

The number indicates the gray level and the optional z is used to enable compression in sprite definition.

23.4 Importing Sprite

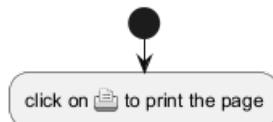
You can also launch the GUI to generate a sprite from an existing image.

Click in the menubar then on `File/Open Sprite Window`.

After copying an image into your clipboard, several possible definitions of the corresponding sprite will be displayed : you will just have to pickup the one you want.

23.5 Examples

```
@startuml
sprite $printer [15x15/8z] N0tH3W0W208HxFz_kMAhj71Hwpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwv
start
:click on <$printer> to print the page;
@enduml
```



```
@startuml
sprite $bug [15x15/16z] PKzR2i0m2BFMi15p__FEjQEjB1z27aeqCqixa8S40T7C53cKpsHpaYPDJY_12MHM-BLRyywPhrr
sprite $printer [15x15/8z] N0tH3W0W208HxFz_kMAhj71Hwpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwv
sprite $disk {
    444445566677881
    436000000009991
    43600000000ACA1
    53700000001A7A1
    53700000012B8A1
    53800000123B8A1
    63800001233C9A1
    634999AABC99B1
    744566778899AB1
    7456AAAAA99AAB1
    8566AFC228AABB1
    8567AC8118BBBB1
    867BD4433BBBBB1
    39AAAAABBBBBBC1
}

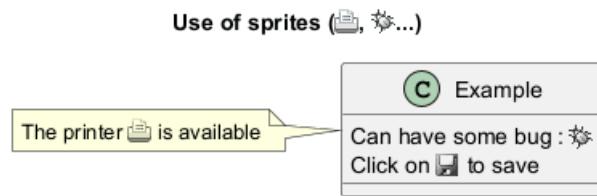
title Use of sprites (<$printer>, <$bug>...)

class Example {
Can have some bug : <$bug>
Click on <$disk> to save
}

note left : The printer <$printer> is available

@enduml
```





23.6 StdLib

The PlantUML StdLib includes a number of ready icons in various IT areas such as architecture, cloud services, logos etc. It including AWS, Azure, Kubernetes, C4, product Logos and many others. To explore these libraries:

- Browse the Github folders of PlantUML StdLib
- Browse the source repos of StdLib collections that interest you. Eg if you are interested in logos you can find that it came from gilbarbara-plantuml-sprites, and quickly find its

sprites-list. (The next section shows how to list selected sprites but unfortunately that's in grayscale whereas this custom listing is in color.)

- Study the in-depth Hitchhiker's Guide to PlantUML, eg sections Standard Library Sprites and PlantUML Stdlib Overview

23.7 Listing Sprites

You can use the `listsprites` command to show available sprites:

- Used on its own, it just shows ArchiMate sprites
- If you include some sprite libraries in your diagram, the command shows all these sprites, as explained in View all the icons with `listsprites`.

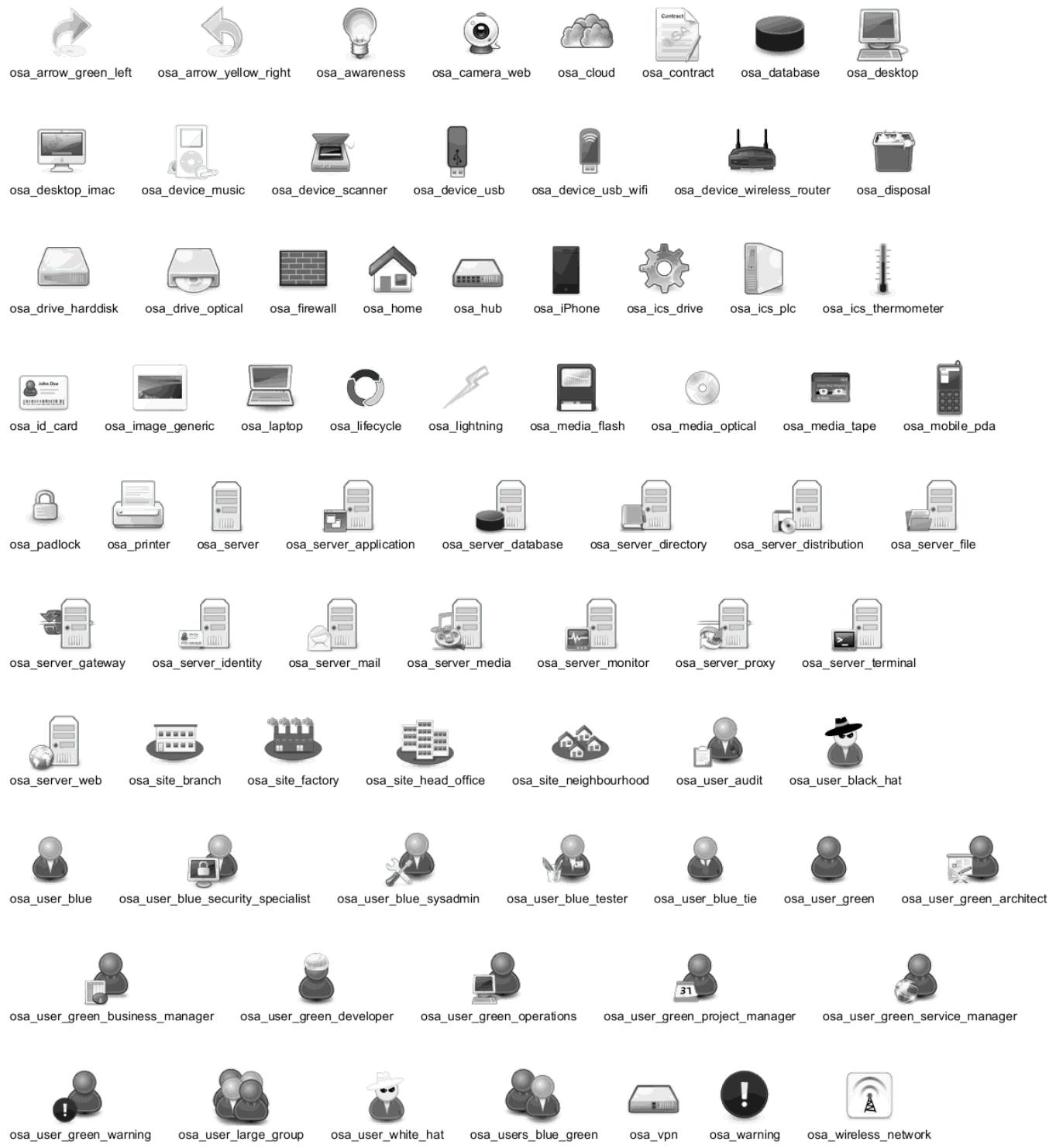
(Example from Hitchhikers Guide to PlantUML)

```
@startuml
!define osaPuml https://raw.githubusercontent.com/Crashedmind/PlantUML-opensecurityarchitecture2-icons/main
!include osaPuml/Common.puml
!include osaPuml/User/all.puml
!include osaPuml/Hardware/all.puml
!include osaPuml/Misc/all.puml
!include osaPuml/Server/all.puml
!include osaPuml/Site/all.puml

listsprites

' From The Hitchhiker's Guide to PlantUML
@enduml
```





Most collections have files called `all` that allow you to see a whole sub-collection at once. Else you need to find the sprites that interest you and include them one by one. Unfortunately, the version of a collection included in StdLib often does not have such `all` files, so as you see above we include the collection from github, not from StdLib.

All sprites are in grayscale, but most collections define specific macros that include appropriate (vendor-specific) colors.

24 Skinparam command

You can change colors and font of the drawing using the `skinparam` command.

Example:

```
skinparam backgroundColor transparent
```

Important: `skinparam` is being phased out, see comments in issue#1464. It is still supported for simple cases (and for backward compatibility), but users should migrate to `style`, which supports more complex cases.

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 Вложенные

Чтобы избежать повторений, можно вложить определения. Так, следующее определение :

```
skinparam xxxxParam1 value1
skinparam xxxxParam2 value2
skinparam xxxxParam3 value3
skinparam xxxxParam4 value4
```

строго эквивалентно:

```
skinparam xxxx {
    Param1 value1
    Param2 value2
    Param3 value3
    Param4 value4
}
```

24.3 Черно-белый

Вы можете заставить использовать черно-белый вывод с помощью команды `skinparam monochrome true`.

```
@startuml
```

```
skinparam monochrome true

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

User -> A: DoWork
activate A

A -> B: Create Request
activate B

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

B --> A: Request Created
```



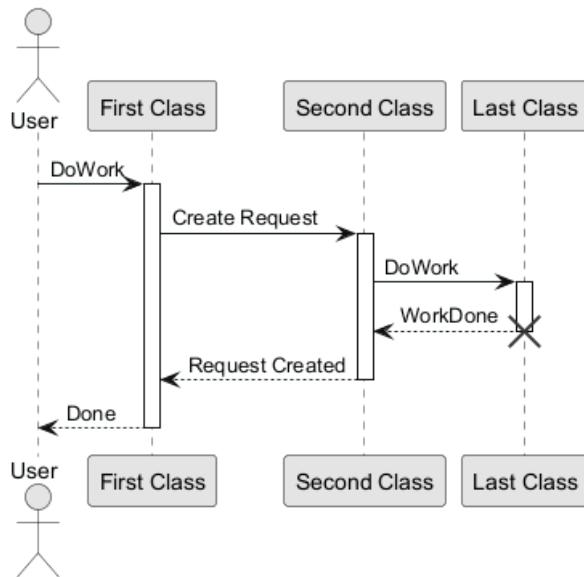
```

deactivate B

A --> User: Done
deactivate A

@enduml

```



24.4 Shadowing

You can disable the shadowing using the `skinparam shadowing false` command.

```
@startuml
```

left to right direction

```

skinparam shadowing<<no_shadow>> false
skinparam shadowing<<with_shadow>> true

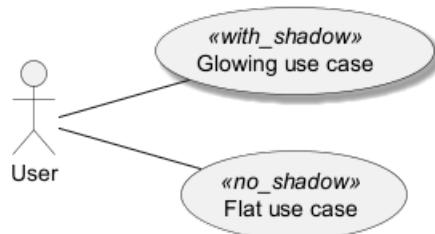
```

```

actor User
(Glowing use case) <<with_shadow>> as guc
(Flat use case) <<no_shadow>> as fuc
User -- guc
User -- fuc

```

```
@enduml
```



24.5 Reverse colors

You can force the use of a black&white output using `skinparam monochrome reverse` command. This can be useful for black background environment.

```
@startuml
```



```
skinparam monochrome reverse
```

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

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

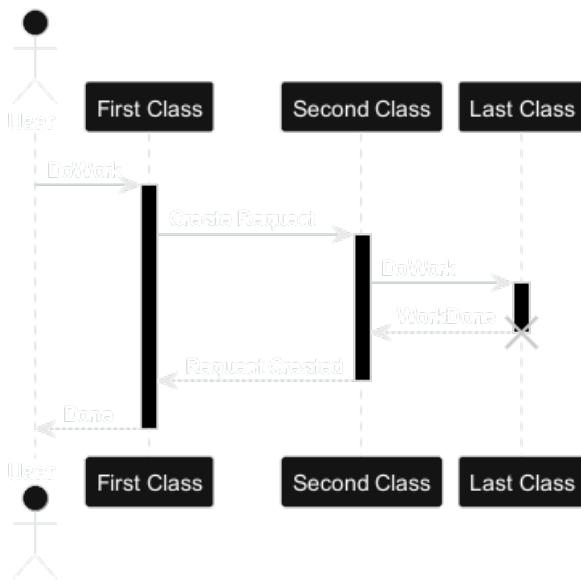
```
A -> B: Create Request
activate B
```

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

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

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

```
@enduml
```



24.6 Colors

You can use either standard color name or RGB code.

```
@startuml
colors
@enduml
```



APPLICATION	Crimson	DeepPink	Indigo	LightYellow	Navy	RoyalBlue	Turquoise
AliceBlue	Cyan	DeepSkyBlue	Ivory	Lime	OldLace	STRATEGY	Violet
AntiqueWhite	DarkBlue	DimGray	Khaki	LimeGreen	Olive	SaddleBrown	Wheat
Aqua	DarkCyan	DimGrey	Lavender	Linen	OliveDrab	Salmon	White
Aquamarine	DarkGoldenRod	DodgerBlue	LavenderBlush	MOTIVATION	Orange	SandyBrown	WhiteSmoke
Azure	DarkGray	FireBrick	LawnGreen	Magenta	OrangeRed	SeaGreen	Yellow
BUSINESS	DarkGreen	FloralWhite	LemonChiffon	Maroon	Orchid	SeaShell	YellowGreen
Beige	DarkGrey	ForestGreen	LightBlue	MediumAquaMarine	PHYSICAL	Sienna	
Bisque	DarkKhaki	Fuchsia	LightCoral	MediumBlue	PaleGoldenRod	Silver	
Black	DarkMagenta	Gainsboro	LightCyan	MediumOrchid	PaleGreen	SkyBlue	
BlanchedAlmond	DarkOliveGreen	GhostWhite	LightGoldenRodYellow	MediumPurple	PaleTurquoise	SlateBlue	
Blue	DarkOrchid	Gold	LightGray	MediumSeaGreen	PaleVioletRed	SlateGray	
BlueViolet	DarkRed	GoldenRod	LightGreen	MediumSlateBlue	PapayaWhip	SlateGrey	
Brown	DarkSalmon	Gray	LightGrey	MediumSpringGreen	PeachPuff	Snow	
BurlyWood	DarkSeaGreen	Green	LightPink	MediumTurquoise	Peru	SpringGreen	
CadetBlue	DarkSlateBlue	GreenYellow	LightSalmon	MediumVioletRed	Pink	SteelBlue	
Chartreuse	DarkSlateGray	Grey	LightSeaGreen	MidnightBlue	Plum	TECHNOLOGY	
Chocolate	DarkSlateGrey	HoneyDew	LightSkyBlue	MintCream	PowderBlue	Tan	
Coral	DarkTurquoise	HotPink	LightSlateGray	MistyRose	Purple	Teal	
CornflowerBlue	DarkViolet	IMPLEMENTATION	LightSlateGrey	Moccasin	Red	Thistle	
Cornsilk	DarkOrange	IndianRed	LightSteelBlue	NavajoWhite	RosyBrown	Tomato	

transparent can only be used for background of the image.

24.7 Font color, name and size

You can change the font for the drawing using `xxxFontColor`, `xxxFontSize` and `xxxFontName` parameters.

Example:

```
skinparam classFontColor red
skinparam classFontSize 10
skinparam classFontName Aapex
```

You can also change the default font for all fonts using `skinparam defaultFontName`.

Example:

```
skinparam defaultFontName Aapex
```

Please note the fontname is highly system dependent, so do not over use it, if you look for portability. Helvetica and Courier should be available on all systems.

A lot of parameters are available. You can list them using the following command:

```
java -jar plantuml.jar -language
```

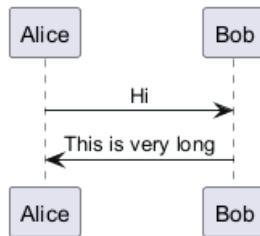
24.8 Text Alignment

Text alignment can be set to `left`, `right` or `center` in `skinparam sequenceMessageAlign`. You can also use `direction` or `reverseDirection` values to align text depending on arrow direction.

Param name	Default value	Comment
<code>sequenceMessageAlign</code>	left	Used for messages in sequence diagrams
<code>sequenceReferenceAlign</code>	center	Used for <code>ref</code> over in sequence diagrams

```
@startuml
skinparam sequenceMessageAlign center
Alice -> Bob : Hi
Bob -> Alice : This is very long
@enduml
```

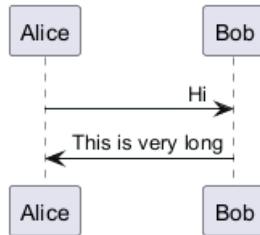




```

@startuml
skinparam sequenceMessageAlign right
Alice -> Bob : Hi
Bob -> Alice : This is very long
@enduml

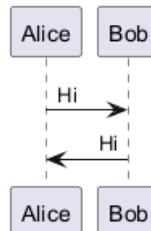
```



```

@startuml
skinparam sequenceMessageAlign direction
Alice -> Bob : Hi
Bob -> Alice: Hi
@enduml

```



24.9 Examples

```

@startuml
skinparam backgroundColor #EEEBCD
skinparam handwritten true

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

ParticipantBorderColor DeepSkyBlue
ParticipantBackgroundColor DodgerBlue
ParticipantFontName Impact
ParticipantFontSize 17
ParticipantFontColor #A9DCDF

ActorBackgroundColor aqua
ActorFontColor DeepSkyBlue
ActorFontSize 17
ActorFontName Aapex
}

```



```

}

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

User -> A: DoWork
activate A

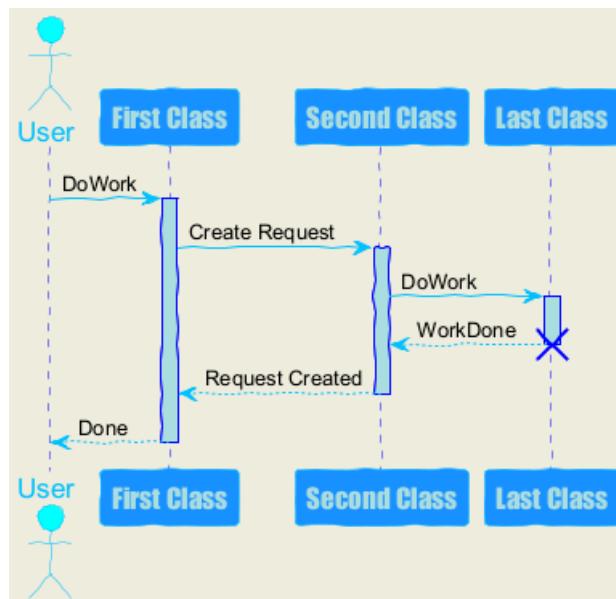
A -> B: Create Request
activate B

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

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A
@enduml

```



```

@startuml
skinparam handwritten true

skinparam actor {
BorderColor black
FontName Courier
    BackgroundColor<< Human >> Gold
}

skinparam usecase {
BackgroundColor DarkSeaGreen
BorderColor DarkSlateGray

BackgroundColor<< Main >> YellowGreen

```



```

BorderColor<< Main >> YellowGreen

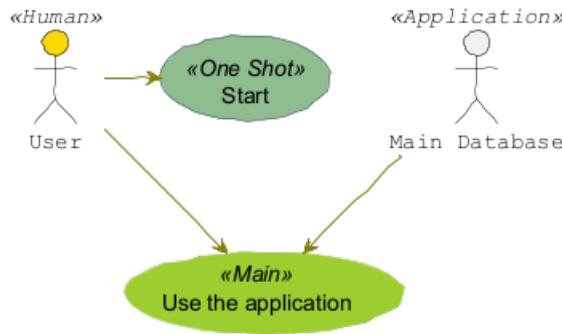
ArrowColor Olive
}

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

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

MySql --> (Use)
@enduml

```



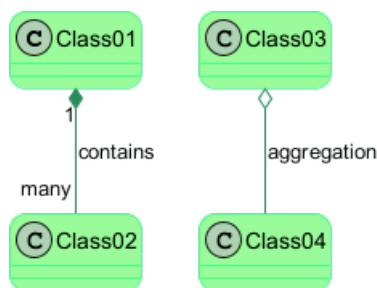
```

@startuml
skinparam roundcorner 20
skinparam class {
BackgroundColor PaleGreen
ArrowColor SeaGreen
BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen

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

Class03 o-- Class04 : aggregation
@enduml

```



```

@startuml
skinparam interface {
  backgroundColor RosyBrown
  borderColor orange
}

skinparam component {
  FontSize 13

```



```

BackgroundColor<<Apache>> LightCoral
BorderColor<<Apache>> #FF6655
FontName Courier
BorderColor black
BackgroundColor gold
ArrowFontName Impact
ArrowColor #FF6655
ArrowFontColor #777777
}

```

```

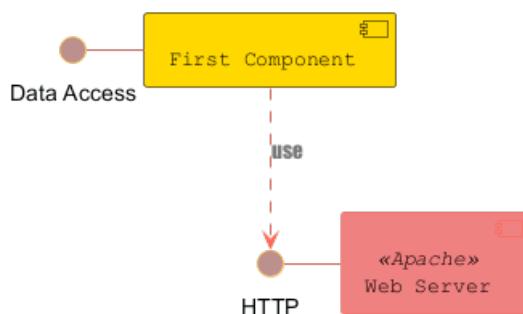
() "Data Access" as DA
[Web Server] << Apache >>

```

```

DA - [First Component]
[First Component] ..> () HTTP : use
HTTP - [Web Server]
@enduml

```



```

@startuml
[AA] <<static lib>>
[BB] <<shared lib>>
[CC] <<static lib>>

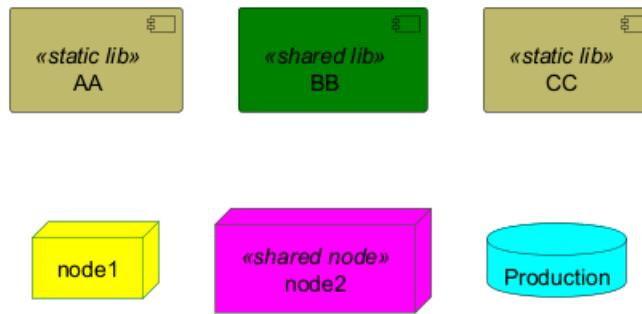
node node1
node node2 <<shared node>>
database Production

skinparam component {
    backgroundColor<<static lib>> DarkKhaki
    backgroundColor<<shared lib>> Green
}

skinparam node {
borderColor Green
backgroundColor Yellow
backgroundColor<<shared node>> Magenta
}
skinparam databaseBackgroundColor Aqua
@enduml

