Guidelines for Using PlantUML

April 21, 2015

Version 1.0

Abstract

This document provides instructions for installing and using PlantUML. It also provides guidance on using the software to create sequence diagrams for Conexxus documentation.

Contributors

Linda Toth, Conexxus
Brian Russell, VeriFone
Susan Chan, W. Capra

Revision History

Revision Date	Revision Number	Revision Editor(s)	Revision Changes
April 21, 2015	1.0	Linda Toth, Conexxus	Release Version
February 2, 2015	Draft 0.3	Linda Toth, Conexxus	Incorporated additional feedback from reviews
February 2, 2015	Draft 0.2	Linda Toth, Conexxus	Incorporated feedback from reviews
January 23, 2015	Draft 0.1	Linda Toth, Conexxus	Initial Draft

Copyright Statement

Disclaimers

Conexxus makes no warranty, express or implied, about, nor does it assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, product, or process described in these materials. Although Conexxus uses reasonable best efforts to ensure this work product is free of any third party intellectual property rights (IPR) encumbrances, it cannot guarantee that such IPR does not exist now or in the future. Conexxus further notifies all users of this standard that their individual method of implementation may result in infringement of the IPR of others. Accordingly, all users are encouraged to carefully review their implementation of this standard and obtain appropriate licenses where needed.

Table of Contents

1 Introduction

This document provides instructions for installing and using PlantUML. It also provides guidance on using the software to create sequence diagrams for Conexxus documentation. It gives instructions and practical examples covering typical diagrams a committee might put together. Additional features and controls may be available. Consult the PlantUML Language Reference Guide for additional details.

PlantUML is an open source tool that can create a variety of diagrams, including sequence diagrams. While it can be integrated into various development environments, this guide will focus on using the standalone executable jar file operating on text files.

Each sequence diagram is created from a set of instructions contained in a text file. Any editor that does not introduce formatting characters (e.g., notepad, wordpad) may be used to create or edit the files. Each sequence diagram will have its own text file. PlantUML creates corresponding graphic file(s) based on the instructions found in the text file.

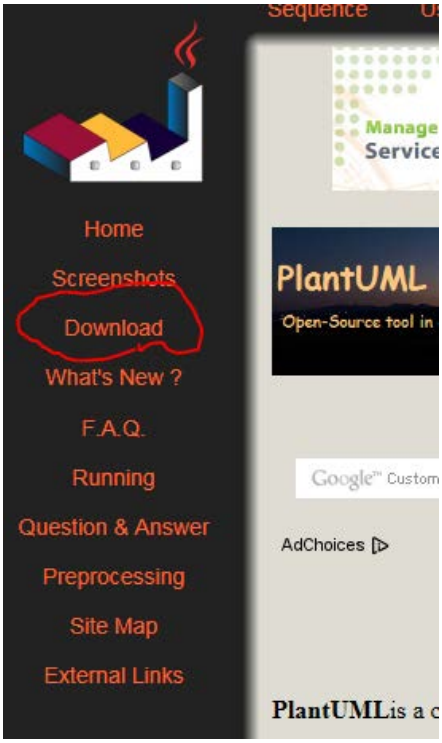
The only requirement to run PlantUML is to have a Java Runtime Environment (JRE) installed on your PC. Your computer likely already has a JRE installed. If not, you can download the latest version at http://java.com/en/.

2 Installing PlantUML

PlantUML can be downloaded at: http://plantuml.sourceforge.net/index.html.

***WARNING:* The website has many clickable ads/questionable downloads, so be very careful of where you click!**

To download the executable jar file and reference manual, click on the download link on the left side of the webpage.

