My Orange
Balance principles

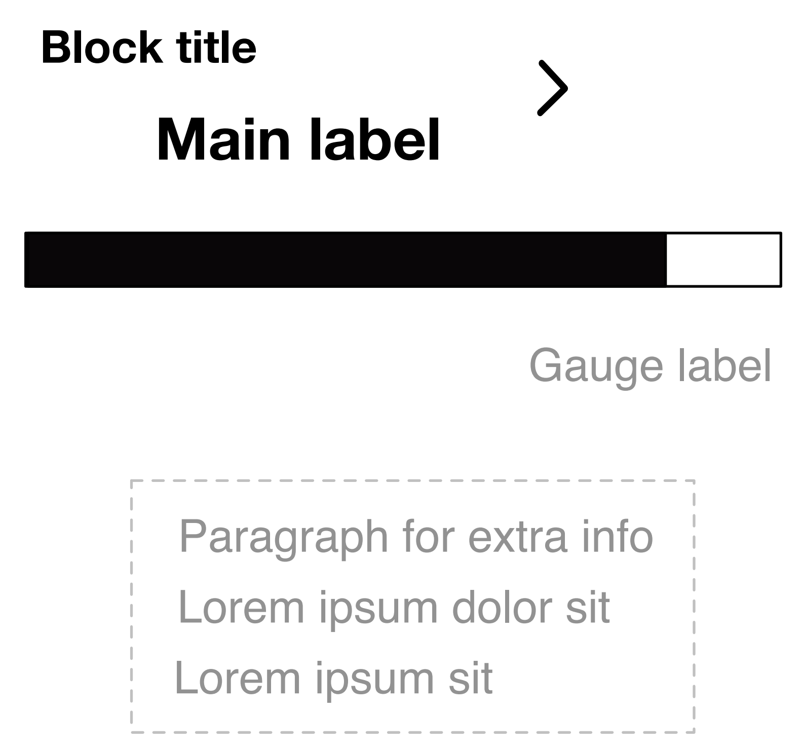
Date: 12 October 2012

Platform: generic
Country: generic
Document type: design pattern specification

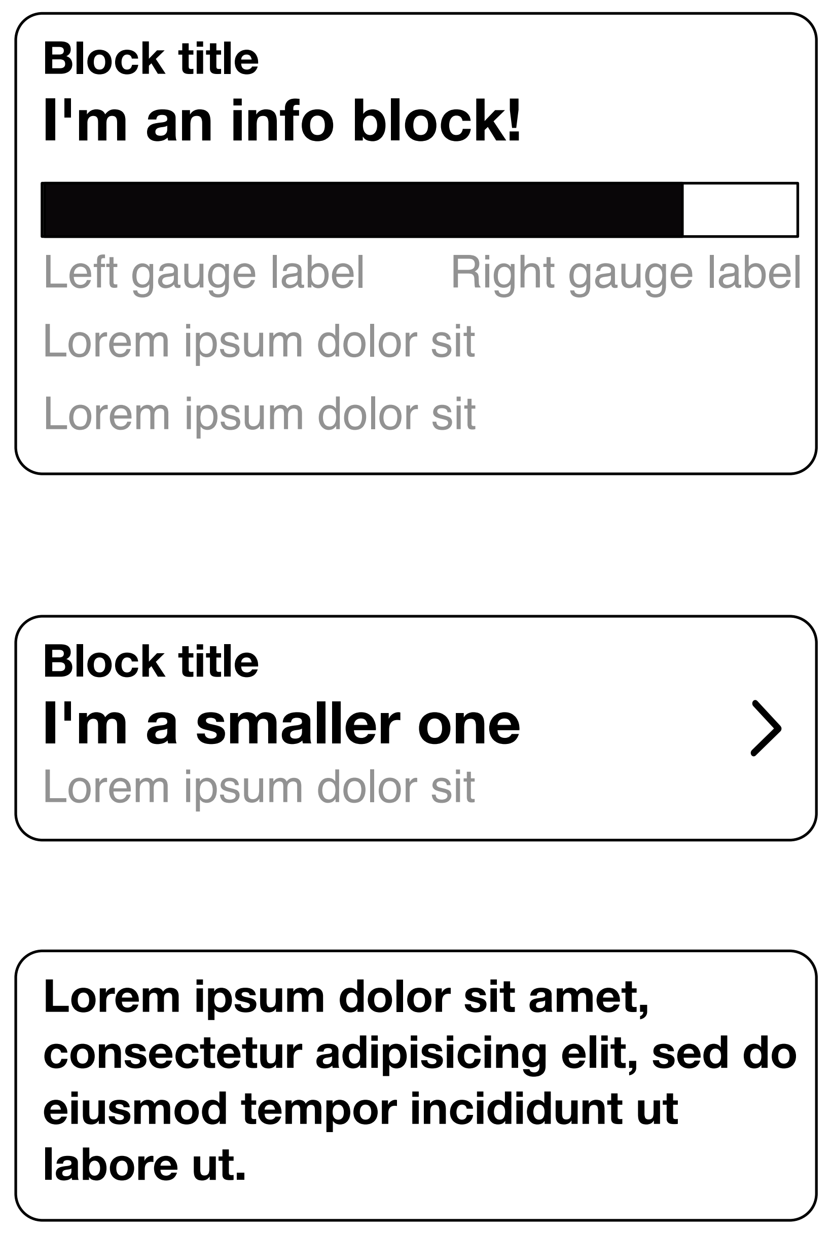
Author: Fabrice Dubois (D&U) – fabrice.m.dubois@orange.com