```





24.10 List of all skinparam parameters

You can use `-language` on the command line or generate a "diagram" with a list of all the `skinparam` parameters using :

- `help skinparams`
- `skinparameters`

24.10.1 Command Line: `-language` command

Since the documentation is not always up to date, you can have the complete list of parameters using this command:

```
java -jar plantuml.jar -language
```

24.10.2 Command: `help skinparams`

That will give you the following result, from this page (*code of this command*): `CommandHelpSkinparam.java`

```
@startuml
help skinparams
@enduml
```

Welcome to PlantUML!
 You can start with a simple UML Diagram like:
 Bob->Alice: Hello
 Or
 class Example
 You will find more information about PlantUML syntax on <https://plantuml.com>
 (Details by typing `license` keyword)



```
PlantUML 1.2025.0
[From string (line 2) ]

@startuml
help skinparams
Syntax Error?
```

24.10.3 Command: `skinparameters`

```
@startuml
skinparameters
@enduml
```





ActivityBackgroundColor	ClassFontStyle	FolderStereotypeFontSize	NoteFontStyle	SequenceDelayFontName
ActivityBorderColor	ClassStereotypeFontColor	FolderStereotypeFontStyle	NoteShadowing	SequenceDelayFontSize
ActivityBorderThickness	ClassStereotypeFontName	FooterFontColor	NoteTextAlignment	SequenceDelayFontStyle
ActivityDiamondFontColor	ClassStereotypeFontSize	FooterFontName	ObjectAttributeFontColor	SequenceDividerBorderThickness
ActivityDiamondFontName	ClassStereotypeFontStyle	FooterFontSize	ObjectAttributeFontName	SequenceDividerFontColor
ActivityDiamondFontSize	CloudFontColor	FooterFontStyle	ObjectAttributeFontSize	SequenceDividerFontName
ActivityDiamondFontStyle	CloudFontName	FrameFontColor	ObjectAttributeFontStyle	SequenceDividerFontSize
ActivityFontColor	CloudFontSize	FrameFontName	ObjectBorderThickness	SequenceDividerFontStyle
ActivityFontName	CloudFontStyle	FrameFontSize	ObjectFontColor	SequenceGroupBackgroundColor
ActivityFontSize	CloudStereotypeFontColor	FrameFontStyle	ObjectFontName	SequenceGroupBorderThickness
ActivityFontStyle	CloudStereotypeFontName	FrameStereotypeFontColor	ObjectFontSize	SequenceGroupFontColor
ActorBackgroundColor	CloudStereotypeFontSize	FrameStereotypeFontName	ObjectFontStyle	SequenceGroupFontName
ActorBorderColor	CloudStereotypeFontStyle	FrameStereotypeFontSize	ObjectStereotypeFontColor	SequenceGroupFontSize
ActorFontColor	ColorArrowSeparationSpace	FrameStereotypeFontStyle	ObjectStereotypeFontName	SequenceGroupFontStyle
ActorFontName	ComponentBorderThickness	GenericDisplay	ObjectStereotypeFontSize	SequenceGroupHeaderFontColor
ActorFontSize	ComponentFontColor	Guillemet	ObjectStereotypeFontStyle	SequenceGroupHeaderFontName
ActorFontStyle	ComponentFontName	Handwritten	PackageBorderThickness	SequenceGroupHeaderFontSize
ActorStereotypeFontColor	ComponentFontSize	HeaderFontColor	PackageFontColor	SequenceGroupHeaderFontStyle
ActorStereotypeFontName	ComponentFontStyle	HeaderFontSize	PackageFontSize	SequenceGroupHeaderFontType
ActorStereotypeFontSize	ComponentStereotypeFontColor	HeaderFontStyle	PackageFontStyle	SequenceGroupLineBorderColor
ActorStereotypeFontStyle	ComponentStereotypeFontName	HexagonBorderThickness	PackageStereotypeFontColor	SequenceGroupMessageAlignment
AgentBorderThickness	ComponentStereotypeFontSize	HexagonFontColor	PackageStereotypeFontName	SequenceGroupTextAlignment
AgentFontColor	ComponentStereotypeFontStyle	HexagonFontName	PackageStereotypeFontSize	SequenceNewpageSeparatorColor
AgentFontName	ConditionEndStyle	HexagonFontSize	PackageStereotypeFontStyle	SequenceParticipant
AgentFontSize	ConditionStyle	HexagonFontStyle	PackageStyle	SequenceParticipantBorderThickness
AgentStereotypeFontColor	ControlFontColor	HexagonStereotypeFontColor	PackageTitleAlignment	SequenceReferenceAlignment
AgentStereotypeFontName	ControlFontName	HexagonStereotypeFontName	Padding	SequenceReferenceBackgroundColor
AgentStereotypeFontSize	ControlFontSize	HexagonStereotypeFontSize	PageBorderColor	SequenceReferenceBorderThickness
AgentStereotypeFontStyle	ControlFontStyle	HexagonStereotypeFontStyle	PageExternalColor	SequenceReferenceFontColor
ArchimateBorderThickness	ControlStereotypeFontColor	HyperlinkColor	PageMargin	SequenceReferenceFontName
ArchimateFontColor	ControlStereotypeFontName	HyperlinkUnderline	ParticipantFontColor	SequenceReferenceFontSize
ArchimateFontName	ControlStereotypeFontSize	IconEMandatoryColor	ParticipantFontName	SequenceReferenceFontStyle
ArchimateFontSize	ControlStereotypeFontStyle	IconPackageBackgroundColor	ParticipantFontSize	SequenceReferenceHeaderBackgroundColor
ArchimateFontStyle	DatabaseFontColor	IconPackageColor	ParticipantFontStyle	SequenceStereotypeFontColor
ArchimateStereotypeFontColor	DatabaseFontName	IconPrivateBackgroundColor	ParticipantPadding	SequenceStereotypeFontName
ArchimateStereotypeFontName	DatabaseFontSize	IconProtectedBackgroundColor	ParticipantStereotypeFontColor	SequenceStereotypeFontSize
ArchimateStereotypeFontSize	DatabaseFontStyle	IconProtectedColor	ParticipantStereotypeFontSize	Shadowing
ArchimateStereotypeFontStyle	DatabaseStereotypeFontColor	IconPublicBackgroundColor	ParticipantStereotypeFontStyle	StackFontColor
ArrowFontColor	DatabaseStereotypeFontName	IconPublicColor	PartitionBorderThickness	StackFontName
ArrowFontName	DatabaseStereotypeFontSize	InterfaceFontColor	PartitionFontColor	StackFontSize
ArrowFontSize	DatabaseStereotypeFontStyle	InterfaceFontName	PartitionFontName	StackStereotypeFontColor
ArrowFontStyle	DefaultFontColor	InterfaceFontSize	PartitionFontSize	StackStereotypeFontName
ArrowHeadColor	DefaultFontName	InterfaceFontStyle	PartitionFontStyle	StackStereotypeFontStyle
ArrowLollipopColor	DefaultFontSize	InterfaceStereotypeFontColor	PathHoverColor	StateAttributeFontColor
ArrowMessageAlignment	DefaultFontStyle	InterfaceStereotypeFontName	PersonBorderThickness	StateAttributeFontName
ArrowThickness	DefaultMonospacedFontName	InterfaceStereotypeFontSize	PersonFontColor	StateAttributeFontSize
ArtifactFontColor	DefaultTextAlignment	InterfaceStereotypeFontStyle	PersonFontName	StateAttributeFontStyle
ArtifactFontName	DesignedBackgroundColor	LabelFontColor	PersonFontSize	StateBorderColor
ArtifactFontSize	DesignedBorderColor	LabelFontName	PersonFontStyle	StateFontColor
ArtifactFontStyle	DesignedDomainBorderThickness	LabelFontSize	PersonStereotypeFontColor	StateFontName
ArtifactStereotypeFontColor	DesignedDomainFontColor	LabelFontStyle	PersonStereotypeFontName	StateFontSize
ArtifactStereotypeFontName	DesignedDomainFontName	LabelStereotypeFontColor	PersonStereotypeFontSize	StateFontStyle
ArtifactStereotypeFontSize	DesignedDomainFontSize	LabelStereotypeFontName	PersonStereotypeFontStyle	StateMessageAlignment
ArtifactStereotypeFontStyle	DesignedDomainFontStyle	LabelStereotypeFontSize	QueueBorderThickness	StereotypePosition
BackgroundColor	DesignedDomainStereotypeFontColor	LabelStereotypeFontStyle	QueueFontColor	StorageFontColor
BiddableBackgroundColor	DesignedDomainStereotypeFontName	LegendBorderThickness	QueueFontName	StorageFontName
BiddableBorderColor	DesignedDomainStereotypeFontSize	LegendFontColor	QueueFontSize	StorageFontSize
BoundaryFontColor	DesignedDomainStereotypeFontStyle	LegendFontName	QueueFontStyle	StorageFontStyle
BoundaryFontName	DiagramBorderColor	LegendFontSize	QueueStereotypeFontColor	StorageStereotypeFontColor
BoundaryFontSize	DiagramBorderThickness	LegendFontStyle	QueueStereotypeFontName	StorageStereotypeFontName
BoundaryFontStyle	DomainBackgroundColor	LexicalBackgroundColor	QueueStereotypeFontSize	StorageStereotypeFontSize
BoundaryStereotypeFontColor	DomainBorderColor	LexicalBorderColor	QueueStereotypeFontStyle	StorageStereotypeFontStyle
BoundaryStereotypeFontName	DomainBorderThickness	LifelineStrategy	Ranksep	Style
BoundaryStereotypeFontSize	DomainFontColor	Linetype	RectangleBorderThickness	SvgLinkTarget
BoundaryStereotypeFontStyle	DomainFontName	MachineBackgroundColor	RectangleFontColor	SwimlaneBorderThickness
BoxPadding	DomainFontSize	MachineBorderColor	RectangleFontName	SwimlaneTitleFontColor
CaptionFontColor	DomainFontStyle	MachineBorderThickness	RectangleFontSize	SwimlaneTitleFontName
CaptionFontName	DomainStereotypeFontColor	MachineFontColor	RectangleFontStyle	SwimlaneTitleFontSize
CaptionFontSize	DomainStereotypeFontName	MachineFontName	RectangleStereotypeFontColor	SwimlaneTitleFontStyle
CaptionFontStyle	DomainStereotypeFontSize	MachineFontSize	RectangleStereotypeFontName	SwimlaneWidth
CardBorderThickness	DomainStereotypeFontStyle	MachineFontStyle	RectangleStereotypeFontSize	SwimlaneWrapTitleWidth
CardFontColor	Dpi	MachineStereotypeFontColor	RequirementBackgroundColor	TabSize
CardFontName	EntityFontColor	MachineStereotypeFontName	RequirementBorderColor	TimingFontColor
CardFontSize	EntityFontName	MachineStereotypeFontSize	RequirementBorderThickness	TimingFontName
CardFontStyle	EntityFontSize	MachineStereotypeFontStyle	RequirementColor	TimingFontSize
CardStereotypeFontColor	EntityFontStyle	MachineStereotypeFontType	RequirementFontColor	TimingFontStyle
CardStereotypeFontName	EntityStereotypeFontColor	MaxAsciiMessageLength	RequirementFontSize	TitleBorderRoundCorner
CardStereotypeFontSize	EntityStereotypeFontName	MaxMessageSize	RequirementFontName	TitleBorderThickness
CardStereotypeFontStyle	EntityStereotypeFontSize	MinClassWidth	RequirementFontSize	TitleFontColor
CircledCharacterFontColor	EntityStereotypeFontStyle	Monochrome	RequirementFontStyle	TitleFontSize
CircledCharacterFontName	FileFontColor	NodeFontColor	RequirementStereotypeFontColor	TitleFontStyle
CircledCharacterFontSize	FileFontName	NodeFontName	RequirementStereotypeFontName	UsecaseBorderThickness
CircledCharacterFontStyle	FileFontSize	NodeFontSize	RequirementStereotypeFontSize	UsecaseFontColor
CircledCharacterRadius	FileFontStyle	NodeFontStyle	RequirementStereotypeFontStyle	UsecaseFontName
ClassAttributeFontColor	FileStereotypeFontColor	NodeStereotypeFontColor	ResponseMessageBelowArrow	UsecaseFontSize
ClassAttributeFontName	FileStereotypeFontName	NodeStereotypeFontName	RoundCorner	UsecaseFontStyle
ClassAttributeFontSize	FileStereotypeFontSize	NodeStereotypeFontSize	SameClassWidth	SequenceActorBorderThickness
ClassAttributeFontStyle	FileStereotypeFontStyle	NodeStereotypeFontStyle	SequenceArrowThickness	SequenceBoxBorderColor
ClassAttributeIconSize	FixCircleLabelOverlapping	NodeStereotypeFontType	SequenceBoxFontColor	SequenceBoxFontSize
ClassBackgroundColor	FolderFontColor	Nodesep	SequenceBoxFontName	SequenceBoxFontStyle
ClassBorderColor	FolderFontName	NoteBackgroundColor	SequenceBoxFontSize	UsecaseStereotypeFontSize
ClassBorderThickness	FolderFontSize	NoteBorderColor	SequenceBoxFontStyle	UsecaseStereotypeFontStyle
ClassFontColor	FolderFontStyle	NoteFontSize	SequenceDelayFontColor	WrapWidth
ClassFontName	FolderStereotypeFontColor	NoteFontName		
ClassFontSize	FolderStereotypeFontName	NoteFontSize		



24.10.4 All Skin Parameters on the Ashley's PlantUML Doc

You can also view each `skinparam` parameters with its results displayed at the page All Skin Parameters of Ashley's PlantUML Doc:

- <https://plantuml-documentation.readthedocs.io/en/latest/formatting/all-skin-params.html>.



25 Preprocessing

Some preprocessing capabilities are included in **PlantUML**, and available for *all* diagrams.

Those functionalities are very similar to the C language preprocessor, except that the special character # has been changed to the exclamation mark !.

25.1 Variable definition [=, ?=]

Although this is not mandatory, we highly suggest that variable names start with a \$.

There are three types of data:

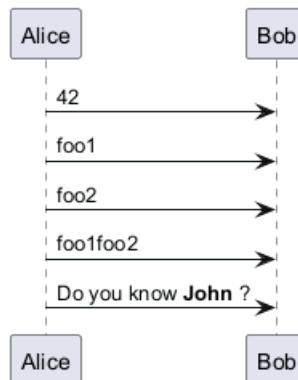
- **Integer number** (*int*);
- **String** (*str*) - these must be surrounded by single quote or double quote;
- **JSON** (*JSON*) - either JSON Array or JSON Object or JSON value created by %str2json.

(*for JSON variable definition and usage, see more details on Preprocessing-JSON page*)

Variables created outside function are **global**, that is you can access them from everywhere (including from functions). You can emphasize this by using the optional **global** keyword when defining a variable.

```
@startuml
!$a = 42
!$ab = "foo1"
!$cd = "foo2"
!$ef = $ab + $cd
!$foo = { "name": "John", "age" : 30 }
```

```
Alice -> Bob : $a
Alice -> Bob : $ab
Alice -> Bob : $cd
Alice -> Bob : $ef
Alice -> Bob : Do you know **$foo.name** ?
@enduml
```



You can also assign a value to a variable, only if it is not already defined, with the syntax: !\$a ?= "foo"

```
@startuml
Alice -> Bob : 1. **$name** should be empty

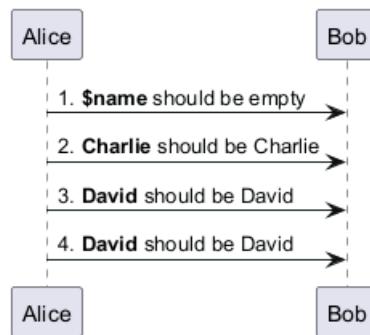
$name ?= "Charlie"
Alice -> Bob : 2. **$name** should be Charlie

$name = "David"
Alice -> Bob : 3. **$name** should be David

$name ?= "Ethan"
Alice -> Bob : 4. **$name** should be David
```



```
@enduml
```



25.2 Boolean expression

25.2.1 Boolean representation [0 is false]

There is not real boolean type, but PlantUML use this integer convention:

- Integer 0 means `false`
- and any non-null number (as 1) or any string (as "1", or even "0") means `true`.

[Ref. QA-9702]

25.2.2 Boolean operation and operator [&&, ||, 0]

You can use boolean expression, in the test, with :

- *parenthesis ()*;
- *and operator &&*;
- *or operator ||*.

(See next example, within `if` test.)

25.2.3 Boolean builtin functions [%false(), %true(), %not(<exp>), %boolval(<exp>)]

For convenience, you can use those boolean builtin functions:

- `%false()`
- `%true()`
- `%not(<exp>)`
- `%boolval(<exp>)`

[See also Builtin functions] [Ref. PR-1873]

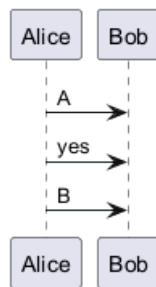
25.3 Conditions [`!if`, `!else`, `!elseif`, `!endif`]

- You can use expression in condition.
- `else` and `elseif` are also implemented

```
@startuml
!$a = 10
!$ijk = "foo"
Alice -> Bob : A
!if ($ijk == "foo") && ($a+10>=4)
Alice -> Bob : yes
!else
Alice -> Bob : This should not appear
!endif
```



```
Alice -> Bob : B
@enduml
```



25.4 While loop [!while, !endwhile]

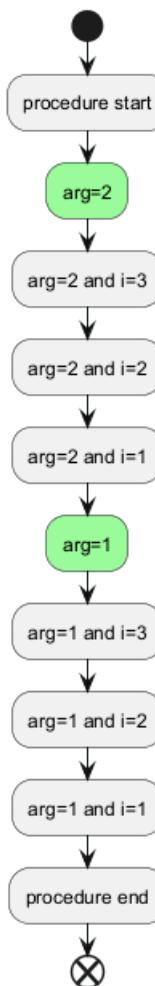
You can use `!while` and `!endwhile` keywords to have repeat loops.

25.4.1 While loop (on Activity diagram)

```
@startuml
!procedure $foo($arg)
:procedure start;
!while $arg!=0
  !$i=3
  #palegreen:arg=$arg;
  !while $i!=0
    :arg=$arg and i=$i;
    !$i = $i - 1
  !endwhile
  !$arg = $arg - 1
!endwhile
:procedure end;
!endprocedure

start
$foo(2)
end
@enduml
```





[Adapted from QA-10838]

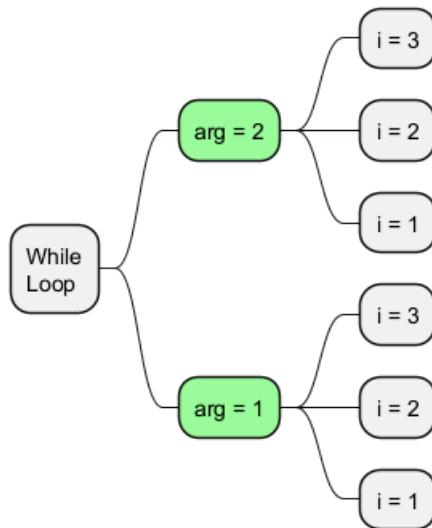
25.4.2 While loop (on Mindmap diagram)

```

@startmindmap
!procedure $foo($arg)
  !$i=3
  **[#palegreen] arg = $arg
  !while $i!=0
    *** i = $i
    !$i = $i - 1
  !endwhile
  !$arg = $arg - 1
!endwhile
!endprocedure

*:While
Loop;
$foo(2)
@endmindmap
  
```





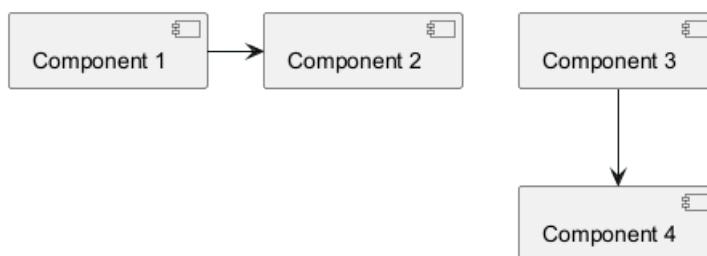
25.4.3 While loop (on Component/Deployment diagram)

```

@startuml
!procedure $foo($arg)
  !while $arg!=0
    [Component $arg] as $arg
    !$arg = $arg - 1
  !endwhile
!endprocedure

$foo(4)

1->2
3->4
@enduml
  
```



[Ref. QA-14088]

25.5 Procedure [!procedure, !endprocedure]

- Procedure names *should* start with a \$
- Argument names *should* start with a \$
- Procedures can call other procedures

Example:

```

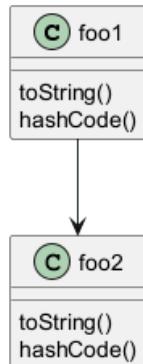
@startuml
!procedure $msg($source, $destination)
  $source --> $destination
!endprocedure
  
```



```
!procedure $init_class($name)
  class $name {
    $addCommonMethod()
  }
!endprocedure
```

```
!procedure $addCommonMethod()
  toString()
  hashCode()
!endprocedure
```

```
$init_class("foo1")
$init_class("foo2")
$msg("foo1", "foo2")
@enduml
```



Variables defined in procedures are **local**. It means that the variable is destroyed when the procedure ends.