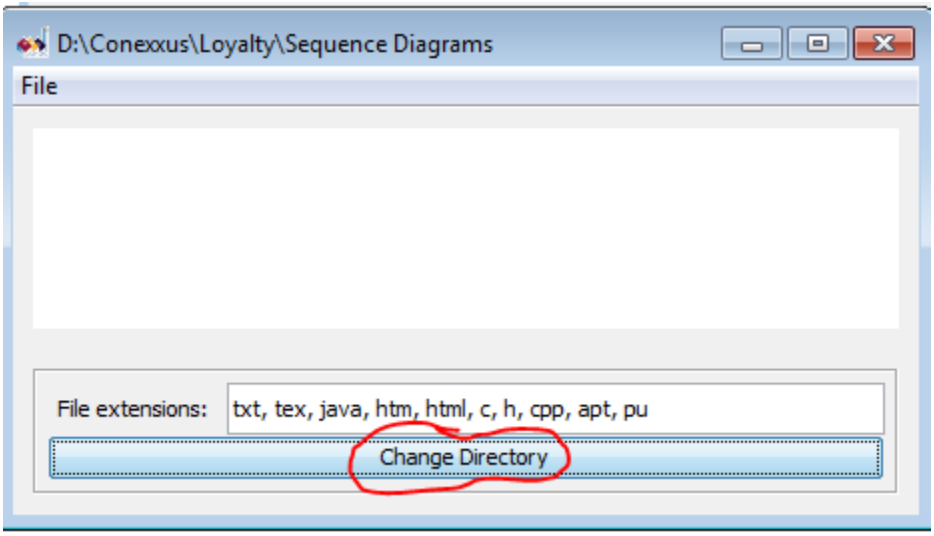


Download the latest compiled jar file as well as the reference manual to a folder of your choosing on your PC. Again, be certain to click only on the PlantUML specific downloads. There are many other clickable downloads/ads.

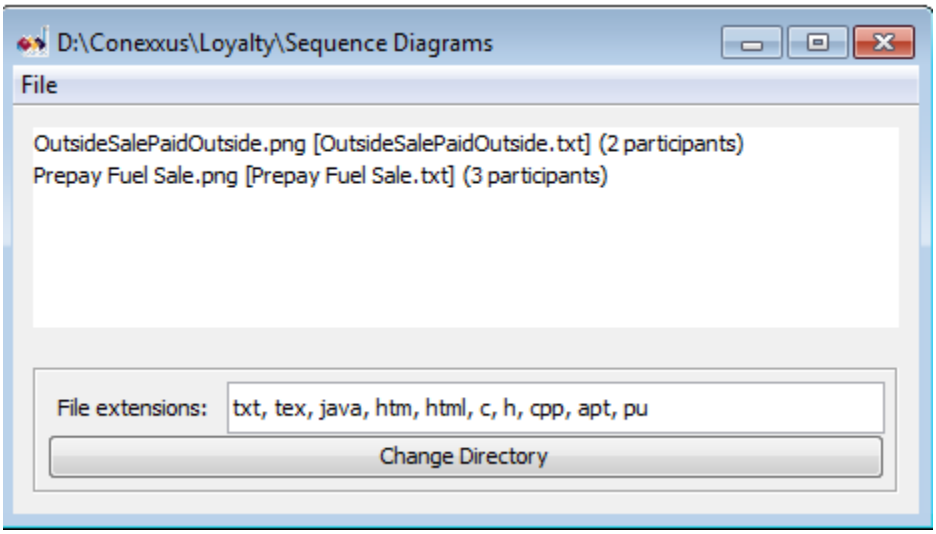
3 Running PlantUML

Create a folder that will contain your working files.

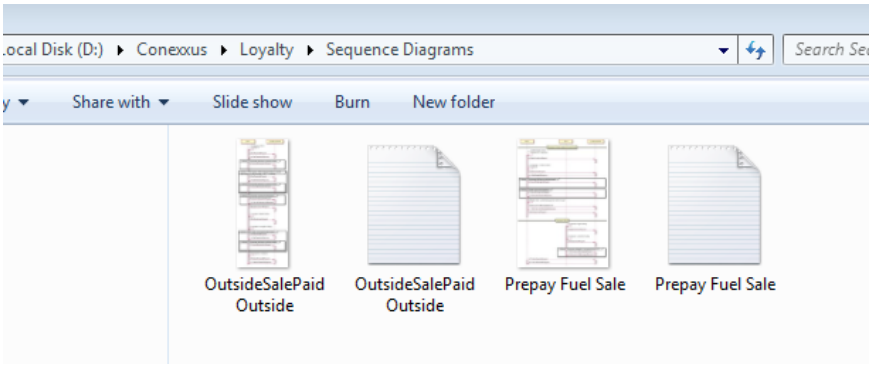
Navigate to the folder where PlantUML was downloaded and double click on the jar file. Use the “change directory” button on the PlantUML GUI window to navigate to your working files folder.



Once the working files folder is populated with one or more text files, PlantUML will automatically detect them, process them, and create corresponding graphic drawing files. The file processing results are shown in the PlantUML GUI.