1 Pattern

Balance view is made of information blocks that are based on a simple UI pattern.



Basic elements.



Examples of information blocks built with the elements.

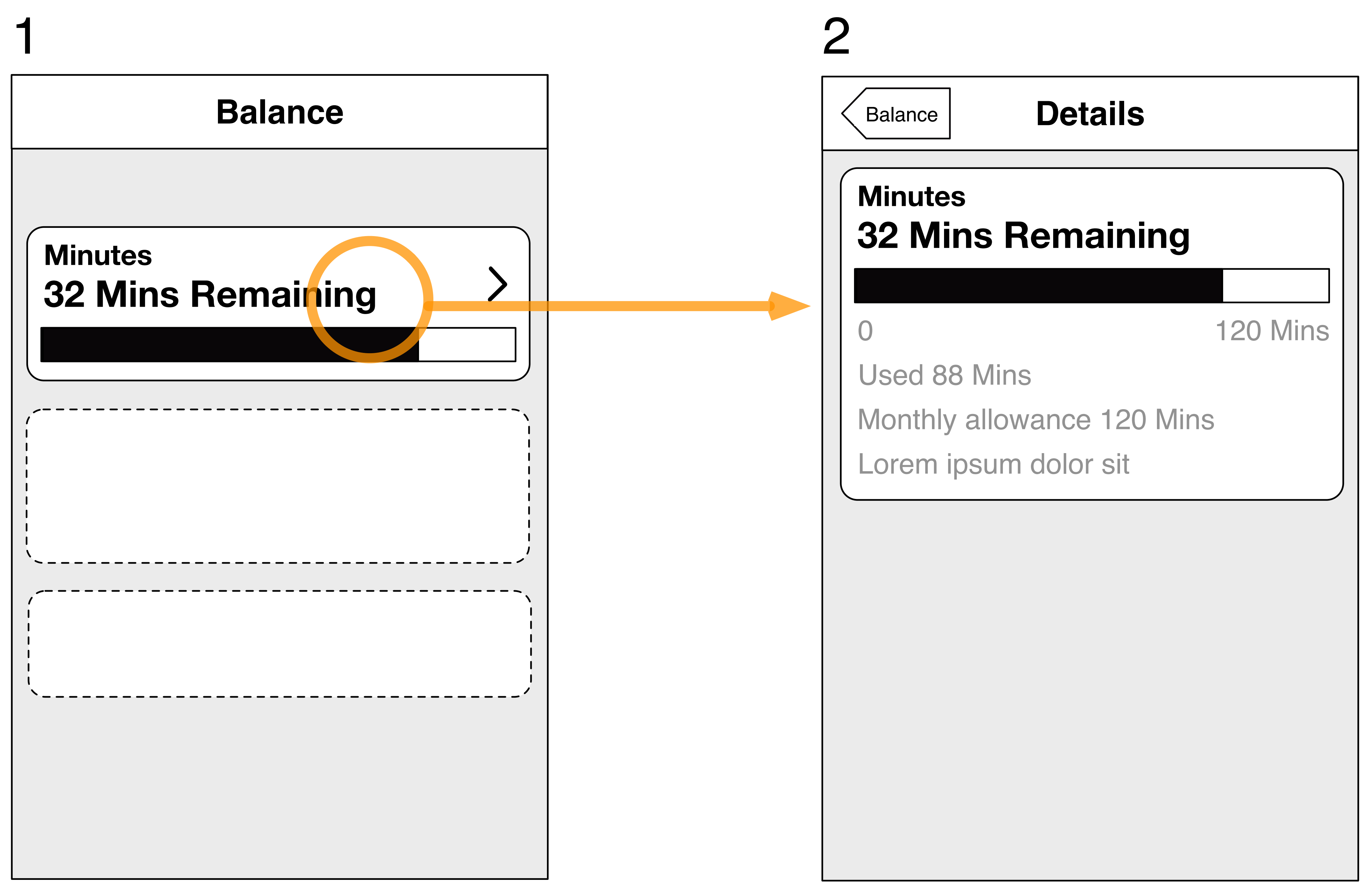
2 Compact views and details views

The pattern allows us to build both compact and details