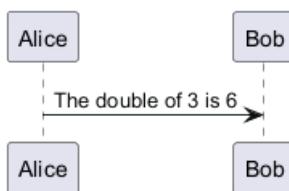
25.6 Return function [`!function`, `!endfunction`]

A return function does not output any text. It just define a function that you can call:

- directly in variable definition or in diagram text
- from other return functions
- from procedures
- Function name *should* start with a \$
- Argument names *should* start with a \$

```
@startuml
!function $double($a)
!return $a + $a
!endfunction
```

```
Alice -> Bob : The double of 3 is $double(3)
@enduml
```

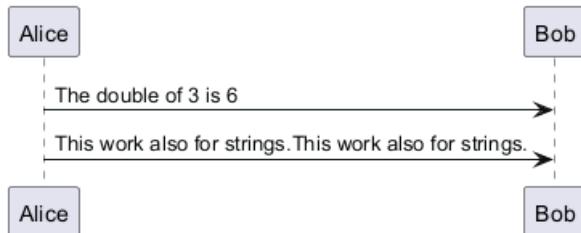


It is possible to shorten simple function definition in one line:



```
@startuml
!function $double($a) !return $a + $a

Alice -> Bob : The double of 3 is $double(3)
Alice -> Bob : $double("This work also for strings.")
@enduml
```

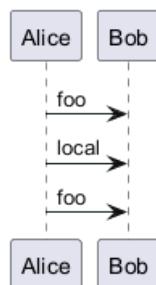


As in procedure (void function), variable are local by default (they are destroyed when the function is exited). However, you can access to global variables from function. However, you can use the `local` keyword to create a local variable if ever a global variable exists with the same name.

```
@startuml
!function $dummy()
!local $ijk = "local"
!return "Alice -> Bob : " + $ijk
!endfunction

!global $ijk = "foo"

Alice -> Bob : $ijk
$dummy()
Alice -> Bob : $ijk
@enduml
```



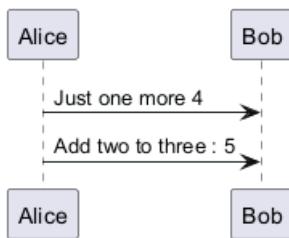
25.7 Default argument value

In both procedure and return functions, you can define default values for arguments.

```
@startuml
!function $inc($value, $step=1)
!return $value + $step
!endfunction

Alice -> Bob : Just one more $inc(3)
Alice -> Bob : Add two to three : $inc(3, 2)
@enduml
```



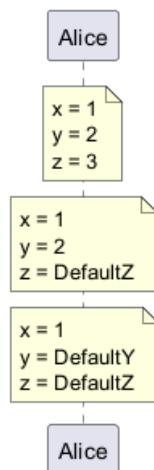


Only arguments at the end of the parameter list can have default values.

```

@startuml
!procedure defaulttest($x, $y="DefaultY", $z="DefaultZ")
note over Alice
  x = $x
  y = $y
  z = $z
end note
!endprocedure

defaulttest(1, 2, 3)
defaulttest(1, 2)
defaulttest(1)
@enduml
  
```



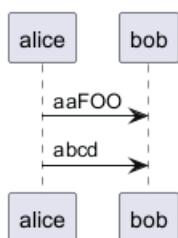
25.8 Unquoted procedure or function [!unquoted]

By default, you have to put quotes when you call a function or a procedure. It is possible to use the `unquoted` keyword to indicate that a function or a procedure does not require quotes for its arguments.

```

@startuml
!unquoted function id($text1, $text2="FOO") !return $text1 + $text2

alice -> bob : id(aa)
alice -> bob : id(ab,cd)
@enduml
  
```



25.9 Keywords arguments

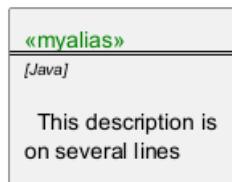
Like in Python, you can use keywords arguments :

```
@startuml
```

```
!unquoted procedure $element($alias, $description="", $label="", $technology="", $size=12, $colour="g
rectangle $alias as "
<color:$colour><$alias></color>
==$label==
//<size:$size>[$technology]</size>//

$description"
!endprocedure

$element(myalias, "This description is %newline()on several lines", $size=10, $technology="Java")
@enduml
```



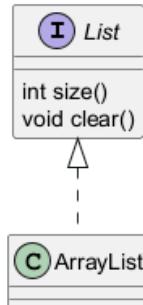
25.10 Including files or URL [**!include**, **!include_many**, **!include_once**]

Use the `!include` directive to include file in your diagram. Using URL, you can also include file from Internet/Intranet. Protected Internet resources can also be accessed, this is described in URL authentication.

Imagine you have the very same class that appears in many diagrams. Instead of duplicating the description of this class, you can define a file that contains the description.

```
@startuml
```

```
interface List
List : int size()
List : void clear()
List <|.. ArrayList
@enduml
```



File List.iuml

```
interface List
List : int size()
List : void clear()
```

The file `List.iuml` can be included in many diagrams, and any modification in this file will change all diagrams that include it.



You can also put several @startuml/@enduml text block in an included file and then specify which block you want to include adding !0 where 0 is the block number. The !0 notation denotes the first diagram.

For example, if you use !include foo.txt!1, the second @startuml/@enduml block within foo.txt will be included.

You can also put an id to some @startuml/@enduml text block in an included file using @startuml(id=MY OWN_ID) syntax and then include the block adding !MY OWN_ID when including the file, so using something like !include foo.txt!MY OWN_ID.

By default, a file can only be included once. You can use !include_many instead of !include if you want to include some file several times. Note that there is also a !include_once directive that raises an error if a file is included several times.

25.11 Including Subpart [!startsub, !endsub, !includesub]

You can also use !startsub NAME and !endsub to indicate sections of text to include from other files using !includesub. For example:

file1.puml:

```
@startuml

A -> A : stuff1
!startsub BASIC
B -> B : stuff2
!endsub
C -> C : stuff3
!startsub BASIC
D -> D : stuff4
!endsub
@enduml
```

file1.puml would be rendered exactly as if it were:

```
@startuml
```

```
A -> A : stuff1
B -> B : stuff2
C -> C : stuff3
D -> D : stuff4
@enduml
```

However, this would also allow you to have another file2.puml like this:

file2.puml

```
@startuml
```

```
title this contains only B and D
!includesub file1.puml!BASIC
@enduml
```

This file would be rendered exactly as if:

```
@startuml
```

```
title this contains only B and D
B -> B : stuff2
D -> D : stuff4
@enduml
```



25.12 Builtin functions [%]

Some functions are defined by default. Their name starts by %

Name	Description
%boolval	Convert a value (<i>String</i> , <i>Integer</i> , <i>JSON value</i>) to boolean value
%call_user_func	Invoke a return function by its name with given arguments.
%chr	Return a character from a give Unicode value
%darken	Return a darken color of a given color with some ratio
%date	Retrieve current date. You can provide an optional format for the date You can provide another optional time (on epoch format)
%dec2hex	Return the hexadecimal string (<i>String</i>) of a decimal value (<i>Int</i>)
%dirname	Retrieve current dirname
%feature	Check if some feature is available in the current PlantUML running version
%false	Return always false
%file_exists	Check if a file exists on the local filesystem
%filename	Retrieve current filename
%function_exists	Check if a function exists
%get_all_theme	Retreive a JSON Array of all PlantUML theme
%get_all_stdlib	Retreive a JSON Array of all PlantUML stdlib names
%get_all_stdlib	Retreive a JSON Object of all PlantUML stdlib information
%get_variable_value	Retrieve some variable value
%getenv	Retrieve environment variable value
%hex2dec	Return the decimal value (<i>Int</i>) of a hexadeciml string (<i>String</i>)
%hsl_color	Return the RGBa color from a HSL color %hsl_color(h, s, l) or %hsl_color(h, s, l, a)
%intval	Convert a String to Int
%invoke_procedure	Dynamically invoke a procedure by its name, passing optional arguments to the called procedure.
%is_dark	Check if a color is a dark one
%is_light	Check if a color is a light one
%lighten	Return a lighten color of a given color with some ratio
%load_json	Load JSON data from local file or external URL
%lower	Return a lowercase string
%mod	Return the remainder of division of two integers (modulo division)
%newline	Return a newline
%not	Return the logical negation of an expression
%now	Return the current epoch time
%ord	Return a Unicode value from a given character
%lighten	Return a lighten color of a given color with some ratio
%random()	Return randomly the integer 0 or 1
%random(n)	Return randomly an interger between 0 and n - 1
%random(min, max)	Return randomly an interger between min and max - 1
%reverse_color	Reverse a color using RGB
%reverse_hsluv_color	Reverse a color using HSLuv
%set_variable_value	Set a global variable
%size	Return the size of any string or JSON structure
%splitstr	Split a string into an array based on a specified delimiter.
%splitstr_regex	Split a string into an array based on a specified REGEX.
%string	Convert an expression to String
%strlen	Calculate the length of a String
%strpos	Search a substring in a string
%substr	Extract a substring. Takes 2 or 3 arguments
%true	Return always true
%upper	Return an uppercase string
%variable_exists	Check if a variable exists
%version	Return PlantUML current version

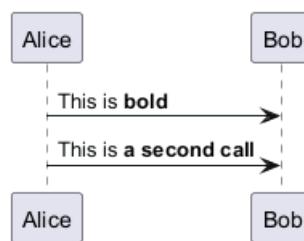


25.13 Logging [!log]

You can use !log to add some log output when generating the diagram. This has no impact at all on the diagram itself. However, those logs are printed in the command line's output stream. This could be useful for debug purpose.

```
@startuml
!function bold($text)
!$result = "<b>" + $text + "</b>"
!log Calling bold function with $text. The result is $result
!return $result
!endfunction

Alice -> Bob : This is bold("bold")
Alice -> Bob : This is bold("a second call")
@enduml
```

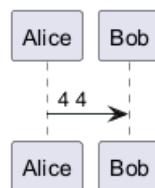


25.14 Memory dump [!dump_memory]

You can use !dump_memory to dump the full content of the memory when generating the diagram. An optional string can be put after !dump_memory. This has no impact at all on the diagram itself. This could be useful for debug purpose.

```
@startuml
!function $inc($string)
!$val = %intval($string)
!log value is $val
!dump_memory
!return $val+1
!endfunction

Alice -> Bob : 4 $inc("3")
!unused = "foo"
!dump_memory EOF
@enduml
```



25.15 Assertion [!assert]

You can put assertions in your diagram.

```
@startuml
Alice -> Bob : Hello
!assert %strpos("abcdef", "cd")==3 : "This always fails"
@enduml
```



Welcome to PlantUML!

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on <https://plantuml.com>
(Details by typing license keyword)



```
PlantUML 1.2025.0
[From string (line 3) ]

@startuml
Alice -> Bob : Hello
!assert %strpos("abcdef", "cd") == 3 : "This always fails"
Assertion error : This always fails
```

25.16 Building custom library [!import, !include]

It's possible to package a set of included files into a single .zip or .jar archive. This single zip/jar can then be imported into your diagram using !import directive.

Once the library has been imported, you can !include file from this single zip/jar.

Example:

```
@startuml

!import /path/to/customLibrary.zip
' This just adds "customLibrary.zip" in the search path

!include myFolder/myFile.iuml
' Assuming that myFolder/myFile.iuml is located somewhere
' either inside "customLibrary.zip" or on the local filesystem

...
```

25.17 Search path

You can specify the java property plantuml.include.path in the command line.

For example:

```
java -Dplantuml.include.path="c:/mydir" -jar plantuml.jar atest1.txt
```

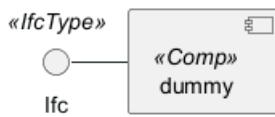
Note the this -D option has to put before the -jar option. -D options after the -jar option will be used to define constants within plantuml preprocessor.

25.18 Argument concatenation [##]

It is possible to append text to a macro argument using the ## syntax.

```
@startuml
!unquoted procedure COMP_TEXTGENCOMP(name)
[name] << Comp >>
interface Ifc << IfcType >> AS name##Ifc
name##Ifc - [name]
!endprocedure
COMP_TEXTGENCOMP(dummy)
@enduml
```





25.19 Dynamic invocation [%invoke_procedure(), %call_user_func()]

You can dynamically invoke a procedure using the special `%invoke_procedure()` procedure. This procedure takes as first argument the name of the actual procedure to be called. The optional following arguments are copied to the called procedure.

For example, you can have:

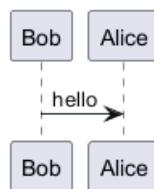
```

@startuml
!procedure $go()
    Bob -> Alice : hello
!endprocedure

!$wrapper = "$go"

%invoke_procedure($wrapper)
@enduml

```

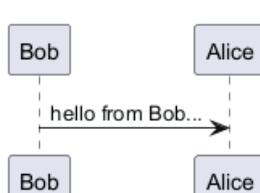


```

@startuml
!procedure $go($txt)
    Bob -> Alice : $txt
!endprocedure

%invoke_procedure("$go", "hello from Bob...")
@enduml

```



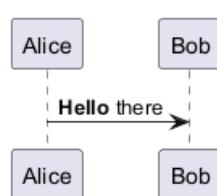
For return functions, you can use the corresponding special function `%call_user_func()`:

```

@startuml
!function bold($text)
!return "<b>" + $text + "</b>"
!endfunction

Alice -> Bob : %call_user_func("bold", "Hello") there
@enduml

```



25.20 Evaluation of addition depending of data types [+]

Evaluation of \$a + \$b depending of type of \$a or \$b

```
@startuml
title
<#LightBlue>|= |= $a |= $b |= <U+0025>string($a + $b) |
<#LightGray>| type | str | str | str (concatenation) |
| example |= "a" |= "b" |= %string("a" + "b") |
<#LightGray>| type | str | int | str (concatenation) |
| ex.|= "a" |= 2 |= %string("a" + 2) |
<#LightGray>| type | str | int | str (concatenation) |
| ex.|= 1 |= "b" |= %string(1 + "b") |
<#LightGray>| type | bool | str | str (concatenation) |
| ex.|= <U+0025>true() |= "b" |= %string(%true() + "b") |
<#LightGray>| type | str | bool | str (concatenation) |
| ex.|= "a" |= <U+0025>false() |= %string("a" + %false()) |
<#LightGray>| type | int | int | int (addition of int) |
| ex.|= 1 |= 2 |= %string(1 + 2) |
<#LightGray>| type | bool | int | int (addition) |
| ex.|= <U+0025>true() |= 2 |= %string(%true() + 2) |
<#LightGray>| type | int | bool | int (addition) |
| ex.|= 1 |= <U+0025>false() |= %string(1 + %false()) |
<#LightGray>| type | int | int | int (addition) |
| ex.|= 1 |= <U+0025>intval("2") |= %string(1 + %intval("2")) |
end title
@enduml
```

	\$a	\$b	%string(\$a + \$b)
type	str	str	str (concatenation)
example	"a"	"b"	ab
type	str	int	str (concatenation)
ex.	"a"	2	a2
type	str	int	str (concatenation)
ex.	1	"b"	1b
type	bool	str	str (concatenation)
ex.	%true()	"b"	1b
type	str	bool	str (concatenation)
ex.	"a"	%false()	a0
type	int	int	int (addition of int)
ex.	1	2	3
type	bool	int	int (addition)
ex.	%true()	2	3
type	int	bool	int (addition)
ex.	1	%false()	1
type	int	int	int (addition)
ex.	1	%intval("2")	3

25.21 Preprocessing JSON

You can extend the functionality of the current Preprocessing with JSON Preprocessing features:

- JSON Variable definition
- Access to JSON data
- Loop over JSON array

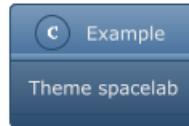
(See more details on [Preprocessing-JSON page](#))

25.22 Including theme [!theme]

Use the !theme directive to change the default theme of your diagram.



```
@startuml
!theme spacelab
class Example {
    Theme spacelab
}
@enduml
```



You will find more information on the dedicated page.

25.23 Migration notes

The current preprocessor is an update from some legacy preprocessor.

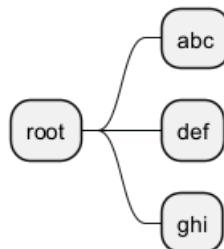
Even if some legacy features are still supported with the actual preprocessor, you should not use them any more (they might be removed in some long term future).

- You should not use `!define` and `!definelong` anymore. Use `!function`, `!procedure` or variable definition instead.
 - `!define` should be replaced by `return !function`
 - `!definelong` should be replaced by `!procedure`.
- `!include` now allows multiple inclusions : you don't have to use `!include_many` anymore
- `!include` now accepts a URL, so you don't need `!includeurl`
- Some features (like `%date%`) have been replaced by builtin functions (for example `%date()`)
- When calling a legacy `!definelong` macro with no arguments, you do have to use parenthesis. You have to use `my_own_definelong()` because `my_own_definelong` without parenthesis is not recognized by the new preprocessor.

Please contact us if you have any issues.

25.24 %splitstr builtin function

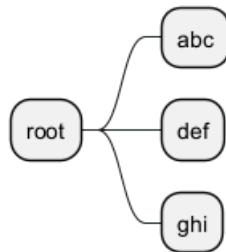
```
@startmindmap
!$list = %splitstr("abc~def~ghi", "~")
* root
!foreach $item in $list
    ** $item
!endfor
@endmindmap
```



Similar to:



```
@startmindmap
* root
!foreach $item in ["abc", "def", "ghi"]
  ** $item
!endfor
@endmindmap
```

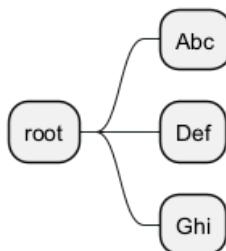


[Ref. QA-15374]

25.25 %splitstr_regex builtin function

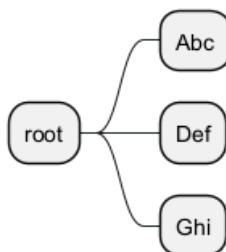
```
@startmindmap
$list = %splitstr_regex("AbcDefGhi", "(?=[A-Z])")

* root
!foreach $item in $list
  ** $item
!endfor
@endmindmap
```



Similar to:

```
@startmindmap
* root
!foreach $item in ["Abc", "Def", "Ghi"]
  ** $item
!endfor
@endmindmap
```



[Ref. QA-18827]

25.26 %get_all_theme builtin function

You can use the `%get_all_theme()` builtin function to retrieve a JSON array of all PlantUML theme.

```
@startjson  
%get_all_theme()  
@endjson
```

none
amiga
aws-orange
black-knight
bluegray
blueprint
carbon-gray
cerulean
cerulean-outline
cloudscape-design
crt-amber
crt-green
cyborg
cyborg-outline
hacker
lightgray
mars
materia
materia-outline
metal
mimeograph
minty
mono
plain
reddress-darkblue
reddress-darkgreen
reddress-darkorange
reddress-darkred
reddress-lightblue
reddress-lightgreen
reddress-lightorange
reddress-lightred
sandstone
silver
sketchy
sketchy-outline
spacelab
spacelab-white
sunlust
superhero
superhero-outline
toy
united
vibrant



[from version 1.2024.4]

25.27 %get_all_stdlib builtin function

25.27.1 Compact version (only standard library name)

You can use the %get_all_stdlib() builtin function to retrieve a JSON array of all PlantUML stdlib names.

```
@startjson  
%get_all_stdlib()  
@endjson
```

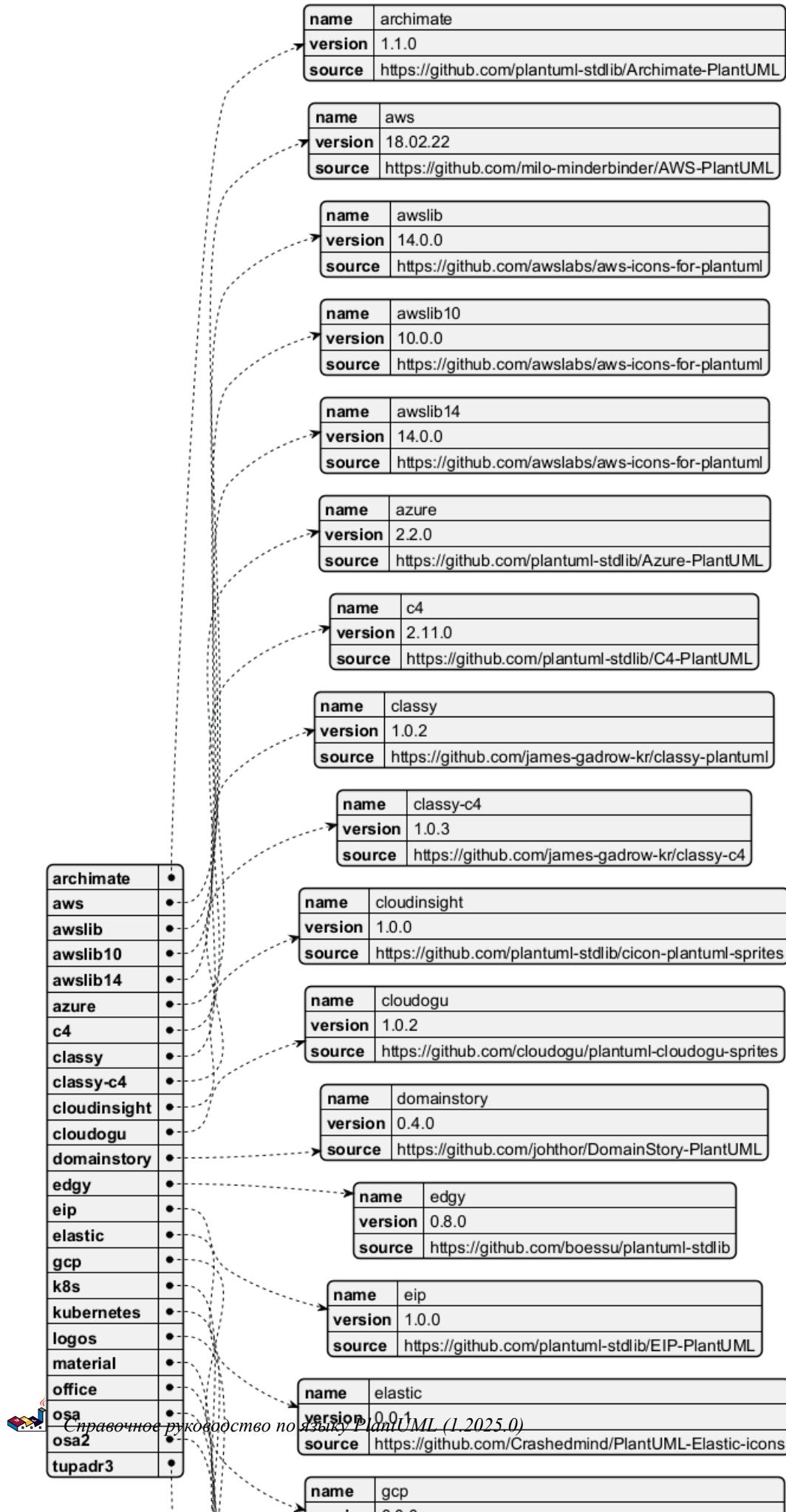
archimate
aws
awslib
awslib10
awslib14
azure
c4
classy
classy-c4
cloudinsight
cloudogu
domainstory
edgy
eip
elastic
gcp
k8s
kubernetes
logos
material
office
osa
osa2
tupadr3

25.27.2 Detailed version (with version and source)

With whatever parameter, you can use %get_all_stdlib(detailed) to retrieve a JSON object of all PlantUML stdlib.

```
@startjson  
%get_all_stdlib(detailed)  
@endjson
```





[from version 1.2024.4]

25.28 %random builtin function

You can use the %random builtin function to retrieve a random integer.