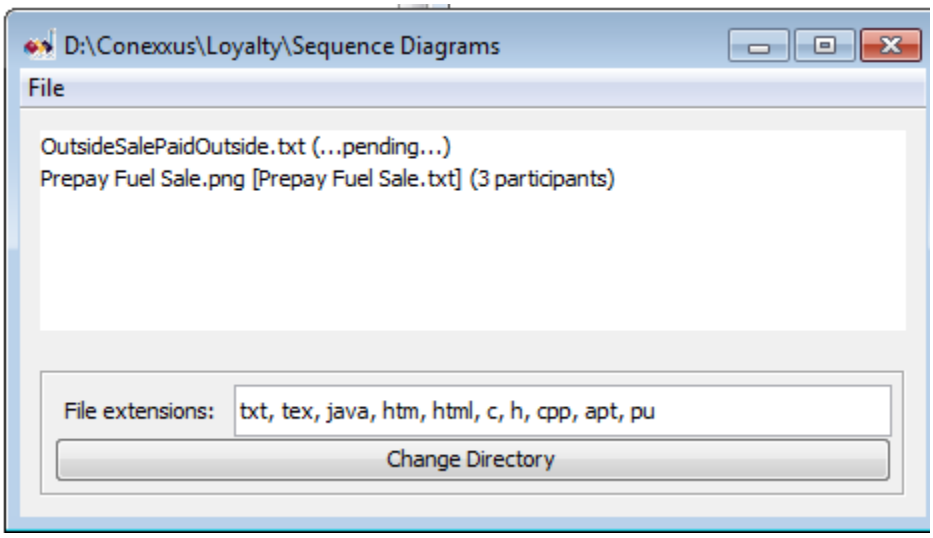


The graphic files are created in the working directory.



They can be viewed by double clicking on the file in Windows Explorer (opens with default graphics viewer) or by double clicking on the file in the PlantUML GUI (opens in a GUI window)

At program startup or when the text file is changed and saved, PlantUML automatically reprocesses the file. While that is happening, you will see a "....pending..." message by the file.



By default, PlantUML generates .png files. Png files are preferred for Conexxus documents because of how it renders fine lines. As you are making changes in your text file, you can keep the png file open next to the text file. The graphic file will regenerate when the text file is saved, so you can quickly see the updated results.

4 Sequence Diagram Guidelines

4.1 PlantUML Text File

Each sequence diagram should have a separate plain text file that will be used to generate the sequence diagram and should also be archived on the Conexxus website. Descriptive file names are recommended. This file contains processing instructions starting with `@startuml` on the first line and `@enduml` on the last line. All other processing instructions appear in between in this order:

```
@startuml
[Formatting/Style Instructions ...]
[Participants ...]
[Diagram Specific Instructions ...]
@enduml
```

`Conexxus_PlantUMLTemplate.txt` is a template file, which has the sections defined and populated with comments, may be used to start a new diagram.

4.2 Formatting and Style Instructions

So that all Conexxus sequence diagrams are consistent, use PlantUML defaults along with the following instructions:

```
skinparam defaultFontName Arial
'lifeline
skinparam sequenceLifeLineBackgroundColor #E0E0E0
'divider
skinparam sequenceDividerFontColor black
skinparam sequenceDividerFontStyle plain
skinparam sequenceDividerBackgroundColor #FEFECE
'groupings
skinparam sequenceGroupBackgroundColor whitesmoke
skinparam sequenceGroupHeaderFontColor black
skinparam sequenceGroupHeaderFontStyle italic
skinparam sequenceGroupHeaderFontSize 13
'else
skinparam sequenceGroupFontStyle italic bold
skinparam sequenceGroupFontSize 13

autonumber  "<sup>00</sup>"
hide footbox
```

These instructions are in `ConexxusSkinParam.txt` and can be included in your text file using the instruction:

```
!include ConexxusSkinParam.txt
```

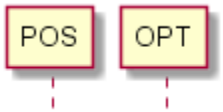
If you include the file, copy it into your working folder so PlantUML can find it. Regardless of whether you copy and paste the standard format instructions, or include the Conexxus file, place this as the next set of instructions after `@startuml`. Do not include any other style or formatting instructions.

4.3 Participants

Although PlantUML does not require you to define participants (i.e., entities), it is recommended best practice to define all participants together and before they are used. This can be done right after the formatting instructions using the keyword `participant`. Participants are shown left to right in the order that they are declared.

Example:

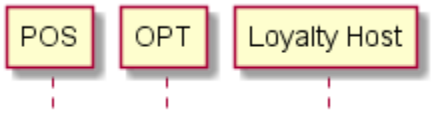
```
participant POS
participant OPT
```



If the participant name is long or needs spaces, declare the participant using a short name and the display name. The short name can be used in diagram specific processing instructions.

Example:

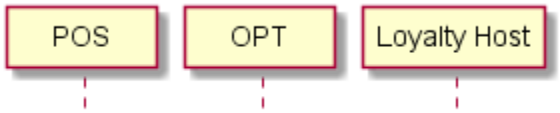
```
participant POS
participant OPT
participant LH as "Loyalty Host"
```



To control the layout of the diagram, you can add spaces in the display name to better position the participants.

Example:

```
participant POS as "    POS    "
participant OPT as "    OPT    "
participant LH as "Loyalty Host"
```



4.4 Diagram Specific Instructions

Diagram specific instructions follow participant declarations and precede the `@enduml` instruction. Each script fragment below assumes that participants have been defined similar to those examples found in Section 4.3 Participants.