views of each block. User selects compact view to enter the details view. The figure to the right illustrates how it would work on an iPhone interface.

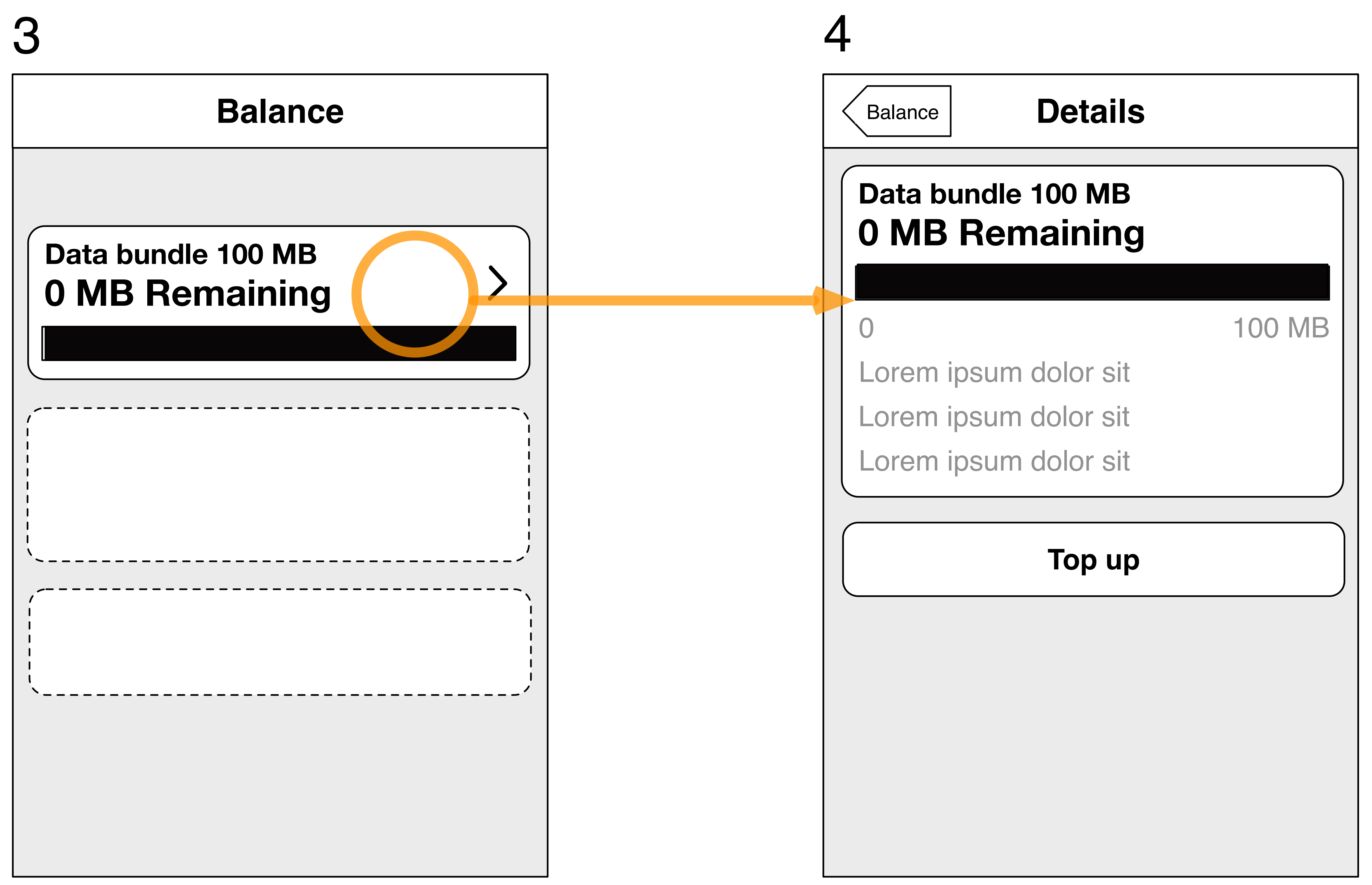
The **details view** can be as rich in information as required, because it will be shown on demand only.