Nb param.	Input	Output
0	%random()	returns 0 or 1
1	%random(n)	returns an interger between 0 and n - 1
2	%random(min, max)	returns an interger between min and max - 1

```
@startcreole
| Nb param. | Input | Output |
| 0 | <U+0025>random() | %random() |
| 1 | <U+0025>random(5) | %random(5) |
| 2 | <U+0025>random(7, 15) | %random(7, 15) |
@endcreole
```

Nb param.	Input	Output
0	%random()	1
1	%random(5)	4
2	%random(7, 15)	9

[from version 1.2024.2]

25.29 %boolval builtin function

You can use the %boolval builtin function to manage boolean value.

```
@startcreole
<#ccc>|= Input |= Output |
| <U+0025>boolval(1) | %boolval(1) |
| <U+0025>boolval(0) | %boolval(0) |
| <U+0025>boolval(<U+0025>true()) | %boolval(%true()) |
| <U+0025>boolval(<U+0025>false()) | %boolval(%false()) |
| <U+0025>boolval(true) | %boolval(true) |
| <U+0025>boolval(false) | %boolval(false) |
| <U+0025>boolval(TRUE) | %boolval(TRUE) |
| <U+0025>boolval(FALSE) | %boolval(FALSE) |
| <U+0025>boolval("true") | %boolval("true") |
| <U+0025>boolval("false") | %boolval("false") |
| <U+0025>boolval(<U+0025>str2json("true")) | %boolval(%str2json("true")) |
| <U+0025>boolval(<U+0025>str2json("false")) | %boolval(%str2json("false")) |
@endcreole
```

Input	Output
%boolval(1)	1
%boolval(0)	0
%boolval(%true())	1
%boolval(%false())	0
%boolval(true)	1
%boolval(false)	0
%boolval(TRUE)	1
%boolval(FALSE)	0
%boolval("true")	1
%boolval("false")	0
%boolval(%str2json("true"))	1
%boolval(%str2json("false"))	0

[Ref. PR-1873, from version 1.2024.7]



26 Unicode

The PlantUML language use *letters* to define actor, usecase and so on.

But *letters* are not only A-Z latin characters, it could be *any kind of letter from any language*.

26.1 Examples

```
@startuml
skinparam handwritten true
skinparam backgroundColor #EEEBDC

actor 使用者
participant "頭等艙" as A
participant "第二類" as B
participant "最後一堂課" as 別的東西

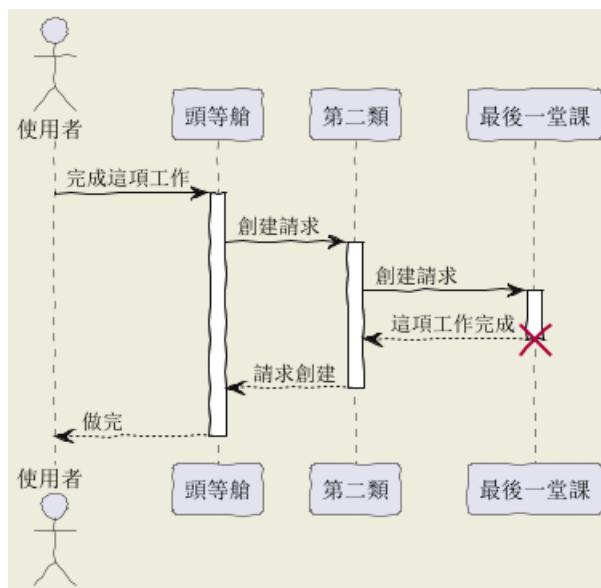
使用者 -> A: 完成這項工作
activate A

A -> B: 創建請求
activate B

B -> 別的東西: 創建請求
activate 別的東西
別的東西 --> B: 這項工作完成
destroy 別的東西

B --> A: 請求創建
deactivate B

A --> 使用者: 做完
deactivate A
@enduml
```



```
@startuml
(*) --> "膩平台"
--> === S1 ===
--> 鞠躬向公眾
--> === S2 ===
```



--> 這傢伙波武器
--> (*)

```
skinparam backgroundColor #AFFFFF
skinparam activityStartColor red
skinparam activityBarColor SaddleBrown
skinparam activityEndColor Silver
skinparam activityBackgroundColor Peru
skinparam activityBorderColor Peru
@enduml
```



@startuml

```
skinparam usecaseBackgroundColor DarkSeaGreen
skinparam usecaseArrowColor Olive
skinparam actorBorderColor black
skinparam usecaseBorderColor DarkSlateGray
```

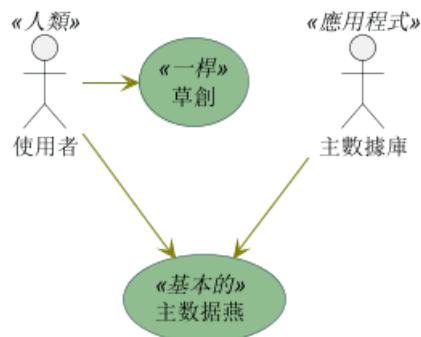
使用者 << 人類 >>
 "主數據庫" as 數據庫 << 應用程式 >>
 (草創) << 一桿 >>
 "主数据燕" as (贏余) << 基本的 >>

使用者 -> (草創)
 使用者 --> (贏余)

數據庫 --> (贏余)

@enduml





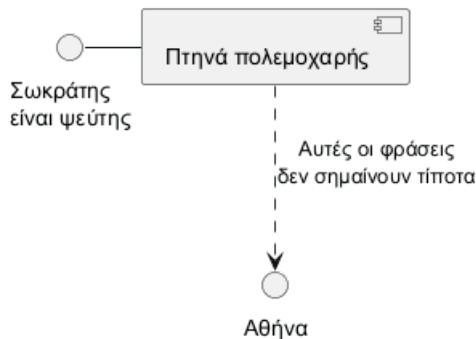
@startuml

() "Σωκράτηςψεύτης" as Σωκράτης

Σωκράτης - [Πιτηνά πολεμοχαρής]

[Πιτηνά πολεμοχαρής] ..> () Αθήνα : Αυτές οι φράσεις σημαίνουν τίποτα

@enduml



26.2 Charset

The default charset used when *reading* the text files containing the UML text description is system dependent.

Normally, it should just be fine, but in some case, you may want to use another charset. For example, with the command line:

```
java -jar plantuml.jar -charset UTF-8 files.txt
```

Or, with the ant task:

```
<!-- Put images in c:/images directory -->
<target name="main">
<plantuml dir=".src" charset="UTF-8" />
```

Depending of your Java installation, the following charset should be available: ISO-8859-1, UTF-8, UTF-16BE, UTF-16LE, UTF-16.

26.3 Using Unicode Character on PlantUML

On PlantUML diagram, you can integrate:

- Special characters using &#XXXX; or <U+XXXX> form;
- Emoji using <:XXXX:> or <:NameOfEmoji:>form.



27 PlantUML Standard Library

Welcome to the guide on PlantUML's official **Standard Library** (`stdlib`). Here, we delve into this integral resource that is now included in all official releases of PlantUML, facilitating a richer diagram creation experience. The library borrows its file inclusion convention from the "C standard library", a well-established protocol in the programming world.

27.0.1 Standard Library Overview

The Standard Library is a repository of files and resources, constantly updated to enhance your PlantUML experience. It forms the backbone of PlantUML, offering a range of functionalities and features to explore.

27.0.2 Contribution from the Community

A significant portion of the library's contents are generously provided by third-party contributors. We extend our heartfelt gratitude to them for their invaluable contributions that have played a pivotal role in enriching the library.

We encourage users to delve into the abundant resources the Standard Library offers, to not only enhance their diagram crafting experience but also possibly contribute and be a part of this collaborative endeavor.

27.1 List of Standard Library

You can list standard library folders using the special diagram:

```
@startuml  
stdlib  
@enduml
```



archimate

Version 1.1.0

Delivered by <https://github.com/plantuml-stdlib/Archimate-PlantUML>**aws**

Version 18.02.22

Delivered by <https://github.com/milo-minderbinder/AWS-PlantUML>**awslib**

Version 14.0.0

Delivered by <https://github.com/awslabs/aws-icons-for-plantuml>**awslib10**

Version 10.0.0

Delivered by <https://github.com/awslabs/aws-icons-for-plantuml>**awslib14**

Version 14.0.0

Delivered by <https://github.com/awslabs/aws-icons-for-plantuml>**azure**

Version 2.2.0

Delivered by <https://github.com/plantuml-stdlib/Azure-PlantUML>**c4**

Version 2.11.0

Delivered by <https://github.com/plantuml-stdlib/C4-PlantUML>**classy**

Version 1.0.2

Delivered by <https://github.com/james-gadrow-kr/classy-plantuml>**classy-c4**

Version 1.0.3

Delivered by <https://github.com/james-gadrow-kr/classy-c4>**cloudinsight**

Version 1.0.0

Delivered by <https://github.com/plantuml-stdlib/cicon-plantuml-sprites>**cloudogu**

Version 1.0.2

Delivered by <https://github.com/cloudogu/plantuml-cloudogu-sprites>**domainstory**

Version 0.4.0

Delivered by <https://github.com/johthor/DomainStory-PlantUML>**edgy**

Version 0.8.0

Delivered by <https://github.com/boessu/plantuml-stdlib>**eip**

Version 1.0.0

Delivered by <https://github.com/plantuml-stdlib/EIP-PlantUML>**elastic**

Version 0.0.1

Delivered by <https://github.com/Crashedmind/PlantUML-Elastic-icons>**gcp**

Version 6.0.0

Delivered by <https://github.com/Crashedmind/PlantUML-icons-GCP>**k8s**

Version 1.0.0

Delivered by <https://github.com/dcasati/kubernetes-PlantUML>**kubernetes**

Version 5.3.45

Delivered by <https://github.com/plantuml-stdlib/plantuml-kubernetes-sprites>**logos**

Version 1.1.0

Delivered by <https://github.com/plantuml-stdlib/gilbarbara-plantuml-sprites>**material**

Version 0.0.1

Delivered by <https://github.com/Templarian/MaterialDesign>

Version 1.0.0

Delivered by <https://github.com/Roemer/plantuml-office>**osa**

It is also possible to use the command line `java -jar plantuml.jar -stdlib` to display the same list.

Finally, you can extract the full standard library sources using `java -jar plantuml.jar -extractstdlib`. All files will be extracted in the folder `stdlib`.

Sources used to build official PlantUML releases are hosted here <https://github.com/plantuml/plantuml-stdlib>. You can create Pull Request to update or add some library if you find it relevant.

27.2 ArchiMate [archimate]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/archimate
src	https://github.com/ebbypeter/ArchiMate-PlantUML
orig	https://en.wikipedia.org/wiki/ArchiMate

This repository contains ArchiMate PlantUML macros and other includes for creating Archimate Diagrams easily and consistently.

```
@startuml
!include <archimate/Archimate>

title Archimate Sample - Internet Browser

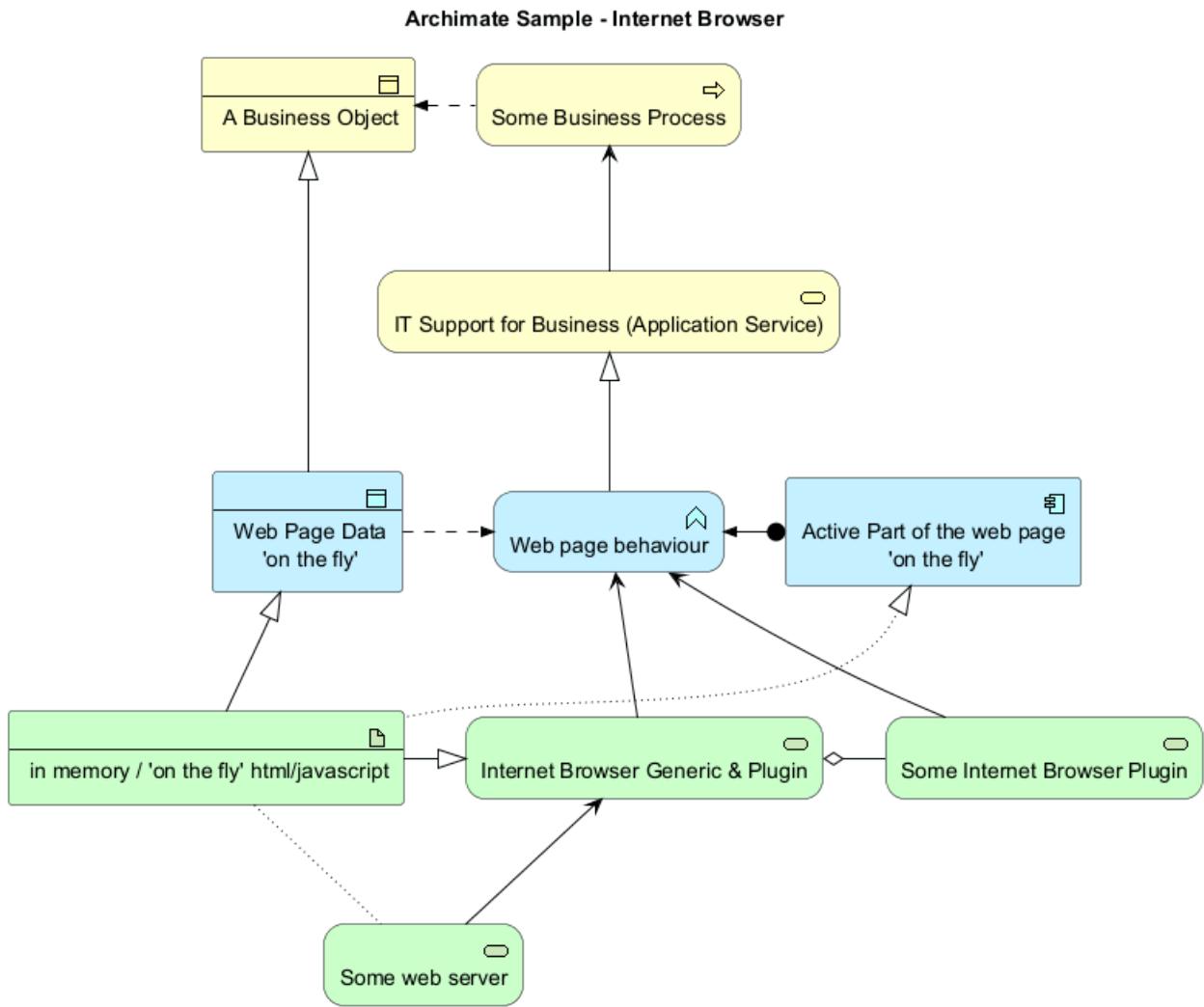
' Elements
Business_Object(businessObject, "A Business Object")
Business_Process(someBusinessProcess,"Some Business Process")
Business_Service(itSupportService, "IT Support for Business (Application Service)")

Application_DataObject(dataObject, "Web Page Data \n 'on the fly'")
Application_Function(webpageBehaviour, "Web page behaviour")
Application_Component(ActivePartWebPage, "Active Part of the web page \n 'on the fly')

Technology_Artifact(inMemoryItem,"in memory / 'on the fly' html/javascript")
Technology_Service(internetBrowser, "Internet Browser Generic & Plugin")
Technology_Service(internetBrowserPlugin, "Some Internet Browser Plugin")
Technology_Service(webServer, "Some web server")

'Relationships
Rel_Flow_Left(someBusinessProcess, businessObject, "")
Rel_Serving_Up(itSupportService, someBusinessProcess, "")
Rel_Specialization_Up(webpageBehaviour, itSupportService, "")
Rel_Flow_Right(dataObject, webpageBehaviour, "")
Rel_Specialization_Up(dataObject, businessObject, "")
Rel_Assignment_Left(ActivePartWebPage, webpageBehaviour, "")
Rel_Specialization_Up(inMemoryItem, dataObject, "")
Rel_Realization_Up(inMemoryItem, ActivePartWebPage, "")
Rel_Specialization_Right(inMemoryItem,internetBrowser, "")
Rel_Serving_Up(internetBrowser, webpageBehaviour, "")
Rel_Serving_Up(internetBrowserPlugin, webpageBehaviour, "")
Rel_Aggregation_Right(internetBrowser, internetBrowserPlugin, "")
Rel_Access_Up(webServer, inMemoryItem, "")
Rel_Serving_Up(webServer, internetBrowser, "")
@enduml
```





27.2.1 List possible sprites

You can list all possible sprites for Archimate using the following diagram:

```
@startuml
listsprite
@enduml
```





27.3 Amazon Labs AWS Library [awslib]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/awslib
src	https://github.com/awslabs/aws-icons-for-plantuml
orig	https://aws.amazon.com/en/architecture/icons/

The Amazon Labs AWS library provides PlantUML sprites, macros, and other includes for Amazon Web Services (AWS) services and resources.

Used to create PlantUML diagrams with AWS components. All elements are generated from the official AWS Architecture Icons and when combined with PlantUML and the C4 model, are a great way to communicate your design, deployment, and topology as code.

```

@startuml
!include <awslib/AWSCommon>
!include <awslib/InternetOfThings/IoTRule>
!include <awslib/Analytics/KinesisDataStreams>
!include <awslib/ApplicationIntegration/SimpleQueueService>

left to right direction

agent "Published Event" as event #fff

IoTRule(iotRule, "Action Error Rule", "error if Kinesis fails")
KinesisDataStreams(eventStream, "IoT Events", "2 shards")
SimpleQueueService(errorQueue, "Rule Error Queue", "failed Rule actions")

event --> iotRule : JSON message
iotRule --> eventStream : messages
iotRule --> errorQueue : Failed action message
@enduml

```



27.4 Azure library [azure]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/azure
src	https://github.com/RicardoNiepel/Azure-PlantUML/
orig	Microsoft Azure

The Azure library consists of Microsoft Azure icons.

Use it by including the file that contains the sprite, eg: !include <azure/Analytics/AzureEventHub>. When imported, you can use the sprite as normally you would, using <\$sprite_name>.

You may also include the AzureCommon.puml file, eg: !include <azure/AzureCommon>, which contains helper macros defined. With the AzureCommon.puml imported, you can use the NAME_OF_SPRITE(parameters...) macro.