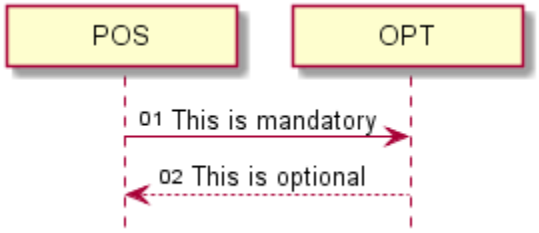
4.4.1 Messages

Messages between participants are specified by defining the two participants, the direction, the line type, and the corresponding text to display above the message.

Mandatory messages are rendered with solid lines, indicated in the instructions by "`->`". Optional steps are rendered with dashed lines, indicated in the instruction by "`-->`".

Example:

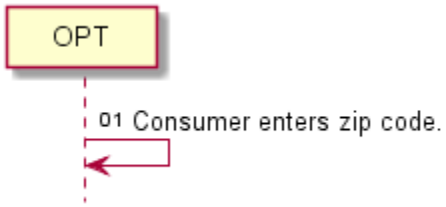
```
POS->OPT:  This is mandatory
OPT-->POS:  This is optional
```



External actions should be shown as a message to self.

Example:

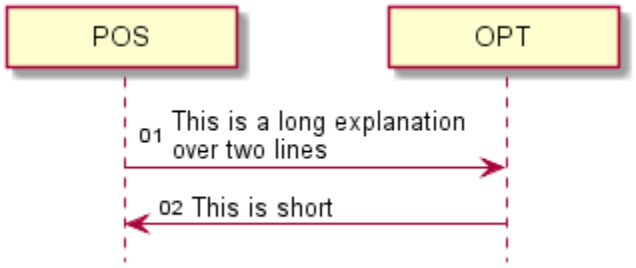
```
OPT -> OPT:  Consumer enters zip code.
```



Message text can be formatted to appear on multiple lines by using (\n) which will force a new line.

Example:

```
POS->OPT:  This is a long explanation \nover two lines
OPT->POS:  This is short
```



Superscript numbers precede each message. The statement
`autonumber "<sup>00</sup>"`
in the `ConexxusSkinParam.txt` file creates the numbers. The Conexxus sequence diagram document can use these numbers when referring to various messages in explanations and/or notes.

4.4.2 Lifelines

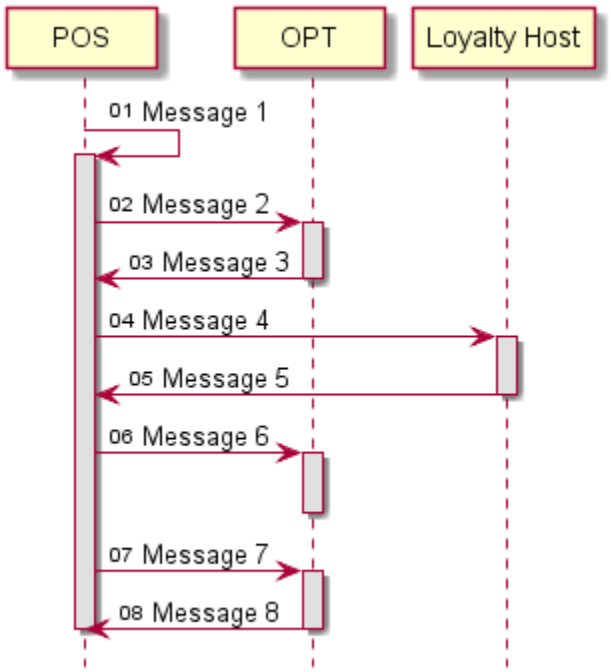
Use the keywords `activate` and `deactivate` to control the lifeline of a participant. The lifeline should be inactive (will be shown as dashed) when not in use. These keywords act on the previous message. Therefore place the activate keyword *after* the message that starts the lifeline and the deactivate keyword *after* the message the ends the lifeline. Messages normally connect two (and only two) participants. Messages should not pass through another active participant.