And the **compact view** can afford to be as simple as desired, because it can 'delegate' details to a details view anyway.

NOTE: when a block is really simple it may not even need a detailed view.



Together with the detailed view can be revealed extra controls that make sense in the context.



An example of contextual action for this data bundle block.

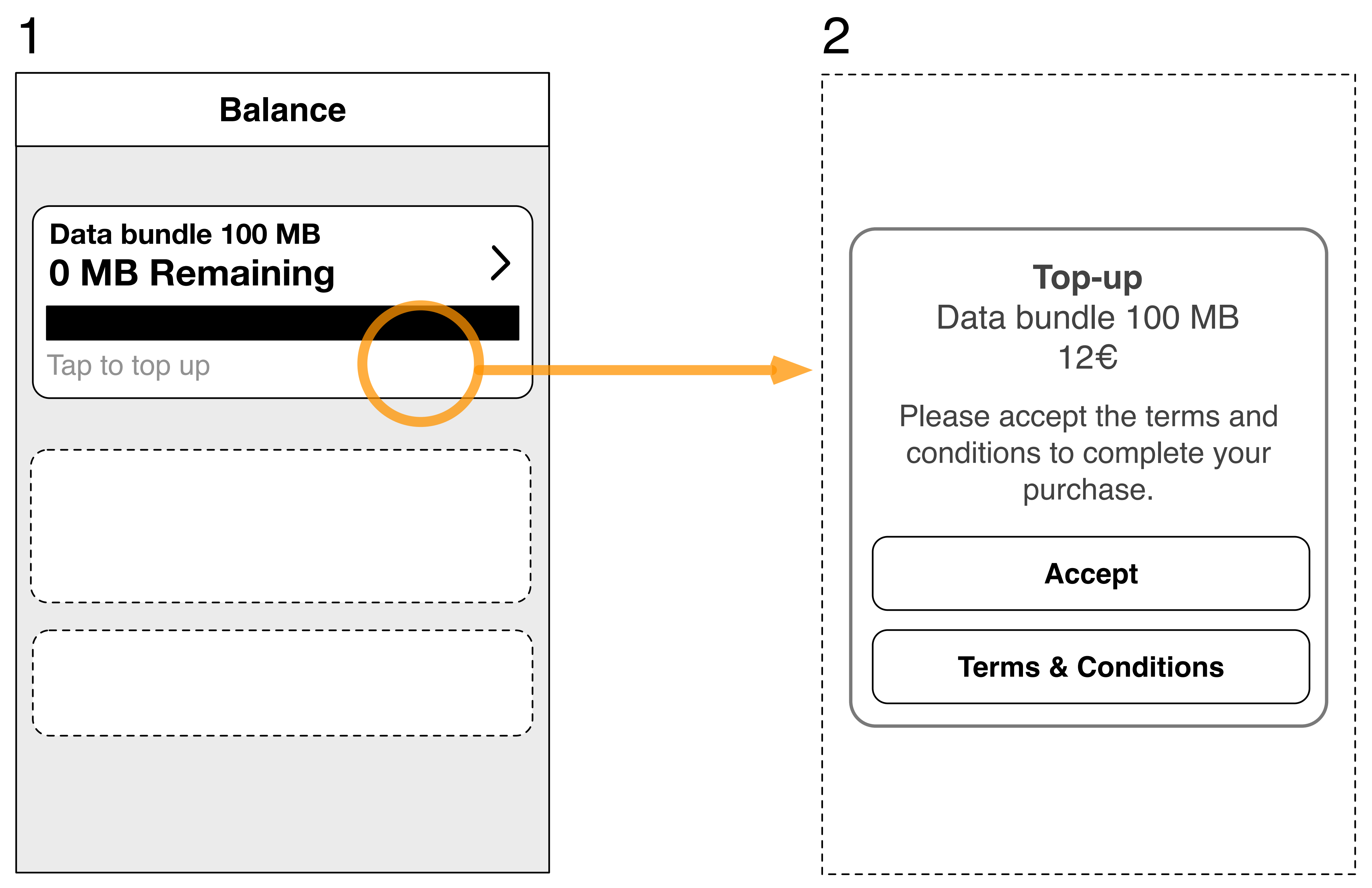
3 Interactive blocks

Blocks should support touch interactivity when relevant. The chevron symbol should be used to indicate that the blocks supports it.