Example of usage:

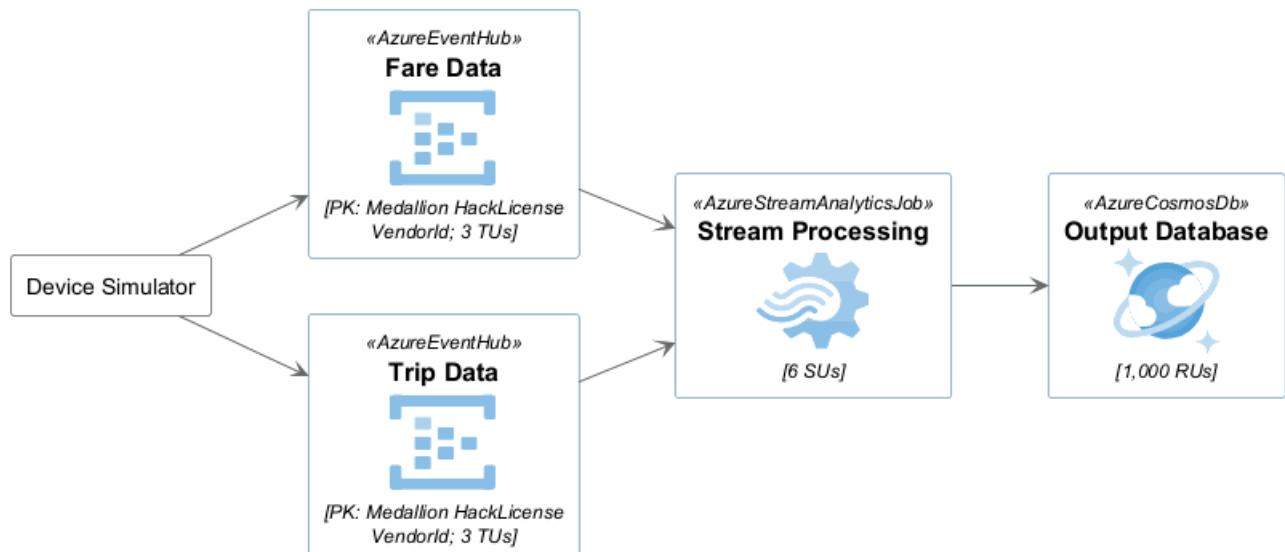
```
@startuml
!include <azure/AzureCommon>
!include <azure/Analytics/AzureEventHub>
!include <azure/Analytics/AzureStreamAnalyticsJob>
!include <azure/Databases/AzureCosmosDb>
```

left to right direction

```
agent "Device Simulator" as devices #fff
```

```
AzureEventHub(fareDataEventHub, "Fare Data", "PK: Medallion HackLicense VendorId; 3 TUs")
AzureEventHub(tripDataEventHub, "Trip Data", "PK: Medallion HackLicense VendorId; 3 TUs")
AzureStreamAnalyticsJob(streamAnalytics, "Stream Processing", "6 SUs")
AzureCosmosDb(outputCosmosDb, "Output Database", "1,000 RUs")
```

```
devices --> fareDataEventHub
devices --> tripDataEventHub
fareDataEventHub --> streamAnalytics
tripDataEventHub --> streamAnalytics
streamAnalytics --> outputCosmosDb
@enduml
```



27.5 C4 Library [C4]

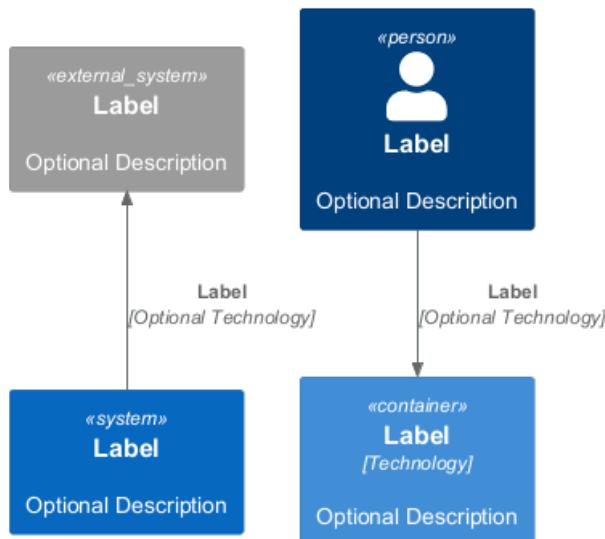
Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/C4
src	https://github.com/plantuml-stdlib/C4-PlantUML
orig	https://en.wikipedia.org/wiki/C4_model

```
@startuml
!include <C4/C4_Container>

Person(personAlias, "Label", "Optional Description")
Container(containerAlias, "Label", "Technology", "Optional Description")
System(systemAlias, "Label", "Optional Description")

System_Ext(extSystemAlias, "Label", "Optional Description")

Rel(personAlias, containerAlias, "Label", "Optional Technology")
Rel_U(systemAlias, extSystemAlias, "Label", "Optional Technology")
@enduml
```



27.6 Cloud Insight [cloudinsight]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/cloudinsight
src	https://github.com/rabelenda/cicon-plantuml-sprites
orig	Cloudinsight icons

This repository contains PlantUML sprites generated from Cloudinsight icons, which can easily be used in PlantUML diagrams for nice visual representation of popular technologies.

```
@startuml
!include <cloudinsight/tomcat>
!include <cloudinsight/kafka>
!include <cloudinsight/java>
!include <cloudinsight/cassandra>

title Cloudinsight sprites example

skinparam monochrome true

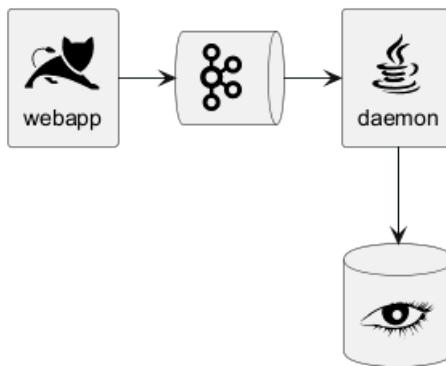
rectangle "<$tomcat>\nwebapp" as webapp
```



```
queue "<$kafka>" as kafka
rectangle "<$java>\ndaemon" as daemon
database "<$cassandra>" as cassandra
```

```
webapp -> kafka
kafka -> daemon
daemon --> cassandra
@enduml
```

Cloudinsight sprites example



27.7 Cloudogu [cloudogu]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/cloudogu
src	https://github.com/cloudogu/plantuml-cloudogu-sprites
orig	https://cloudogu.com

The Cloudogu library provides PlantUML sprites, macros, and other includes for Cloudogu services and resources.

```
@startuml
!include <cloudogu/common>
!include <cloudogu/dogus/jenkins>
!include <cloudogu/dogus/cloudogu>
!include <cloudogu/dogus/scm>
!include <cloudogu/dogus/smeagol>
!include <cloudogu/dogus/nexus>
!include <cloudogu/tools/k8s>

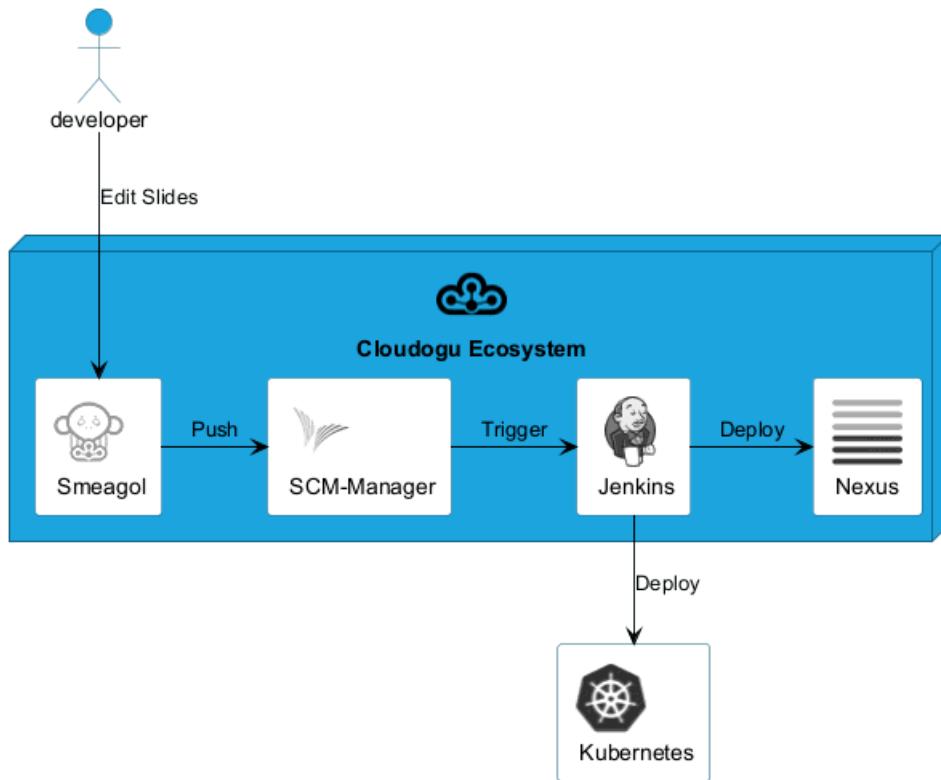
node "Cloudogu Ecosystem" <<$cloudogu>> {
    DOGU_JENKINS(jenkins, Jenkins) #ffffff
    DOGU_SCM(scm, SCM-Manager) #ffffff
    DOGU_SMEAGOL(smeagol, Smeagol) #ffffff
    DOGU_NEXUS(nexus, Nexus) #ffffff
}

TOOL_K8S(k8s, Kubernetes) #ffffff

actor developer

developer --> smeagol : "Edit Slides"
smeagol -> scm : Push
scm -> jenkins : Trigger
jenkins -> nexus : Deploy
jenkins --> k8s : Deploy
@enduml
```





All cloudogu sprites

See all possible cloudogu sprites on plantuml-cloudogu-sprites.

27.8 EDGY: An Open Source tool for collaborative Enterprise Design [edgy]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/edgy
src	https://github.com/boessu/plantuml-stdlib/tree/master/edgy
orig	https://enterprise.design/

“To become whole, enterprises must embrace a holistic, collaborative way of design: transcending silos, combining perspectives, looking for connections instead of divisions. An enterprise designed together works better together.”

-Bard Papegaij, Wolfgang Goebel and Milan Guenther, curators of EDGY 23

EDGY helps to visualize, communicate, and co-design enterprises across different disciplines. EDGY is a design language that provides guidelines for enterprises to create effective and efficient digital products, services, and experiences. It was developed by the EDGY team with input from industry experts, researchers, and practitioners in order to address common challenges faced when developing complex systems. The foundation of Edgy is based on four key principles: simplicity, modularity, scalability, and adaptability. These principles are designed to help enterprises create products that can be easily maintained over time while also being able to scale up or down as needed. Additionally, the language provides a set of guidelines for designing user interfaces, data models, business processes, and more, making it an essential toolkit for any organization looking to improve their offerings.

27.8.1 Basic Elements and Interconnections

EDGY is an open-source language for enterprise design that uses only four base elements: people, activity, object, and outcome. These elements can be specialized into facet and intersection elements, which describe the enterprise from different perspectives: identity, architecture, and experience.

27.8.2 Elements

The basic syntax of an element or a facet is:



```
$element/facet("label", [identifier], [lightColor])
```

Parameter	Description
label	Mandatory: label of the element.
identifier	Dependant: Identifies the element (for creating relations). Optional if you don't link them to other elements/facets.
lightColor	Optional: 0 sets the standard color. 1 sets a lighter color. As default, facets do have lighter colors than elements.

```
@startuml
```

```
!include <edgy/edgy>
```

```
$baseFacet("Basic elements") {
```

```
$people("People")
note bottom
    The individuals co-creating
    the enterprise or using
    products.
end note
```

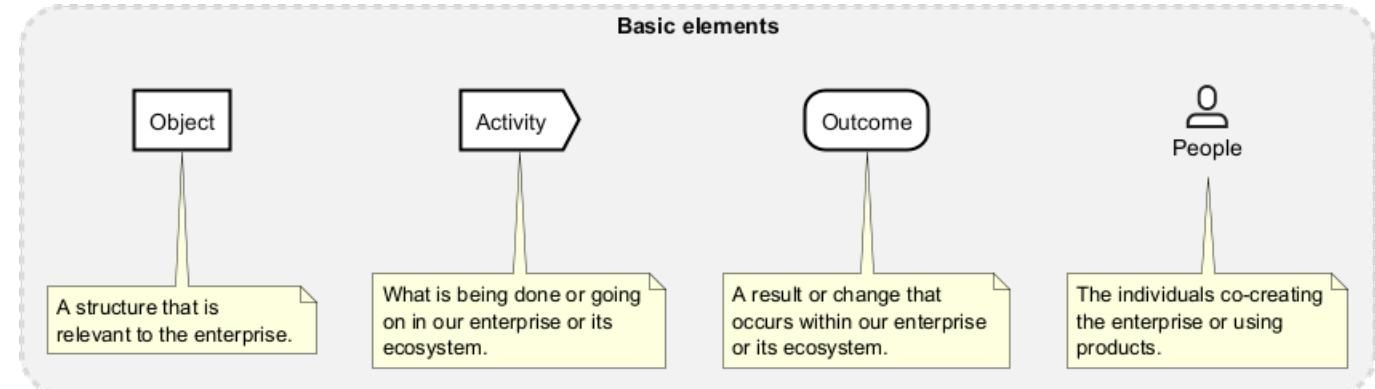
```
$outcome("Outcome")
note bottom
    A result or change that
    occurs within our enterprise
    or its ecosystem.
end note
```

```
$activity("Activity")
note bottom
    What is being done or going
    on in our enterprise or its
    ecosystem.
end note
```

```
$object("Object")
note bottom
    A structure that is
    relevant to the enterprise.
end note
```

```
}
```

```
@enduml
```



27.8.3 Relationships

The elements (or facets) can be connected with three types of relationships: link, flow and tree.

```
$link/flow/tree(fromIdentifier, toIdentifier, ["Description"])
```



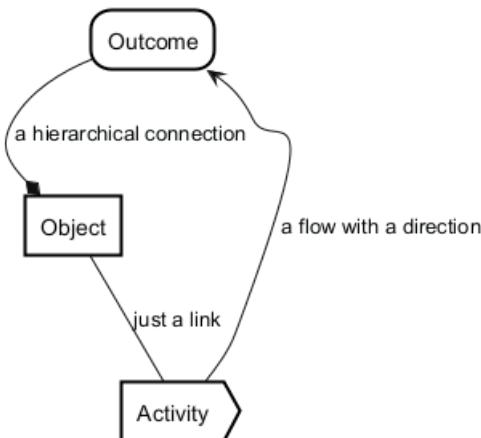
Parameter	Description
fromIdentifier	Mandatory: Identifies the starting element of a relation.
toIdentifier	Mandatory: Identifies the ending element of a relation.
label	Optional: label of the element.

All relations can have a direction hint as a suffix (Up/Down/Left/Right). See examples in the chapter "Facets". While it does often help to give PlantUML (basically GraphViz) a direction hint, it not always helps. if you don't get the exact result you expect: don't waste too much lifetime on it.

```
@startuml
!include <edgy/edgy>

$outcome("Outcome", outcome)
$activity("Activity", activity)
$object("Object", object)

$link(object, activity, "just a link")
$flow(activity, outcome, "a flow with a direction")
$tree(outcome, object, "a hierarchical connection")
@enduml
```



There are quite some hierarchical linking in edgy. Or maps. So it is also possible to group/nesting elements:

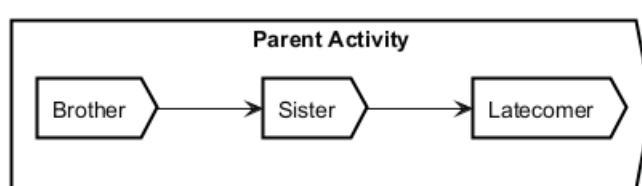
```
@startuml
!include <edgy/edgy>

left to right direction

$activity("Parent Activity") {
    $activity("Brother", child1, 1)
    $activity("Sister", child2, 1)
    $activity("Latecomer", child3, 1)
}

$flow(child1, child2)
$flow(child2, child3)

@enduml
```



27.8.4 Facets

A facet is a perspective that relates to any enterprise, featuring a set of questions that an enterprise needs to answer in order to achieve a coherent design. There are three facets in EDGY: Identity, Architecture, and Experience. Each facet references five enterprise elements: three facet elements, and two intersection elements at the overlap with the neighbouring facets.

27.8.5 Identity

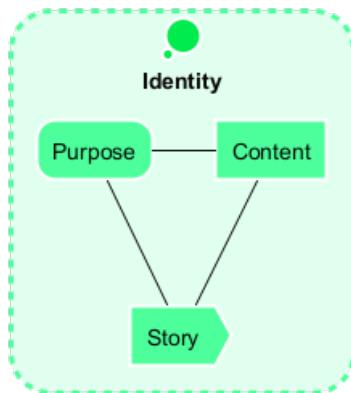
The Identity Facet describes why the enterprise exists and what it stands for.

```
@startuml
!include <edgy/edgy>

$identityFacet(Identity, identity) {
    $content(Content, content)
    $purpose(Purpose, purpose)
    $story(Story, story)
}

$linkLeft(content, purpose)
$linkDown(content, story)
$linkDown(purpose, story)

@enduml
```



27.8.6 Architecture

The Architecture facet is about the structures and processes that enable the enterprise to operate and deliver.

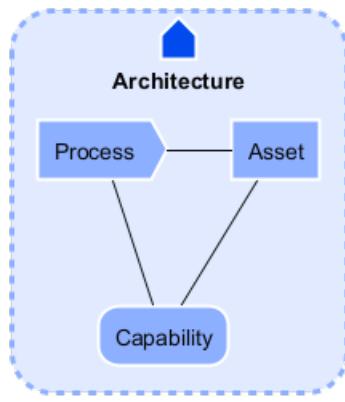
```
@startuml
!include <edgy/edgy>

$architectureFacet(Architecture) {
    $process(Process, process)
    $asset(Asset, asset)
    $capability(Capability, capability)
}

$linkRight(process, asset)
$linkDown(process, capability)
$linkDown(asset, capability)

@enduml
```





27.8.7 Experience

The Experience Facet is about the impact that the enterprise has on people and their lives through its interactions.

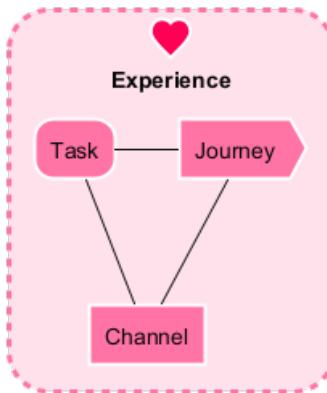
```

@startuml
!include <edgy/edgy>

$experienceFacet(Experience) {
    $task(Task, task)
    $journey(Journey, journey)
    $channel(Channel, channel)
}

$linkRight(task, journey)
$linkDown(task, channel)
$linkDown(journey, channel)

@enduml
  
```



27.8.8 Intersections

Intersections are lenses that connect facets and disciplines, such as organisation, product, and brand.

```

@startuml
!include <edgy/edgy>

$experienceFacet(Experience, experience)
$architectureFacet(Architecture, architecture)
$identityFacet(Identity, identity)

$organisationFacet(Organisation, org) {
    $organisation(Organisation, organisation)
}
  
```



```

$brandFacet(Brand) {
$brand(Brand, brand)
}

$productFacet(Product){
$product(Product, product)
}

$flow(brand, identity, "represents/evokes")
$flow(brand, experience, "Supports/appears in")

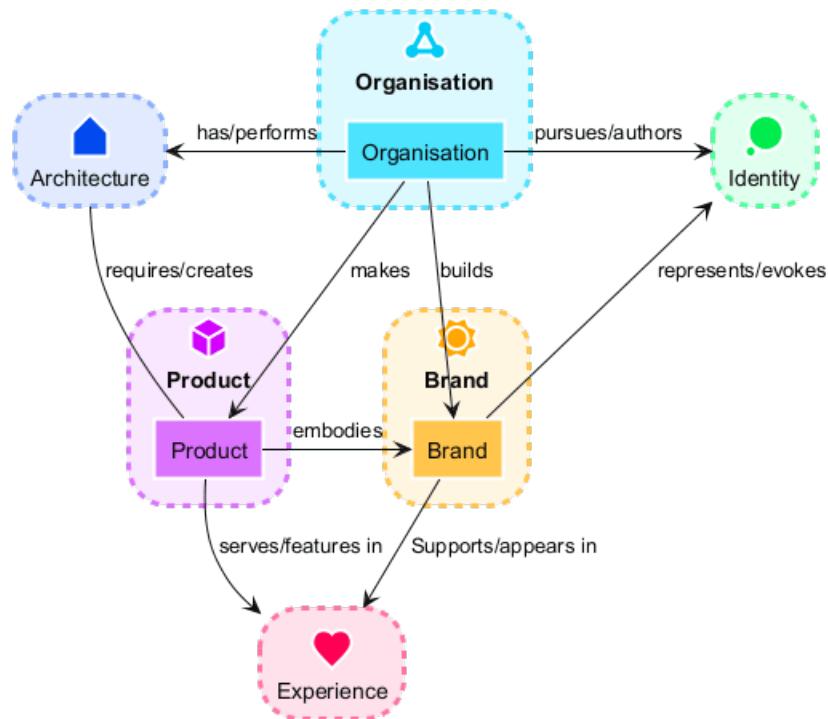
$flowLeft(organisation, identity, "pursues/authors")
$flowRight(organisation, architecture, "has/performers")

$flow(product, experience, "serves/features in")
$linkUp(product, architecture, "requires/creates")

$flow(organisation, brand, "builds")
$flow(organisation, product, "makes")
$flowLeft(product, brand, "embodies")

@enduml

```



27.8.9 Alternative visual styling

Finally, there is also an alternative representation that focuses on rectangles with stereotypes. The approach described above is 100% compatible. It can therefore be activated with a simple swap from `!include <edgy/edgy>` to `!include <edgy/edgy2>`. This can sometimes be useful if the people involved do not immediately know the color codes and concrete meanings of the EDGY elements by heart. Also color-blind people can benefit from this ;-)

```

@startuml
!include <edgy/edgy2>

$baseFacet("Basic elements") {

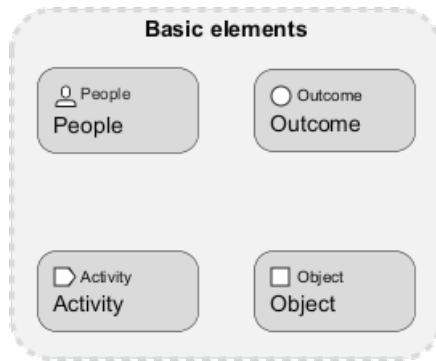
```



```

$people("People")
$outcome("Outcome")
$activity("Activity")
$object("Object")
}
@enduml

```



27.9 Elastic library [elastic]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/elastic
src	https://github.com/Crashedmind/PlantUML-Elastic-icons
orig	Elastic

The Elastic library consists of Elastic icons. It is similar in use to the AWS and Azure libraries (it used the same tool to create them).