In this example, the POS is activated by message 1 and remains active until the last response is received. Both the OPT and the Loyalty Host (defined as LH in the participant list) are only active when receiving or responding to a message.

```
POS->POS: Message 1
activate POS
POS->OPT: Message 2
activate OPT
OPT->POS: Message 3
deactivate OPT

POS->LH: Message 4
activate LH
LH->POS: Message 5
deactivate LH

POS->OPT: Message 6
activate OPT
deactivate OPT
POS->OPT: Message 7
activate OPT
OPT->POS: Message 8
deactivate OPT
deactivate POS
```

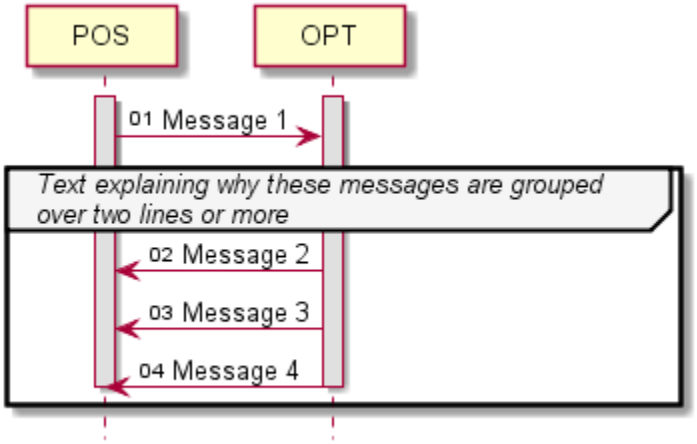


4.4.3 Groupings

Messages can be grouped together in a box by using the keyword `group`, specifying the text to include in the grouping, ending with the keyword `end`. The explanation can be formatted to force a new line with '`\n`'.

Example:

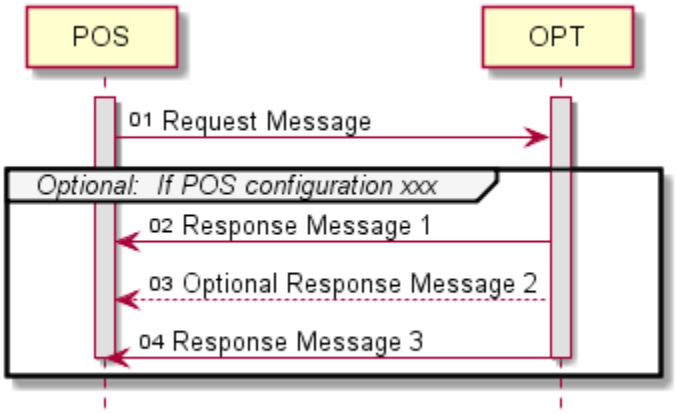
```
POS->OPT: Message 1
group Text explaining why these messages are grouped \nover
two lines or more
    OPT->POS: Message 2
    OPT->POS: Message 3
    OPT->POS: Message 4
end
```



Groupings can be especially useful if one or more messages are dependent on certain conditions. In this case, use “Optional:” in the grouping text. Messages in the grouping that are mandatory if the condition is met should be shown as a solid line. Dashed lines indicate optional messages even if the controlling condition is met.

Example:

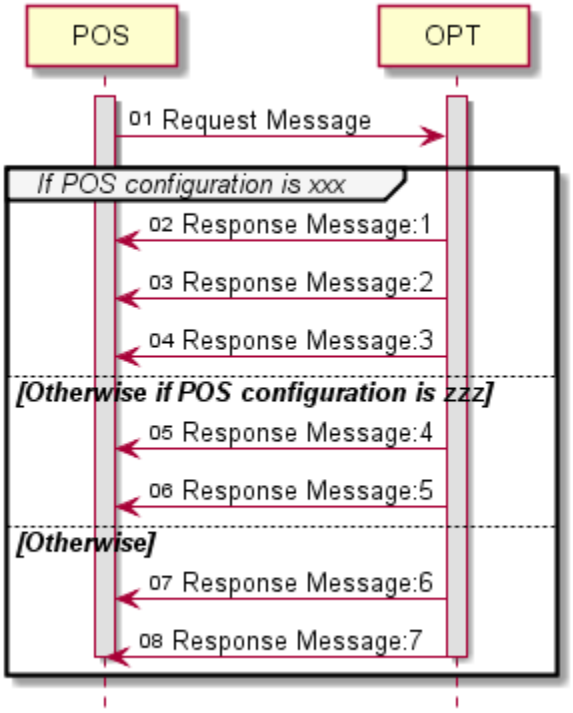
```
POS->OPT: Request Message
group Optional:  If POS configuration xxx
    OPT->POS: Response Message 1
    OPT-->POS: Optional Response Message 2
    OPT->POS: Response Message 3
end
```



If a choice of messages should be sent based on various alternative conditions, the keywords `group` and `else` should be used together, specifying “If” and “Otherwise” in the text as appropriate. This could typically be used to capture alternate use case flows.

Example:

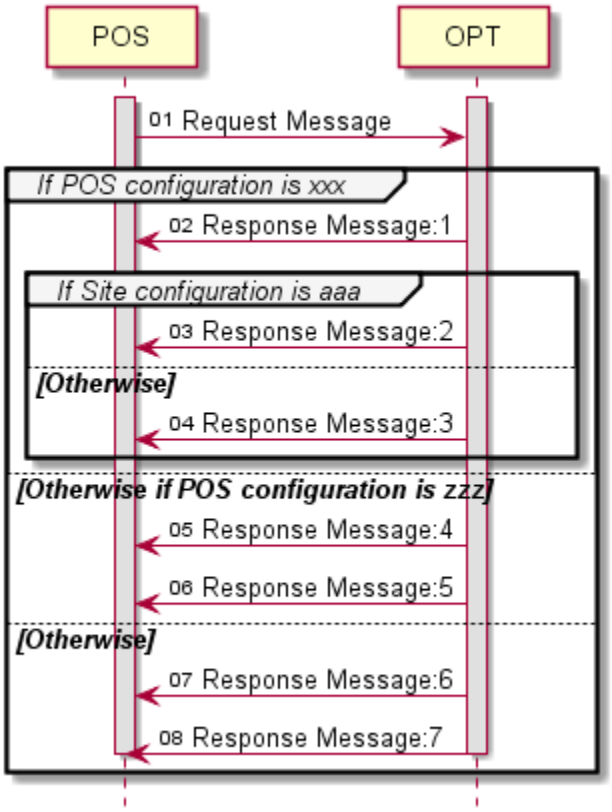
```
POS->OPT: Request Message
group If POS configuration is xxx
    OPT->POS: Response Message:1
    OPT->POS: Response Message:2
    OPT->POS: Response Message:3
else Otherwise if POS configuration is zzz
    OPT->POS: Response Message:4
    OPT->POS: Response Message:5
else Otherwise
    OPT->POS: Response Message:6
    OPT->POS: Response Message:7
end
```



Message groups may be logically nested.

Example:

```
POS->OPT: Request Message
group If POS configuration is xxx
    OPT->POS: Response Message:1
    group If Site configuration is aaa
     OPT->POS: Response Message:2
    else Otherwise
        OPT->POS: Response Message:3
    end
else Otherwise if POS configuration is zzz
    OPT->POS: Response Message:4
    OPT->POS: Response Message:5
else Otherwise
    OPT->POS: Response Message:6
    OPT->POS: Response Message:7
end
```



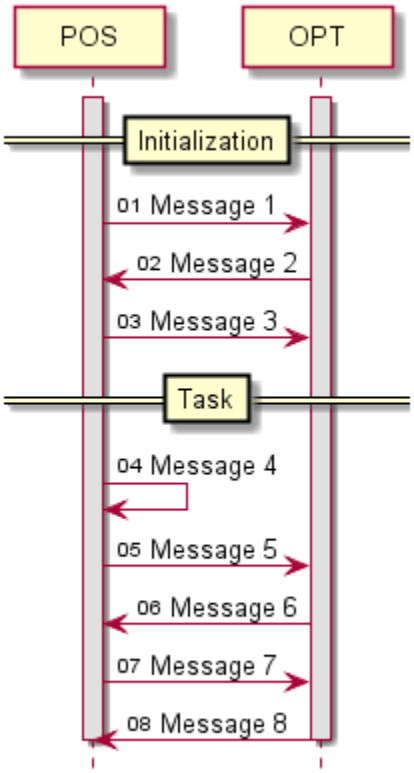
4.4.4 Dividers

Use a divider to separate a sequence diagram into logical or functional sections. A divider is specified by "==" on both sides of the text to display.

Example:

```
==Initialization==
POS->OPT: Message 1
OPT->POS: Message 2
POS->OPT: Message 3

==Task==
POS->POS: Message 4
POS->OPT: Message 5
OPT->POS: Message 6
POS->OPT: Message 7
OPT->POS: Message 8
```