Use it by including the file that contains the sprite, eg: !include elastic/elasticsearch/elasticsearch>. When imported, you can use the sprite as normally you would, using <\$sprite_name>.

You may also include the common.puml file, eg: !include <elastic/common>, which contains helper macros defined. With the common.puml imported, you can use the NAME//OF//SPRITE(parameters...) macro.

Example of usage:

```

@startuml
!include <elastic/common>
!include <elastic/elasticsearch/elasticsearch>
!include <elastic/logstash/logstash>
!include <elastic/kibana/kibana>

```

```

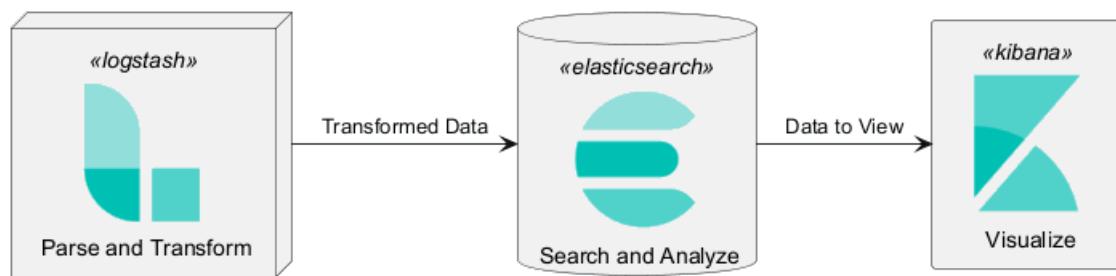
ELASTICSEARCH(ElasticSearch, "Search and Analyze", database)
LOGSTASH(Logstash, "Parse and Transform", node)
KIBANA(Kibana, "Visualize", agent)

```

```

Logstash -right-> Elasticsearch: Transformed Data
ElasticSearch -right-> Kibana: Data to View
@enduml

```



All Elastic Sprite Set

```
@startuml
'Adapted from https://github.com/Crashedmind/PlantUML-Elastic-icons/blob/master/All.puml

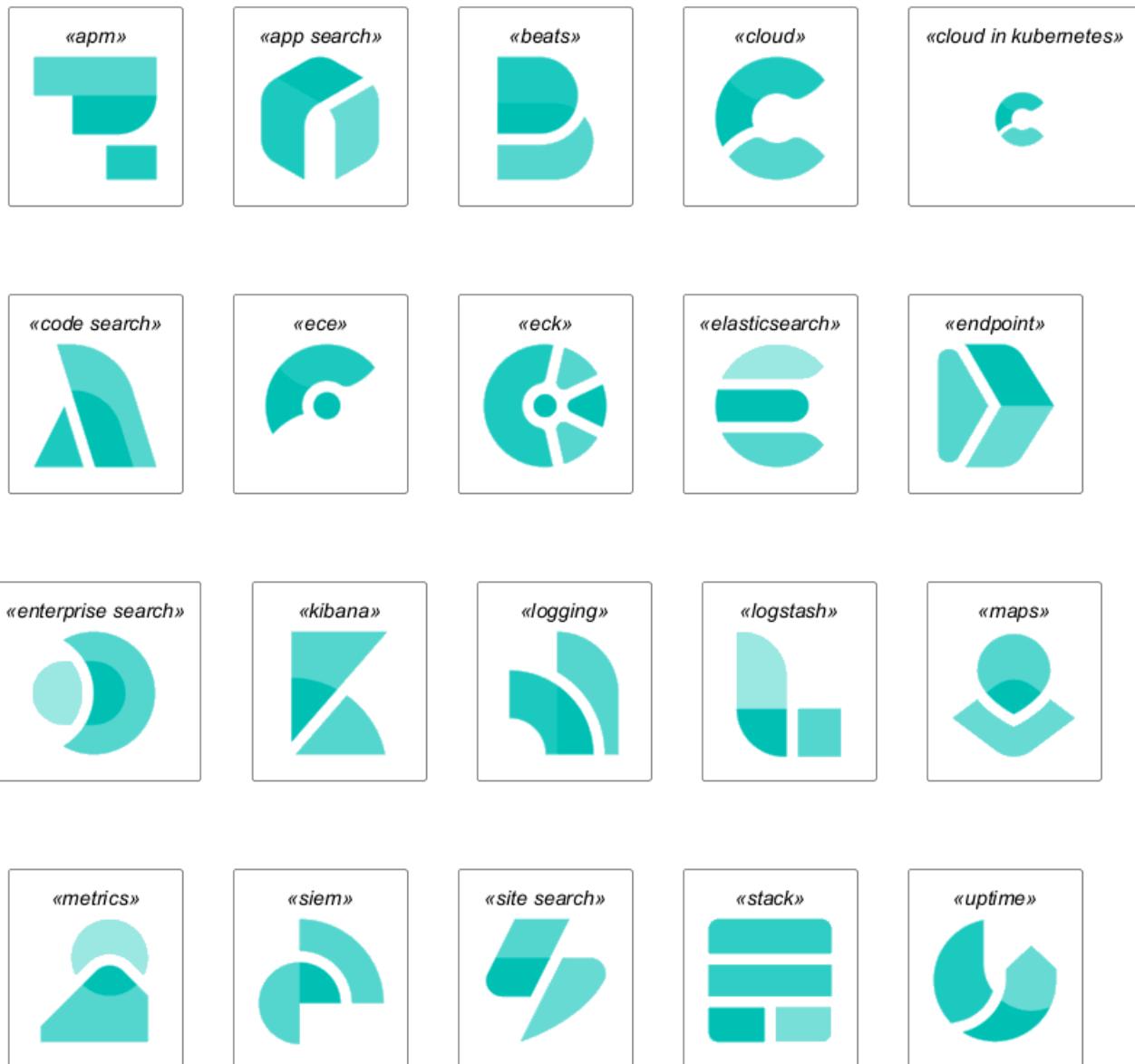
'Elastic stuff here
'=====

!include <elastic/common>
!include <elastic/apm/apm>
!include <elastic/app_search/app_search>
!include <elastic/beats/beats>
!include <elastic/cloud/cloud>
!include <elastic/cloud_in_kubernetes/cloud_in_kubernetes>
!include <elastic/code_search/code_search>
!include <elastic/ece/ece>
!include <elastic/eck/eck>
' Beware of the difference between Crashedmind and plantuml-stdlib version: with '_' usage!
!include <elastic/elasticsearch/elasticsearch>
!include <elastic/endpoint/endpoint>
!include <elastic/enterprise_search/enterprise_search>
!include <elastic/kibana/kibana>
!include <elastic/logging/logging>
!include <elastic/logstash/logstash>
!include <elastic/maps/maps>
!include <elastic/metrics/metrics>
!include <elastic/siem/siem>
!include <elastic/site_search/site_search>
!include <elastic/stack/stack>
!include <elastic/uptime/uptime>

skinparam agentBackgroundColor White

APM(apm)
APP_SEARCH(app_search)
BEATS(beats)
CLOUD(cloud)
CLOUD_IN_KUBERNETES(cloud_in_kubernetes)
CODE_SEARCH(code_search)
ECE(ece)
ECK(eck)
ELASTICSEARCH(elastic_search)
ENDPOINT(endpoint)
ENTERPRISE_SEARCH(enterprise_search)
KIBANA(kibana)
LOGGING(logging)
LOGSTASH(logstash)
MAPS.maps)
METRICS(metrics)
SIEM(siem)
SITE_SEARCH(site_search)
STACK(stack)
UPTIME(uptime)
@enduml
```





27.10 Google Material Icons [material]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/material
src	https://github.com/Templarian/MaterialDesign
orig	Material Design Icons

This library consists of a free Material style icons from Google and other artists.

Use it by including the file that contains the sprite, eg: `!include <material/ma_folder_move>`. When imported, you can use the sprite as normally you would, using `<$ma_sprite_name>`. Notice that this library requires an `ma_` prefix on sprites names, this is to avoid clash of names if multiple sprites have the same name on different libraries.

You may also include the `common.puml` file, eg: `!include <material/common>`, which contains helper macros defined. With the `common.puml` imported, you can use the `MA_NAME_OF_SPRITE(parameters...)` macro, note again the use of the prefix `MA_`.

Example of usage:

```
@startuml
!include <material/common>
```



```
' To import the sprite file you DON'T need to place a prefix!
!include <material/folder_move>

MA_FOLDER_MOVE(Red, 1, dir, rectangle, "A label")
@enduml
```



Notes:

When mixing sprites macros with other elements you may get a syntax error if, for example, trying to add a rectangle along with classes. In those cases, add { and } after the macro to create the empty rectangle.

Example of usage:

```
@startuml
!include <material/common>
' To import the sprite file you DON'T need to place a prefix!
!include <material/folder_move>

MA_FOLDER_MOVE(Red, 1, dir, rectangle, "A label") {
}

class foo {
    bar
}
@enduml
```



27.11 Kubernetes [kubernetes]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/kubernetes
src	https://github.com/michiel/plantuml-kubernetes-sprites
orig	Kubernetes

```
@startuml
!include <kubernetes/k8s-sprites-unlabeled-25pct>
package "Infrastructure" {
    component "<$master>\nmaster" as master
    component "<$etcd>\netcd" as etcd
    component "<$node>\nnode" as node
}
@enduml
```





27.12 Logos [logos]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/logos
src	https://github.com/plantuml-stdlib/gilbarbara-plantuml-sprites
orig	Gil Barbara's logos

This repository contains PlantUML sprites generated from Gil Barbara's logos, which can easily be used in PlantUML diagrams for nice visual aid.

```
@startuml
!include <logos/flask>
!include <logos/kafka>
!include <logos/kotlin>
!include <logos/cassandra>

title Gil Barbara's logos example

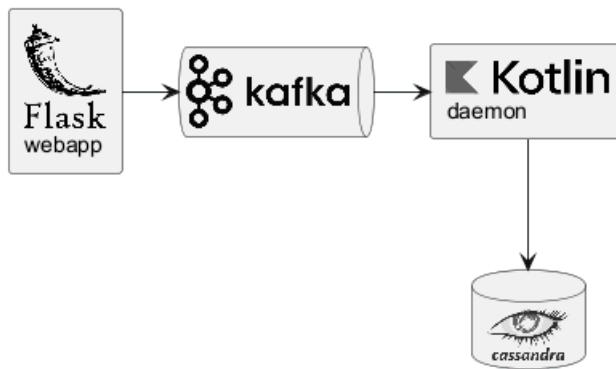
skinparam monochrome true

rectangle "<$flask>\nwebapp" as webapp
queue "<$kafka>" as kafka
rectangle "<$kotlin>\ndaemon" as daemon
database "<$cassandra>" as cassandra

webapp -> kafka
kafka -> daemon
daemon --> cassandra
@enduml
```



Gil Barbara's logos example



```

@startuml
scale 0.7
!include <logos/apple-pay>
!include <logos/dinersclub>
!include <logos/discover>
!include <logos/google-pay>
!include <logos/jcb>
!include <logos/maestro>
!include <logos/mastercard>
!include <logos/paypal>
!include <logos/unionpay>
!include <logos/visaelectron>
!include <logos/visa>
' ...

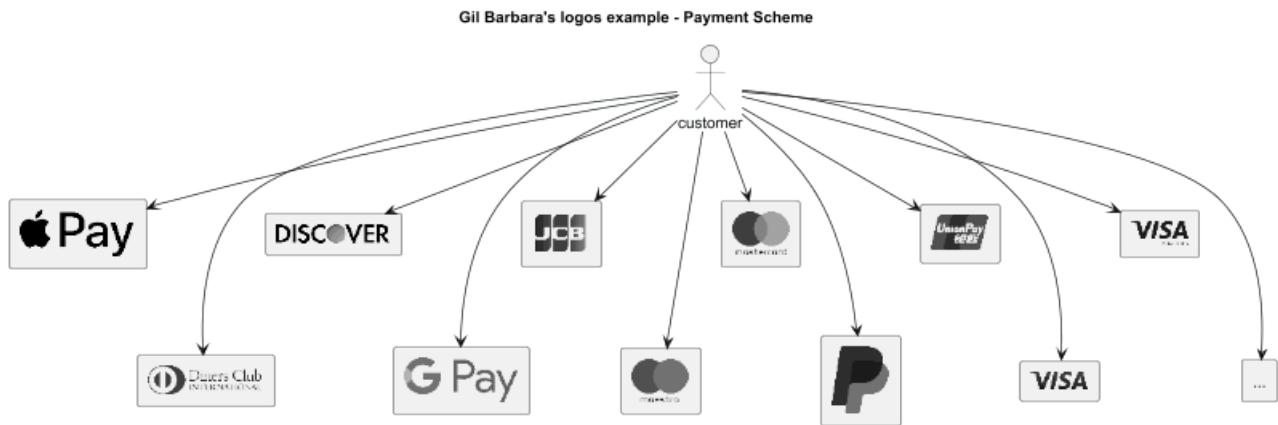
title Gil Barbara's logos example - **Payment Scheme**

actor customer
rectangle "<$apple-pay>" as ap
rectangle "<$dinersclub>" as dc
rectangle "<$discover>" as d
rectangle "<$google-pay>" as gp
rectangle "<$jcb>" as j
rectangle "<$maestro>" as ma
rectangle "<$mastercard>" as m
rectangle "<$paypal>" as p
rectangle "<$unionpay>" as up
rectangle "<$visa>" as v
rectangle "<$visaelectron>" as ve
rectangle "..." as etc

customer --> ap
customer ---> dc
customer --> d
customer ---> gp
customer --> j
customer ---> ma
customer --> m
customer ---> p
customer --> up
customer ---> v
customer --> ve
customer ---> etc
  
```



```
@enduml
```



27.13 Office [office]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/office
src	https://github.com/Roemer/plantuml-office
orig	

There are sprites (*.puml) and colored png icons available. Be aware that the sprites are all only monochrome even if they have a color in their name (due to automatically generating the files). You can either color the sprites with the macro (see examples below) or directly use the fully colored pngs. See the following examples on how to use the sprites, the pngs and the macros.

Example of usage:

```
@startuml
!include <tupadr3/common>

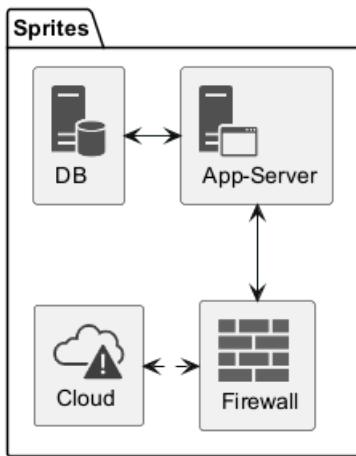
!include <office/Servers/database_server>
!include <office/Servers/application_server>
!include <office/Concepts/firewall_orange>
!include <office/Clouds/cloud_disaster_red>

title Office Icons Example

package "Sprites" {
    OFF_DATABASE_SERVER(db,DB)
    OFF_APPLICATION_SERVER(app,App-Server)
    OFF_FIREWALL_ORANGE(fw,Firewall)
    OFF_CLOUD_DISASTER_RED(cloud,Cloud)
    db <-> app
    app <--> fw
    fw <.left.> cloud
}
@enduml
```



Office Icons Example



```

@startuml
!include <tupadr3/common>

!include <office/servers/database_server>
!include <office/servers/application_server>
!include <office/Concepts/firewall_orange>
!include <office/Clouds/cloud_disaster_red>

' Used to center the label under the images
skinparam defaultTextAlignment center

title Extended Office Icons Example

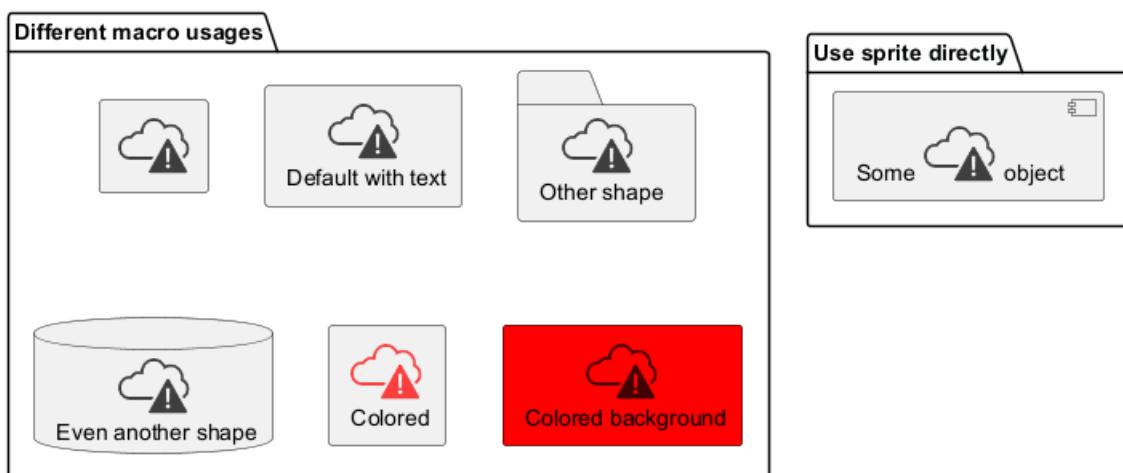
package "Use sprite directly" {
    [Some <$cloud_disaster_red> object]
}

package "Different macro usages" {
    OFF_CLOUD_DISASTER_RED(cloud1)
    OFF_CLOUD_DISASTER_RED(cloud2,Default with text)
    OFF_CLOUD_DISASTER_RED(cloud3,Other shape,Folder)
    OFF_CLOUD_DISASTER_RED(cloud4,Even another shape,Database)
    OFF_CLOUD_DISASTER_RED(cloud5,Colored,Rectangle, red)
    OFF_CLOUD_DISASTER_RED(cloud6,Colored background) #red
}
@enduml

```



Extended Office Icons Example



27.14 Open Security Architecture (OSA) [osa]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/osa
src	https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons
orig	https://www.opensecurityarchitecture.org

@startuml

```
'Adapted from https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons/blob/master/all
scale .5
!include <osa/arrow/green/left/left>
!include <osa/arrow/yellow/right/right>
!include <osa/awareness/awareness>
!include <osa/contract/contract>
!include <osa/database/database>
!include <osa/desktop/desktop>
!include <osa/desktop/imac/imac>
!include <osa/device_music/device_music>
!include <osa/device_scanner/device_scanner>
!include <osa/device_usb/device_usb>
!include <osa/device_wireless_router/device_wireless_router>
!include <osa/disposal/disposal>
!include <osa/drive_optical/drive_optical>
!include <osa/firewall/firewall>
!include <osa/hub/hub>
!include <osa/ics/drive/drive>
!include <osa/ics/plc/plc>
!include <osa/ics/thermometer/thermometer>
!include <osa/id/card/card>
!include <osa/laptop/laptop>
!include <osa/lifecycle/lifecycle>
!include <osa/lightning/lightning>
!include <osa/media_flash/media_flash>
!include <osa/media_optical/media_optical>
!include <osa/media_tape/media_tape>
!include <osa/mobile/pda/pda>
!include <osa/padlock/padlock>
!include <osa/printer/printer>
!include <osa/site_branch/site_branch>
!include <osa/site_factory/site_factory>
!include <osa/vpn/vpn>
```



```
!include <osa/wireless/network/network>

rectangle "OSA" {
rectangle "Left:\n <$left>" 
rectangle "Right:\n <$right>" 
rectangle "Awareness:\n <$awareness>" 
rectangle "Contract:\n <$contract>" 
rectangle "Database:\n <$database>" 
rectangle "Desktop:\n <$desktop>" 
rectangle "Imac:\n <$imac>" 
rectangle "Device_music:\n <$device_music>" 
rectangle "Device_scanner:\n <$device_scanner>" 
rectangle "Device_usb:\n <$device_usb>" 
rectangle "Device_wireless_router:\n <$device_wireless_router>" 
rectangle "Disposal:\n <$disposal>" 
rectangle "Drive_optical:\n <$drive_optical>" 
rectangle "Firewall:\n <$firewall>" 
rectangle "Hub:\n <$hub>" 
rectangle "Drive:\n <$drive>" 
rectangle "Plc:\n <$plc>" 
rectangle "Thermometer:\n <$thermometer>" 
rectangle "Card:\n <$card>" 
rectangle "Laptop:\n <$laptop>" 
rectangle "Lifecycle:\n <$lifecycle>" 
rectangle "Lightning:\n <$lightning>" 
rectangle "Media_flash:\n <$media_flash>" 
rectangle "Media_optical:\n <$media_optical>" 
rectangle "Media_tape:\n <$media_tape>" 
rectangle "Pda:\n <$pda>" 
rectangle "Padlock:\n <$padlock>" 
rectangle "Printer:\n <$printer>" 
rectangle "Site_branch:\n <$site_branch>" 
rectangle "Site_factory:\n <$site_factory>" 
rectangle "Vpn:\n <$vpn>" 
rectangle "Network:\n <$network>" 
}
@enduml
```



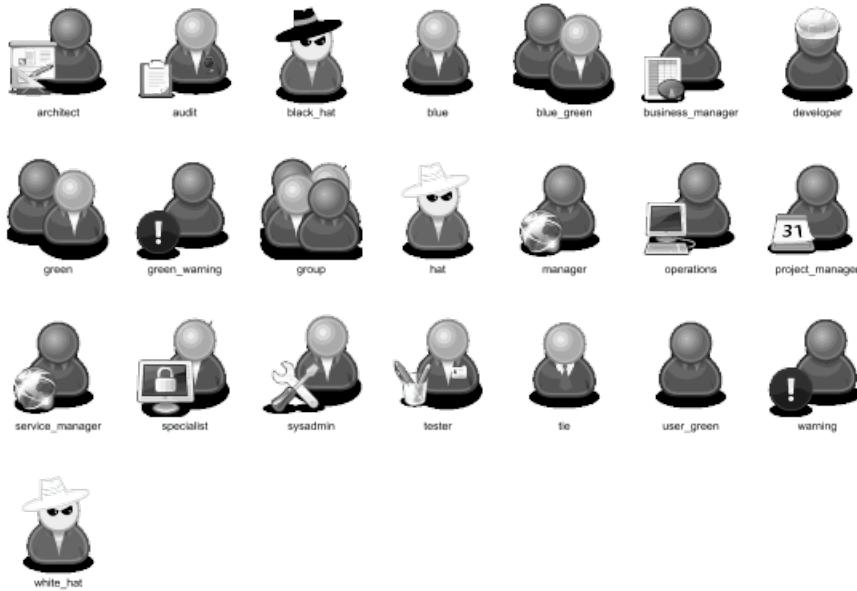


```
@startuml
scale .5
!include <osa/user/audit/audit>
'beware of 'hat-sprite'
!include <osa/user/black/hat/hat-sprite>
!include <osa/user/blue/blue>
!include <osa/user/blue/security/specialist/specialist>
!include <osa/user/blue/sysadmin/sysadmin>
!include <osa/user/blue/tester/tester>
!include <osa/user/blue/tie/tie>
!include <osa/user/green/architect/architect>
!include <osa/user/green/business/manager/manager>
!include <osa/user/green/developer/developer>
!include <osa/user/green/green>
!include <osa/user/green/operations/operations>
!include <osa/user/green/project/manager/manager>
!include <osa/user/green/service/manager/manager>
!include <osa/user/green/warning/warning>
!include <osa/user/large/group/group>
!include <osa/users/blue/green/green>
!include <osa/user/white/hat/hat>
```

listsprites



```
@enduml
```



27.15 Tupadr3 library [tupadr3]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/tupadr3
src	https://github.com/tupadr3/plantuml-icon-font-sprites
orig	https://github.com/tupadr3/plantuml-icon-font-sprites#icon-sets

This library contains several libraries of icons (including Devicons and Font Awesome).

Use it by including the file that contains the sprite, eg: !include <font-awesome/common>. When imported, you can use the sprite as normally you would, using <\$sprite_name>.

You may also include the common.puml file, eg: !include <font-awesome/common>, which contains helper macros defined. With the common.puml imported, you can use the NAME_OF_SPRITE(parameters...) macro.

Example of usage:

```
@startuml
!include <tupadr3/common>
!include <tupadr3/font-awesome/server>
!include <tupadr3/font-awesome/database>

title Styling example

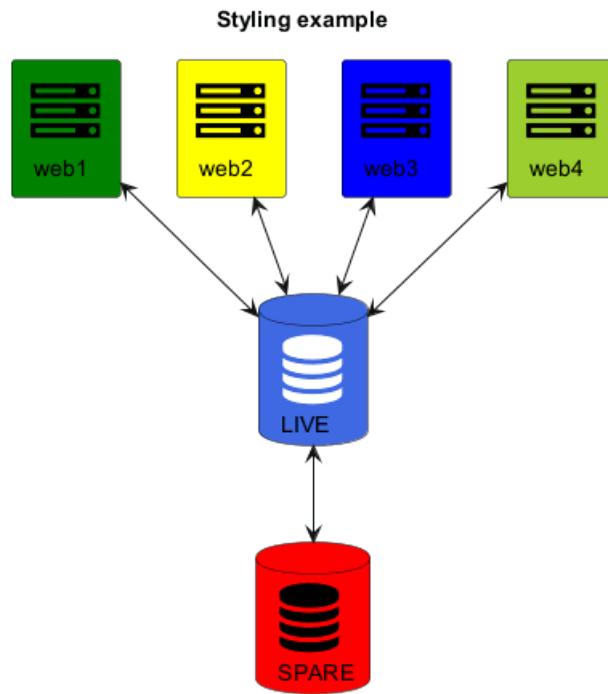
FA_SERVER(web1,web1) #Green
FA_SERVER(web2,web2) #Yellow
FA_SERVER(web3,web3) #Blue
FA_SERVER(web4,web4) #YellowGreen

FA_DATABASE(db1,LIVE,database,white) #RoyalBlue
FA_DATABASE(db2,SPARE,database) #Red

db1 <--> db2

web1 <--> db1
web2 <--> db1
web3 <--> db1
web4 <--> db1
@enduml
```

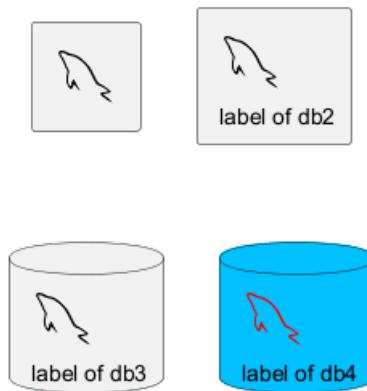




```

@startuml
!include <tupadr3/common>
!include <tupadr3/devicons/mysql>

DEV_MYSQL(db1)
DEV_MYSQL(db2,label of db2)
DEV_MYSQL(db3,label of db3,database)
DEV_MYSQL(db4,label of db4,database,red) #DeepSkyBlue
@enduml
  
```



27.16 AWS library [aws]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/aws
src	https://github.com/milo-minderbinder/AWS-PlantUML
orig	https://aws.amazon.com/en/architecture/icons/

Warning: We are thinking about deprecating this library.

So you should probably use <awslib> instead (see above).

□ hr □ □



The AWS library consists of Amazon AWS icons, it provides icons of two different sizes (normal and large).

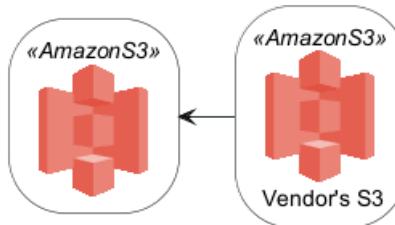
Use it by including the file that contains the sprite, eg: !include <aws/Storage/AmazonS3/AmazonS3>. When imported, you can use the sprite as normally you would, using <\$sprite_name>.

You may also include the common.puml file, eg: !include <aws/common>, which contains helper macros defined. With the common.puml imported, you can use the NAME_OF_SPRITE(parameters...) macro.

Example of usage:

```
@startuml  
!include <aws/common>  
!include <aws/Storage/AmazonS3/AmazonS3>
```

```
AMAZONS3(s3_internal)  
AMAZONS3(s3_partner, "Vendor's S3")  
s3_internal <- s3_partner  
@enduml
```



Contents

1 Диаграммы последовательностей	1
1.1 Основные примеры	1
1.2 Объявление участников	2
1.3 Объявление участника в многострочной форме	4
1.4 Использование небуквенных символов в названиях участников	4
1.5 Сообщения к самому себе	5
1.6 Выравнивание текста	5
1.6.1 Текст ответного сообщения под стрелкой	5
1.7 Изменение стиля стрелок	6
1.8 Изменить цвет стрелок	6
1.9 Нумерация сообщений в последовательностях	7
1.10 Название, Заголовок и футер страницы	10
1.11 Разбиение диаграмм	11
1.12 Группировка сообщений	11
1.13 Дополнительная метка группы	12
1.14 Примечания в сообщениях	13
1.15 Другие примечания	14
1.16 Изменение формы примечаний	15
1.17 Заметка отображаемая поверх всех участников	15
1.18 Выравнивание нескольких заметок по одному уровню [/]	16
1.19 Creole и HTML	17
1.20 Разделитель	18
1.21 Ссылки	18
1.22 Задержка на диаграммах	19
1.23 Перенос текста	19
1.24 Промежутки	20
1.25 Активация и деактивация линии существования	21
1.26 Return	22
1.27 Отображение создания участника процессом	23
1.28 Быстрый синтаксис для активации, деактивации и создания	23
1.29 Входящие и исходящие сообщения	25
1.30 Короткие стрелки для входящих и исходящих сообщений	26
1.31 Якоря и длительность	27
1.32 Стереотипы и отметки	27
1.33 Расположение стереотипов	29
1.33.1 Расположение сверху (<i>настройка по умолчанию</i>)	29
1.33.2 Расположение снизу	29
1.34 Больше информации в заголовках	29
1.35 Группировка участников	31
1.36 Удаление футера	32
1.37 Skipparam	32
1.38 Изменение отступов	34
1.39 Приложение: Примеры всех типов стрелок	35
1.39.1 Обычная стрелка	35
1.39.2 Возвращающаяся трелка	36
1.39.3 Входящие и исходящие сообщения (с '[' , ']')	37
1.39.4 Входящие сообщения (с '[')	37
1.39.5 Исходящие сообщения (с ']')	39
1.39.6 Короткие входящие и исходящие сообщения (с '?')	40
1.39.7 Короткие входящие (с '?')	40
1.39.8 Короткие исходящие (с "?")	41
1.40 Разные стили отображения	43
1.40.1 По умолчанию	43
1.40.2 LifelineStrategy	43
1.40.3 style strictuml	44
1.41 Скрытие ни с кем не связанных участников	44



1.42 Цвет сгруппированных сообщений	45
1.43 Основной фрейм	45
1.44 Slanted or odd arrows	45
1.45 Параллельные сообщения (<i>в архитектуре teoz</i>)	47
2 Диаграмма вариантов использования	48
2.1 Прецеденты	48
2.2 Акторы (Роли, Участники)	48
2.3 Изменение стиля отображения актора	49
2.3.1 Человечек из палочек (<i>по умолчанию</i>)	49
2.3.2 В виде бюста	49
2.3.3 Контур человечка	50
2.4 Описание прецедентов	50
2.5 Использование пакетов	51
2.6 Простой пример	52
2.7 Обобщение (наследование)	53
2.8 Использование заметок	53
2.9 Шаблоны	54
2.10 Смена направления стрелок	55
2.11 Разделение диаграмм	56
2.12 Направление слева направо	56
2.13 Параметры отображения	57
2.14 Полноценный пример	58
2.15 Бизнес прецеденты и акторы	58
2.15.1 Бизнес прецедент	59
2.15.2 Бизнес актор	59
2.16 Изменение цвета и стиля стрелок (встраиваемый синтаксис)	59
2.17 Изменение цвета и стиля элементов (встраиваемый синтаксис)	60
2.18 Display JSON Data on Usecase diagram	61
2.18.1 Simple example	61
3 Диаграммы классов	62
3.1 Декларирующий элемент	62
3.2 Взаимосвязи между классами	63
3.3 Метки на взаимосвязях	64
3.4 Использование не буквенных символов	65
3.5 Добавление методов	65
3.6 Указание видимости	67
3.7 Абстрактные и статические	67
3.8 Расширенное тело класса	68
3.9 Заметки и шаблоны	69
3.10 Больше о заметках	70
3.11 Создание заметки для поля данных или метода	70
3.11.1 □ Ограничения	70
3.11.2 Заметка к полю занных или методу	71
3.11.3 Заметка для разных методов с одинаковым именем	71
3.12 Заметки на взаимосвязях	71
3.13 Абстрактные классы и интерфейсы	72
3.14 Скрыть атрибуты, методы	73
3.15 Скрытие классов	74
3.16 Удаление классов	75
3.17 Hide, Remove or Restore tagged element or wildcard	75
3.18 Скрытие или Удаление классов баз связей	77
3.19 Использование дженериков	78
3.20 Определение метки	78
3.21 Пакеты	79
3.22 Стили пакетов	79
3.23 Пространства имён	81
3.24 Настройка создания пространств имён	82



3.25 Lollipop интерфейс	83
3.26 Изменение направления стрелок	83
3.27 Ассоциация классов	84
3.28 Ассоциация в одном и том же классе	85
3.29 Параметры отображения	86
3.30 Параметры отображения для шаблонов	87
3.31 Цветовой градиент	87
3.32 Помощь в расположении классов	88
3.33 Разделение больших файлов	89
3.34 Наследование и Реализация (Имплементация)	90
3.35 Изменение цвета и стиля связей или стрелок (синтаксис квадратных скобочек)	90
3.35.1 Стиль линий	90
3.35.2 Цвет линий	92
3.35.3 Толщина линий	92
3.35.4 Смешивание различных параметров	93
3.36 Изменение цвета и стиля связей или стрелок (встраиваемый синтаксис)	93
3.37 Изменение цвета и стиля класса (встраиваемый синтаксис)	94
3.38 Связи от/на члены класса	96
3.39 Группировка стрелок, ведущих к одному элементу	97
3.39.1 Установка параметра в значение 1 (без группировки)	97
3.39.2 Установка параметра в значение 2 (группировка если 2 и более)	98
3.39.3 Установка параметра в значение 3 (группировка если 3 и более)	98
3.39.4 Установка параметра в значение 4 (группировка если 4 и более)	99
3.40 Display JSON Data on Class or Object diagram	99
3.40.1 Simple example	99
3.41 Packages and Namespaces Enhancement	100
3.42 Qualified associations	101
3.42.1 Minimal example	101
3.42.2 Another example	101
3.43 Change diagram orientation	102
3.43.1 Top to bottom (<i>by default</i>)	102
3.43.2 With Graphviz (<i>layout engine by default</i>)	102
3.43.3 With Smetana (<i>internal layout engine</i>)	103
3.43.4 Left to right	104
3.43.5 With Graphviz (<i>layout engine by default</i>)	104
3.43.6 With Smetana (<i>internal layout engine</i>)	106
4 Диаграмма объектов	108
4.1 Определение объектов	108
4.2 Взаимосвязи между объектами	108
4.2.1 Если рассматривать более универсально,	108
4.2.2 Примеры:	109
4.2.3 Ещё больше примеров можно посмотреть:	109
4.3 Пример ассоциации объектов	109
4.4 Добавление полей	110
4.5 Общие с диаграммами классов функции	110
4.6 Map table or associative array	110
4.7 Program (or project) evaluation and review technique (PERT) with map	113
4.8 Display JSON Data on Class or Object diagram	114
4.8.1 Simple example	114
5 Диаграмма деятельности	116
5.1 Простая деятельность	116
5.2 Метка на стрелках	116
5.3 Изменение направления стрелки	116
5.4 Ветвления	117
5.5 Больше о ветках	118
5.6 Синхронизация	119
5.7 Длинное описание активности	120



5.8	Заметки	120
5.9	Разделы	121
5.10	Skinparam	122
5.11	Восьмиугольник	123
5.12	Полноценный пример	123
6	Диаграмма деятельности (новый синтаксис)	126
6.0.1	Преимущества нового синтаксиса	126
6.0.2	Переход на новый синтаксис	126
6.1	Простое действие	126
6.2	Старт активности, Стоп активности, Стоп поток, Быстрый стоп	126
6.3	Условия (оператор if)	128
6.3.1	1 вариант записи:	128
6.3.2	2 вариант записи	128
6.3.3	3 вариант записи	129
6.3.4	Ряд последовательных проверок условий (горизонтальное отображение)	129
6.3.5	Ряд последовательных проверок условий (вертикальное отображение)	130
6.4	Перебор множества условий для определенного параметра (оператор switch)	131
6.5	Остановка активности в теле условия	132
6.6	Повторяющийся цикл repeat (с проверкой условия в конце цикла)	134
6.7	Прерывание цикла repeat при помощи ключевого слова break	136
6.8	Обработка Goto и меток [label, goto]	137
6.9	Повторяющийся цикл while (с проверкой условия в начале цикла)	138
6.10	Распараллеливание активности [режим многопоточности: fork, fork again, end fork и merge]	140
6.10.1	Простой режим многопоточности	140
6.10.2	Режим многопоточности с объединением потоков в конце	141
6.10.3	Размещение комментария при возвращении к однопоточному режиму (реализация синтаксиса UML):	142
6.10.4	Еще один пример	143
6.11	Распараллеливание активности [режим разделения: split, split again и end split]	144
6.11.1	Разделение	144
6.11.2	Несколько входов	144
6.11.3	Несколько выходов	146
6.12	Заметки	147
6.13	Цвета	148
6.14	Линии вместо стрелок	149
6.15	Изменение цвета и стиля стрелок (синтаксис квадратных скобочек)	150
6.15.1	Стиль линий	150
6.15.2	Заметки на линиях	150
6.15.3	Цвет линий	151
6.15.4	Полный синтаксис	151
6.15.5	Комплексный пример	151
6.16	Коннектор	151
6.17	Цвет коннектора	152
6.18	Объединение (группировка) активностей	153
6.18.1	Группы	153
6.18.2	Разделы, Пакеты, Прямоугольники и Карточки	154
6.19	Дорожки	156
6.20	Прерывание активности	159
6.21	SDL (язык спецификаций с формальной семантикой)	161
6.22	Полноценный пример	163
6.23	Condition Style	164
6.23.1	Inside style (by default)	164
6.23.2	Diamond style	165
6.23.3	InsideDiamond (or FooI) style	166
6.24	Condition End Style	167
6.24.1	Diamond style (by default)	167
6.24.2	Horizontal line (hline) style	168



6.25 Using (global) style	169
6.25.1 Without style (<i>by default</i>)	169
6.25.2 With style	169
7 Диаграмма компонентов	172
7.1 Компоненты	172
7.2 Интерфейсы	172
7.3 Простой пример	173
7.4 Использование заметок	173
7.5 Группирование компонентов	174
7.6 Изменение направления стрелок	175
7.7 Use UML2 notation	176
7.8 Использовании нотации UML1	177
7.9 Use rectangle notation (remove UML notation)	177
7.10 Длинное описание	178
7.11 Индивидуальные цвета	178
7.12 Использование Sprite в стереотипах	178
7.13 Skinparam	179
7.14 Specific SkinParameter	180
7.14.1 componentStyle	180
7.15 Hide or Remove unlinked component	181
7.16 Hide, Remove or Restore tagged component or wildcard	182
7.17 Display JSON Data on Component diagram	184
7.17.1 Simple example	184
7.18 Port [port, portIn, portOut]	185
7.18.1 Port	185
7.18.2 PortIn	185
7.18.3 PortOut	186
7.18.4 Mixing PortIn & PortOut	186
8 Диаграмма развертывания	188
8.1 Объявление элемента	188
8.2 Declaring element (using short form)	190
8.2.1 Actor	190
8.2.2 Component	191
8.2.3 Interface	191
8.2.4 Usecase	191
8.3 Linking or arrow	191
8.4 Bracketed arrow style	194
8.4.1 Line style	194
8.4.2 Line color	195
8.4.3 Line thickness	195
8.4.4 Mix	196
8.5 Change arrow color and style (inline style)	196
8.6 Change element color and style (inline style)	197
8.7 Nestable elements	198
8.8 Packages and nested elements	198
8.8.1 Example with one level	198
8.8.2 Other example	199
8.8.3 Full nesting	200
8.9 Alias	205
8.9.1 Simple alias with as	205
8.9.2 Examples of long alias	205
8.10 Round corner	207
8.11 Specific SkinParameter	207
8.11.1 roundCorner	207
8.12 Appendix: All type of arrow line	209
8.13 Appendix: All type of arrow head or '0' arrow	209
8.13.1 Type of arrow head	209



8.13.2 Type of '0' arrow or circle arrow	210
8.14 Appendix: Test of inline style on all element	211
8.14.1 Simple element	211
8.14.2 Nested element	212
8.14.3 Without sub-element	212
8.14.4 With sub-element	213
8.15 Appendix: Test of style on all element	214
8.15.1 Simple element	214
8.15.2 Global style (on componentDiagram)	214
8.15.3 Style for each element	215
8.15.4 Nested element (without level)	219
8.15.5 Global style (on componentDiagram)	219
8.15.6 Style for each nested element	220
8.15.7 Nested element (with one level)	222
8.15.8 Global style (on componentDiagram)	222
8.15.9 Style for each nested element	223
8.16 Appendix: Test of stereotype with style on all element	225
8.16.1 Simple element	225
8.17 Display JSON Data on Deployment diagram	227
8.17.1 Simple example	227
8.18 Mixing Deployment (Usecase, Component, Deployment) element within a Class or Object diagram	227
8.18.1 Mixing all elements	227
8.19 Port [port, portIn, portOut]	229
8.19.1 Port	229
8.19.2 PortIn	230
8.19.3 PortOut	230
8.19.4 Mixing PortIn & PortOut	231
8.20 Change diagram orientation	232
8.20.1 Top to bottom (<i>by default</i>)	232
8.20.2 With Graphviz (<i>layout engine by default</i>)	232
8.20.3 With Smetana (<i>internal layout engine</i>)	233
8.20.4 Left to right	234
8.20.5 With Graphviz (<i>layout engine by default</i>)	234
8.20.6 With Smetana (<i>internal layout engine</i>)	235
9 Диаграмма состояний	237
9.1 Простое состояние	237
9.2 Change state rendering	237
9.3 Составное состояние	238
9.3.1 Внутренние состояния	238
9.3.2 Внутреннее состояние к внутреннему состоянию	239
9.4 Длинные имена	240
9.5 History [[H], [H*]]	241
9.6 Ветвление [fork, join]	241
9.7 Параллельные состояния	242
9.7.1 Горизонтальное разделение --	242
9.7.2 Вертикальное разделение 	243
9.8 Условия [choice]	244
9.9 Stereotypes full example [start, choice, fork, join, end, history, history*]	244
9.9.1 Start, choice, fork, join, end	244
9.9.2 History, history*	246
9.9.3 Minimal example with all stereotypes	246
9.10 Point [entryPoint, exitPoint]	246
9.11 Pin [inputPin, outputPin]	247
9.12 Expansion [expansionInput, expansionOutput]	248
9.13 Направления стрелок	249
9.14 Изменение цвета и отображения	250
9.15 Заметки	250