4.4.5 Notes

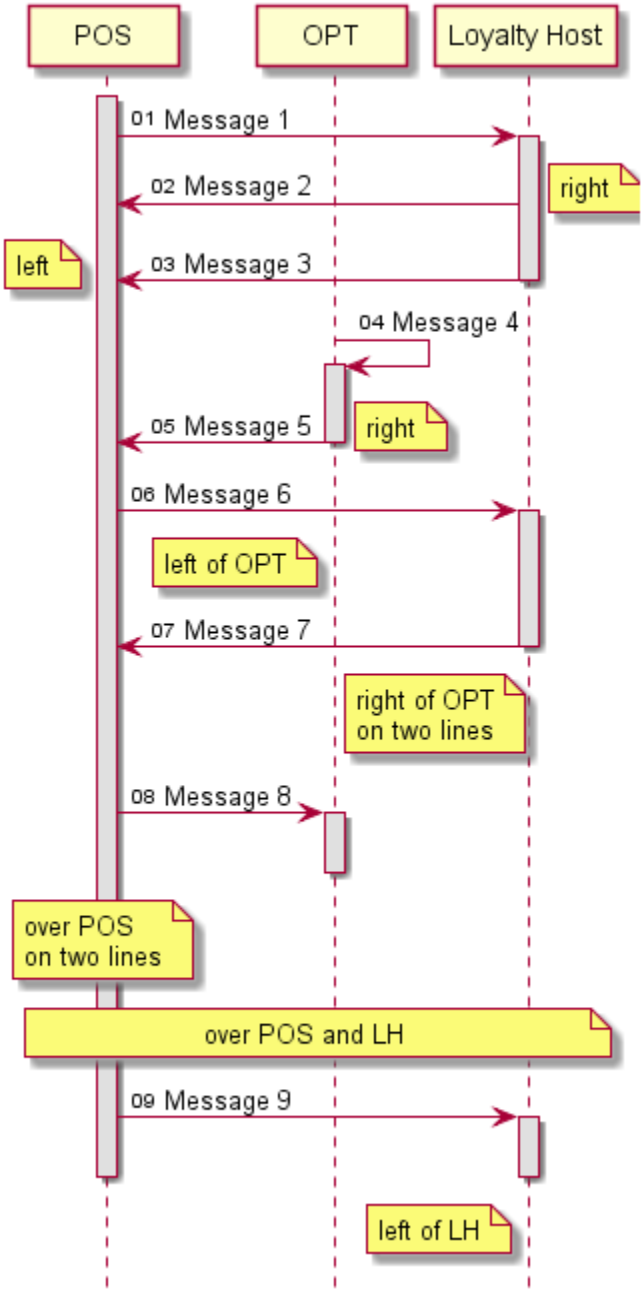
Notes can be added to the sequence diagram using the keyword `note` as outlined in the table below.

Keyword	Horizontal Placement	Vertical Placement
`note left`	To the left of the leftmost participant in the immediately preceding message	On the same vertical line as the immediately preceding message
`note right`	To the right of the rightmost participant in the immediately preceding message	On the same vertical line as the immediately preceding message
`note left of (participant)`	Immediately to the left of the specified participant	On a new vertical line
`note right of (participant)`	Immediately to the right of the specified participant	On a new vertical line
`note over (participant)`	Over the specified participant	On a new vertical line
`note over (participant1, participant2)`	Over the specified participants	On a new vertical line

Text for notes can be split over multiple lines by inserting new lines (\n) into the note text. Alternatively, the keyword `note` may be used, followed by the note text on one or more lines, followed by keyword `end note`. Both methods are illustrated in the example below.

Example:

```
activate POS
POS->LH: Message 1
activate LH
LH->POS: Message 2
note right: right
LH->POS: Message 3
note left:  left
deactivate LH
OPT->OPT: Message 4
activate OPT
OPT->POS: Message 5
note right
    right
end note
deactivate OPT
POS->LH: Message 6
note left of OPT
    left of OPT
end note
activate LH
LH->POS: Message 7
note right of OPT: right of OPT\non two lines
deactivate LH
POS->OPT: Message 8
note over POS
    over POS
    on two lines
end note
note over POS, LH
    over POS and LH
end note
activate OPT
deactivate OPT
POS->LH:  Message 9
note left of LH:  left of LH
activate LH
deactivate LH
deactivate POS
```



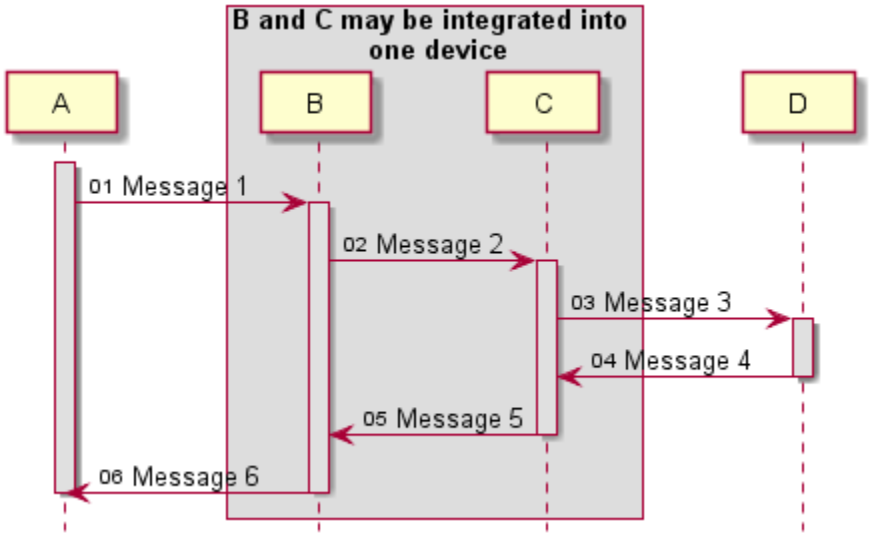
4.4.6 Vertical Boxes

Participants may be grouped together in a vertical box to draw attention to functionality, purpose or architectural considerations. Use the keywords box and end box in the participant declarations to define where the box should be drawn.

Example:

```
participant A as "    A    "
box "B and C may be integrated into \n one device"
  participant B as "    B    "
  participant C as "    C    "
end box
participant D as "    D    "

activate A
A->B: Message 1
activate B
B->C: Message 2
activate C
C->D: Message 3
activate D
D->C: Message 4
deactivate D
C->B: Message 5
deactivate C
B->A: Message 6
deactivate B
deactivate A
```



4.4.7 Splitting Diagrams

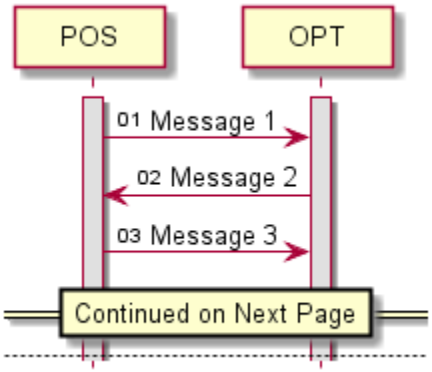
When a sequence diagram is too long to fit on a page without losing readability, split it into one or more graphic files using the keyword newpage. Each additional graphic file generated will have 00x appended (e.g., example.png, example_001.png, example_002.png). Place a divider at the bottom of the first page and at the top of the second page to make it clear that the diagram is split.

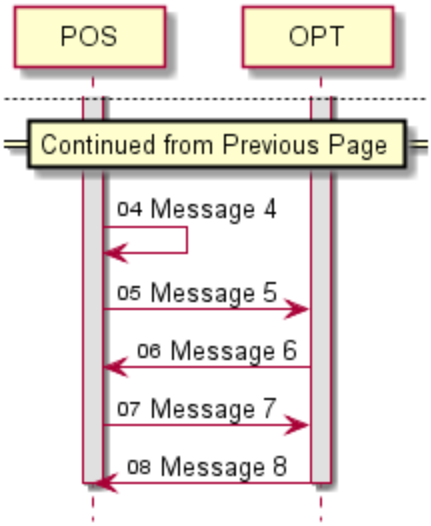
Example:

```
POS->OPT: Message 1
OPT->POS: Message 2
POS->OPT: Message 3

==Continued on Next Page==
newpage
==Continued from Previous Page==

POS->POS: Message 4
POS->OPT: Message 5
OPT->POS: Message 6
POS->OPT: Message 7
OPT->POS: Message 8
```





4.4.8 Comments

Comments can be included in the file by starting the line with a single quote (‘).

Example:

```
‘This is a short comment
```

To put a comment on several lines you can start with /’ and end the quote with ‘/.

Example:

```
/' This is how you can
span multiple lines
of comments
'/
```

4.4.9 Whitespace

PlantUML ignores blank lines and whitespace at the beginning of a line. Liberal use of whitespace to offset logical blocks in the text file can enhance readability.

For example, this:

```
POS->OPT: Request Message
group If POS configuration is xxx
OPT->POS: Response Message:1
group If Site configuration is aaa
OPT->POS: Response Message:2
else Otherwise
OPT->POS: Response Message:3
end
end
```

Is equivalent to:

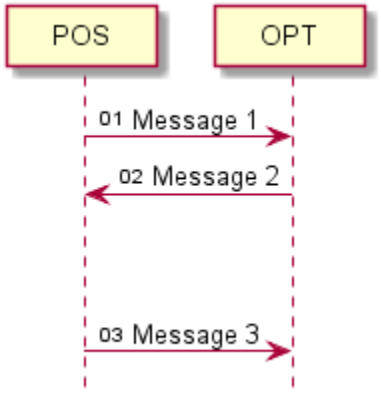
```
POS->OPT: Request Message
group If POS configuration is xxx
    OPT->POS: Response Message:1
    group If Site configuration is aaa
        OPT->POS: Response Message:2
    else Otherwise
        OPT->POS: Response Message:3
    end
end
```

However, the second iteration with additional whitespace is much easier to follow, debug, and modify if future changes are required.

If whitespace between messages is required in the diagram, use one or more "|||".

Example:

```
POS->OPT: Message 1
OPT->POS: Message 2
|||
|||
POS->OPT: Message 3
```



4.4.10 Diagram Titles

Diagram titles should not be used. Once the sequence diagram is included in a process flow document, it will be labeled with a figure caption. Instead, use a descriptive name for the file name.