9.16 Заметки о переходах	251
9.17 Еще о заметках	251
9.18 Inline color	252
9.19 Skinparam	253
9.20 Changing style	254
9.21 Change state color and style (inline style)	256
9.22 Alias	257
9.23 Display JSON Data on State diagram	258
9.23.1 Simple example	258
9.24 State description	258
9.25 Style for Nested State Body	259
10 Timing Diagram	260
10.1 Declaring element or participant	260
10.2 Binary and Clock	261
10.3 Adding message	262
10.4 Relative time	262
10.5 Anchor Points	263
10.6 Participant oriented	263
10.7 Setting scale	264
10.8 Initial state	265
10.9 Intricated state	265
10.9.1 Intricated or undefined robust state	265
10.9.2 Intricated or undefined binary state	266
10.10 Hidden state	266
10.11 Hide time axis	268
10.12 Using Time and Date	268
10.13 Change Date Format	269
10.14 Manage time axis labels	269
10.14.1 Label on each tick (<i>by default</i>)	269
10.14.2 Manual label (<i>only when the state changes</i>)	270
10.15 Adding constraint	271
10.16 Highlighted period	271
10.17 Using notes	272
10.18 Adding texts	273
10.19 Complete example	274
10.20 Digital Example	275
10.21 Adding color	276
10.22 Using (global) style	277
10.22.1 Without style (<i>by default</i>)	277
10.22.2 With style	277
10.23 Applying Colors to specific lines	278
10.24 Compact mode	279
10.24.1 By default	279
10.24.2 Global mode with mode compact	280
10.24.3 Local mode with only compact on element	280
10.25 Scaling analog signal	281
10.25.1 Without scaling: 0-max (<i>by default</i>)	281
10.25.2 With scaling: min-max	282
10.26 Customise analog signal	282
10.26.1 Without any customisation (<i>by default</i>)	282
10.26.2 With customisation (on scale, ticks and height)	283
10.27 Order state of robust signal	283
10.27.1 Without order (<i>by default</i>)	283
10.27.2 With order	284
10.27.3 With order and label	284
10.28 Defining a timing diagram	285
10.28.1 By Clock (@clk)	285



10.28.2 By Signal (@S)	285
10.28.3 By Time (@time)	286
10.29 Annotate signal with comment	287
11 Display JSON Data	289
11.1 Complex example	289
11.2 Highlight parts	290
11.3 Using different styles for highlight	290
11.4 JSON basic element	291
11.4.1 Synthesis of all JSON basic element	291
11.5 JSON array or table	292
11.5.1 Array type	292
11.5.2 Minimal array or table	293
11.5.3 Number array	293
11.5.4 String array	293
11.5.5 Boolean array	293
11.6 JSON numbers	293
11.7 JSON strings	294
11.7.1 JSON Unicode	294
11.7.2 JSON two-character escape sequence	294
11.8 Minimal JSON examples	295
11.9 Empty table or list	296
11.10 Using (global) style	296
11.10.1 Without style (<i>by default</i>)	296
11.10.2 With style	297
11.11 Display JSON Data on Class or Object diagram	298
11.11.1 Simple example	298
11.11.2 Complex example: with all JSON basic element	298
11.12 Display JSON Data on Deployment (Usecase, Component, Deployment) diagram	299
11.12.1 Simple example	299
11.13 Display JSON Data on State diagram	300
11.13.1 Simple example	300
11.14 Creole on JSON	301
12 Display YAML Data	303
12.1 Complex example	303
12.2 Specific key (with symbols or unicode)	304
12.3 Highlight parts	304
12.3.1 Normal style	304
12.3.2 Customised style	305
12.4 Using different styles for highlight	306
12.5 Using (global) style	307
12.5.1 Without style (<i>by default</i>)	307
12.5.2 With style	307
12.6 Creole on YAML	308
13 Network Diagram with nwdiag	310
13.1 Простая диаграмма	310
13.1.1 Определите сеть	310
13.1.2 Определение некоторых элементов или серверов в сети	310
13.1.3 Полный пример	310
13.2 Define multiple addresses	311
13.3 Grouping nodes	312
13.3.1 Define group inside network definitions	312
13.3.2 Define group outside of network definitions	313
13.3.3 Define several groups on same network	313
13.3.4 Example with 2 group	313
13.3.5 Example with 3 groups	314
13.4 Extended Syntax (for network or group)	315



13.4.1 Network	315
13.4.2 Group	316
13.5 Using Sprites	317
13.6 Using OpenIconic	318
13.7 Same nodes on more than two networks	319
13.8 Peer networks	320
13.9 Peer networks and group	320
13.9.1 Without group	320
13.9.2 Group on first	321
13.9.3 Group on second	322
13.9.4 Group on third	323
13.10 Add title, caption, header, footer or legend on network diagram	324
13.11 With or without shadow	325
13.11.1 With shadow (by default)	325
13.11.2 Without shadow	325
13.12 Change width of the networks	326
13.12.1 First example	326
13.12.2 Second example	329
13.13 Other internal networks	332
13.14 Using (global) style	334
13.14.1 Without style (by default)	334
13.14.2 With style	335
13.15 Appendix: Test of all shapes on Network diagram (nwdiag)	336
14 Salt (Wireframe)	339
14.1 Простые виджеты	339
14.2 Text area	339
14.3 Open, close dropdown	340
14.4 Использование сетки	341
14.5 Группа	341
14.6 Использование разделителя	342
14.7 Древовидный виджет	342
14.8 Таблица дерева иерархии	343
14.9 Окружающие скобки	344
14.10 Добавление вкладок	344
14.11 Использование меню	345
14.12 Продвинутая таблица	346
14.13 Полосы прокрутки	346
14.14 Цвет элементов	347
14.15 Creole on Salt	348
14.16 Спрайты	350
14.17 OpenIconic	350
14.18 Add title, header, footer, caption or legend	351
14.19 Zoom, DPI	352
14.19.1 Whitout zoom (by default)	352
14.19.2 Scale	352
14.19.3 DPI	352
14.20 Include Salt "on activity diagram"	353
14.21 Include salt "on while condition of activity diagram"	355
14.22 Include salt "on repeat while condition of activity diagram"	356
14.23 Skipparam	357
14.24 Style	358
15 ArchiMate Diagram	359
15.1 Archimate keyword	359
15.2 Defining Junctions	359
15.3 Example 1	360
15.4 Example 2	361
15.5 List possible sprites	362



15.6 ArchiMate Macros	362
15.6.1 Archimate Macros and Library	362
15.6.2 Archimate elements	362
15.6.3 Archimate relationships	363
15.6.4 Appendix: Examples of all Archimate RelationTypes	364
16 Диаграмма Ганта	368
16.1 Объявление задач	368
16.1.1 Объем работы	368
16.1.2 Начало	369
16.1.3 Конец	369
16.1.4 Start/End	370
16.2 Однострочное объявление (с конъюнкцией and)	370
16.3 Добавление ограничений	370
16.4 Короткие имена и псевдонимы задач (алиасы)	371
16.5 Задачи с одинаковыми наименованиями	371
16.6 Настройка цветов	372
16.7 Статус завершения	372
16.7.1 Добавление процента зависимости завершения	372
16.7.2 Изменить цвет завершения (при помощи модуля <style>)	372
16.7.3 Изменение цвета незаконченной части задачи (при помощи модуля <style>)	373
16.8 Вехи	374
16.8.1 Относительная веха (использование ограничений)	374
16.8.2 Абсолютная веха (использование фиксированной даты)	374
16.8.3 Веха максимального завершения задач	374
16.9 Гиперссылки	375
16.10 Календарь	375
16.11 Раскраска дней	375
16.12 Календарные шкалы	376
16.12.1 Дневная шкала (<i>по умолчанию</i>)	376
16.12.2 Недельная шкала	376
16.12.3 Месячная шкала	377
16.12.4 Квартальная шкала	377
16.12.5 Годовая шкала	378
16.12.6 Шкала точного периода командой <code>between</code>	378
16.12.7 Без точного периода	378
16.12.8 С точным периодом	379
16.13 Масштабирование (пример для всех шкал)	379
16.13.1 Масштаб на дневной шкале	379
16.13.2 Без увеличения	379
16.13.3 С увеличением	380
16.13.4 Масштаб на недельной шкале	380
16.13.5 Без увеличения	380
16.13.6 С увеличением	381
16.13.7 Масштаб на месячной шкале	381
16.13.8 Без увеличения	381
16.13.9 С увеличением	381
16.13.10 Масштаб на квартальной шкале	382
16.13.11 Без увеличения	382
16.13.12 С увеличением	382
16.13.13 Масштаб на годовой шкале	382
16.13.14 Без увеличения	382
16.13.15 С увеличением	383
16.14 Недельный масштаб с номерами недель или датами	383
16.14.1 С указанием номеров недель (<i>настройка по умолчанию</i>)	383
16.14.2 С номерами недель (<i>начиная с 1</i>)	383
16.14.3 С указанием дат	384
16.15 Объявление дней нерабочими (закрытыми)	384



16.16 Зависимость размера недели от количества нерабочих дней в ней	385
16.17 Рабочие дни	385
16.18 Упрощенная последовательность задач	386
16.19 Работа с ресурсами	386
16.20 Настройки отображения ресурсов	387
16.20.1 Ресурсы отображены (<i>по умолчанию</i>)	387
16.20.2 Имена ресурсов скрыты	387
16.20.3 Скрыт подвал с ресурсами	388
16.20.4 Ресурсы скрыты на задачах и в подвале	388
16.21 Горизонтальный разделитель	388
16.22 Вертикальный разделитель	389
16.23 Сложный пример	389
16.24 Комментарии	390
16.25 Использование стилей	390
16.25.1 Настройки по умолчанию	390
16.25.2 С модулем <code><style></code>	390
16.25.3 Индивидуальная настройки графических стилей (полный пример)	392
16.25.4 Минималистичный стиль диаграммы	394
16.26 Добавление примечания	395
16.27 Приостановить выполнение задач	397
16.28 Изменение цвета связей между задачами	397
16.29 Задачи или вехи на одной линии	398
16.30 Выделение цветом дня "Сегодня"	399
16.31 Задачи между двумя вехами	399
16.32 Grammar and verbal form	399
16.33 Заголовок, колонитулы, подписи, легенда	399
16.34 Изменение цвета легенды	400
16.35 Отключение подвала диаграммы (пример для всех масштабов)	400
16.36 Language of the calendar	402
16.36.1 English (<i>en, by default</i>)	403
16.36.2 Deutsch (de)	403
16.36.3 Japanese (ja)	403
16.36.4 Chinese (zh)	404
16.36.5 Korean (ko)	404
16.37 Удаление задач или вех	404
16.38 Отражение начала проекта, задачи или вехи за несколько дней до или после сегодняшнего дня	405
16.39 Изменение расположения ярлыка имени задачи	405
16.39.1 Ярлык имени расположен рядом с блоком задачи (<i>настройка по умолчанию</i>)	405
16.39.2 Ярлык в первой колонке	406
16.39.3 Ярлык в последней колонке	407
17 MindMap	409
17.1 Синтаксис OrgMode	409
17.2 Markdown синтаксис	410
17.3 Арифметическая нотация	410
17.4 Многострочные блоки	411
17.5 Диаграммы связей с несколькими исходными узлами	412
17.6 Цвета	413
17.6.1 Объявление настроек в строке блока	413
17.6.2 Объявление в блоке <code><style></code>	414
17.7 Оформление без рамки	416
17.8 Изменение направления диаграммы	417
17.9 Изменение направления развёртки всей диаграммы	417
17.9.1 Слева направо (<i>настройка по умолчанию</i>)	418
17.9.2 Сверху вниз	418
17.9.3 Справа налево	418
17.9.4 Снизу вверх	419
17.10 Полный пример	419



17.11 Стили диаграммы	420
17.11.1 блоки, глубина	420
17.11.2 без рамки	421
17.12 Перенос строк	422
17.13 Использования языка разметки Creole в диаграммах связей	423
18 Структура разбиения работ (WBS)	426
18.1 Синтаксис OrgMode	426
18.2 Change direction	427
18.3 Arithmetic notation	427
18.4 Multilines	428
18.5 Removing box	428
18.5.1 Boxless on Arithmetic notation	429
18.5.2 Several boxless node	429
18.5.3 All boxless node	429
18.5.4 Boxless on OrgMode syntax	430
18.5.5 Several boxless node	430
18.5.6 All boxless node	430
18.6 Colors (with inline or style color)	431
18.7 Using style	432
18.8 Word Wrap	433
18.9 Add arrows between WBS elements	435
18.10 Creole on WBS diagram	436
19 Maths	438
19.1 Standalone diagram	439
19.2 How is this working?	439
20 Entity Relationship Diagram	440
20.1 Information Engineering Relations	440
20.2 Entities	440
20.3 Полный пример	441
21 Общие команды в PlantUML	443
21.0.1 Глобальные элементы	443
21.0.2 Описание синтаксиса Creole	443
21.0.3 Команда управления стилями	443
21.1 Комментарии	443
21.1.1 Простой комментарий	443
21.1.2 Block comment	443
21.2 Zoom	444
21.3 Title	444
21.4 Caption	445
21.5 Footer and header	446
21.6 Legend the diagram	446
21.7 Appendix: Examples on all diagram	447
21.7.1 Activity	447
21.7.2 Archimate	448
21.7.3 Class	449
21.7.4 Component, Deployment, Use-Case	449
21.7.5 Gantt project planning	450
21.7.6 Object	450
21.7.7 MindMap	451
21.7.8 Network (nwdiag)	451
21.7.9 Sequence	452
21.7.10 State	453
21.7.11 Timing	453
21.7.12 Work Breakdown Structure (WBS)	454
21.7.13 Wireframe (SALT)	455



21.8 Appendix: Examples on all diagram with style	455
21.8.1 Activity	456
21.8.2 Archimate	458
21.8.3 Class	459
21.8.4 Component, Deployment, Use-Case	461
21.8.5 Gantt project planning	462
21.8.6 Object	464
21.8.7 MindMap	465
21.8.8 Network (nwdiag)	466
21.8.9 Sequence	468
21.8.10 State	469
21.8.11 Timing	471
21.8.12 Work Breakdown Structure (WBS)	472
21.8.13 Wireframe (SALT)	473
21.9 Mainframe	474
21.10 Appendix: Examples of Mainframe on all diagram	475
21.10.1 Activity	475
21.10.2 Archimate	475
21.10.3 Class	476
21.10.4 Component, Deployment, Use-Case	476
21.10.5 Gantt project planning	476
21.10.6 Object	477
21.10.7 MindMap	477
21.10.8 Network (nwdiag)	477
21.10.9 Sequence	478
21.10.10 State	478
21.10.11 Timing	478
21.10.12 Work Breakdown Structure (WBS)	479
21.10.13 Wireframe (SALT)	479
21.11 Appendix: Examples of title, header, footer, caption, legend and mainframe on all diagram	480
21.11.1 Activity	480
21.11.2 Archimate	480
21.11.3 Class	481
21.11.4 Component, Deployment, Use-Case	482
21.11.5 Gantt project planning	482
21.11.6 Object	483
21.11.7 MindMap	484
21.11.8 Network (nwdiag)	484
21.11.9 Sequence	485
21.11.10 State	486
21.11.11 Timing	486
21.11.12 Work Breakdown Structure (WBS)	487
21.11.13 Wireframe (SALT)	488
22 Creole синтаксис	490
22.1 Emphasized text	490
22.2 Lists	490
22.3 Escape character	491
22.4 Headings	491
22.5 Emoji	492
22.5.1 Unicode block 26	493
22.6 Horizontal lines	493
22.7 Links	494
22.8 Code	494
22.9 Table	495
22.9.1 Create a table	495
22.9.2 Align fields using Table	496
22.9.3 Add color on rows or cells	498



22.9.4 Add color on border and text	498
22.9.5 No border or same color as the background	498
22.9.6 Bold header or not	498
22.10 Tree	499
22.11 Special characters	501
22.12 Legacy HTML	502
22.12.1 Common HTML element	503
22.12.2 Subscript and Superscript element [sub, sup]	504
22.13 OpenIconic	504
22.14 Appendix: Examples of "Creole List" on all diagrams	505
22.14.1 Activity	505
22.14.2 Class	506
22.14.3 Component, Deployment, Use-Case	507
22.14.4 Gantt project planning	508
22.14.5 Object	508
22.14.6 MindMap	509
22.14.7 Network (nwdiag)	509
22.14.8 Note	510
22.14.9 Sequence	510
22.14.10 State	511
22.14.11 WBS	512
22.15 Appendix: Examples of "Creole horizontal lines" on all diagrams	513
22.15.1 Activity	513
22.15.2 Class	514
22.15.3 Component, Deployment, Use-Case	515
22.15.4 Gantt project planning	516
22.15.5 Object	516
22.15.6 MindMap	517
22.15.7 Network (nwdiag)	518
22.15.8 Note	518
22.15.9 Sequence	519
22.15.10 State	520
22.15.11 WBS	521
22.16 Style equivalent (between Creole and HTML)	522
23 Defining and using sprites	524
23.1 Inline SVG sprite	524
23.2 Changing colors	526
23.3 Encoding Sprite	526
23.4 Importing Sprite	527
23.5 Examples	527
23.6 StdLib	528
23.7 Listing Sprites	528
24 Skinparam command	530
24.1 Usage	530
24.2 Вложенные	530
24.3 Черно-белый	530
24.4 Shadowing	531
24.5 Reverse colors	531
24.6 Colors	532
24.7 Font color, name and size	533
24.8 Text Alignment	533
24.9 Examples	534
24.10 List of all skinparam parameters	538
24.10.1 Command Line: -language command	538
24.10.2 Command: help skinparams	538
24.10.3 Command: skinparameters	538
24.10.4 All Skin Parameters on the Ashley's PlantUML Doc	541



25 Preprocessing	542
25.1 Variable definition [=, ?=]	542
25.2 Boolean expression	543
25.2.1 Boolean representation [0 is false]	543
25.2.2 Boolean operation and operator [&&, , ()]	543
25.2.3 Boolean builtin functions [%false(), %true(), %not(<exp>), %boolval(<exp>)]	543
25.3 Conditions [!if, !else, !elseif, !endif]	543
25.4 While loop [!while, !endwhile]	544
25.4.1 While loop (on Activity diagram)	544
25.4.2 While loop (on Mindmap diagram)	545
25.4.3 While loop (on Component/Deployment diagram)	546
25.5 Procedure [.procedure, !endprocedure]	546
25.6 Return function [!function, !endfunction]	547
25.7 Default argument value	548
25.8 Unquoted procedure or function [!unquoted]	549
25.9 Keywords arguments	550
25.10 Including files or URL [!include, !include_many, !include_once]	550
25.11 Including Subpart [!startsub, !endsub, !includesub]	551
25.12 Builtin functions [%]	552
25.13 Logging [!log]	553
25.14 Memory dump [!dump_memory]	553
25.15 Assertion [!assert]	553
25.16 Building custom library [!import, !include]	554
25.17 Search path	554
25.18 Argument concatenation [##]	554
25.19 Dynamic invocation [%invoke_procedure(), %call_user_func()]	555
25.20 Evaluation of addition depending of data types [+]	556
25.21 Preprocessing JSON	556
25.22 Including theme [!theme]	556
25.23 Migration notes	557
25.24 %splitstr builtin function	557
25.25 %splitstr_regex builtin function	558
25.26 %get_all_theme builtin function	559
25.27 %get_all_stdlib builtin function	560
25.27.1 Compact version (only standard library name)	560
25.27.2 Detailed version (with version and source)	560
25.28 %random builtin function	562
25.29 %boolval builtin function	562
26 Unicode	563
26.1 Examples	563
26.2 Charset	565
26.3 Using Unicode Character on PlantUML	565
27 PlantUML Standard Library	566
27.0.1 Standard Library Overview	566
27.0.2 Contribution from the Community	566
27.1 List of Standard Library	566
27.2 ArchiMate [archimate]	568
27.2.1 List possible sprites	569
27.3 Amazon Labs AWS Library [awslib]	570
27.4 Azure library [azure]	571
27.5 C4 Library [C4]	572
27.6 Cloud Insight [cloudinsight]	572
27.7 Cloudogu [cloudogu]	573
27.8 EDGY: An Open Source tool for collaborative Enterprise Design [edgy]	574
27.8.1 Basic Elements and Interconnections	574
27.8.2 Elements	574
27.8.3 Relationships	575



27.8.4 Facets	577
27.8.5 Identity	577
27.8.6 Architecture	577
27.8.7 Experience	578
27.8.8 Intersections	578
27.8.9 Alternative visual styling	579
27.9 Elastic library [elastic]	580
27.10 Google Material Icons [material]	582
27.11 Kubernetes [kubernetes]	583
27.12 Logos [logos]	584
27.13 Office [office]	586
27.14 Open Security Architecture (OSA) [osa]	588
27.15 Tupadr3 library [tupadr3]	591
27.16 AWS library [aws]	592