However, to keep it simple, a single action should be bound to a block (making the whole block a touch area).

The previous page shows how the touch event can be used to reveal a details view.

Here we show that the touch event could also be used to trigger any other action (for blocks that have no details view to reveal).

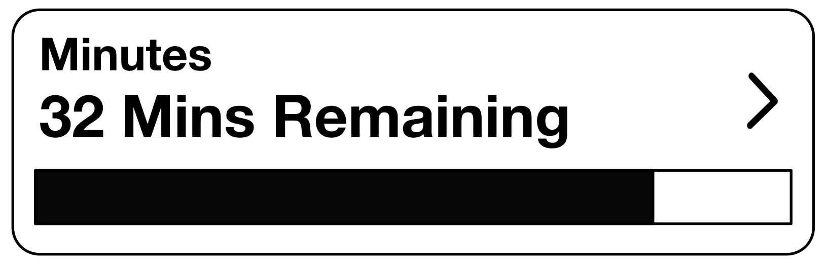


4 Flexibility

This shows how the flexibility of the pattern can be used to balance the right amount of details/information between the details and the compact views, and decide whether there is a need for a details view at all.

We illustrate this with different options for the compact view of the same "Minutes" block.

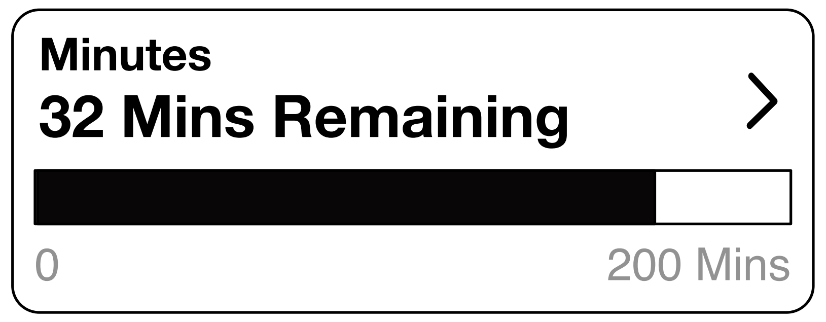
1



Minimalist

Gauges are shown without bounds. Bounds shown only in details view.

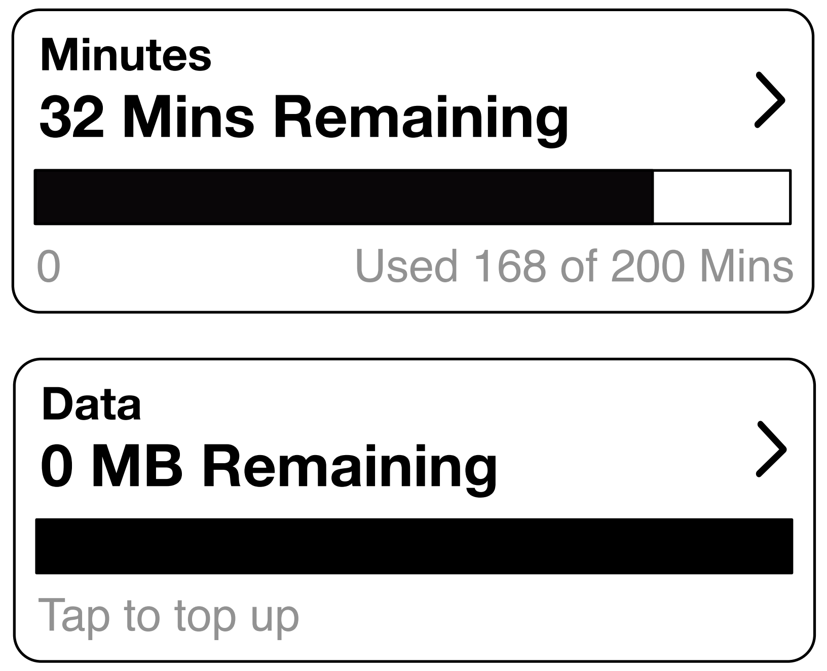
2



Basic bounds (**recommended**)

Gauges show their end values on the compact view.

3

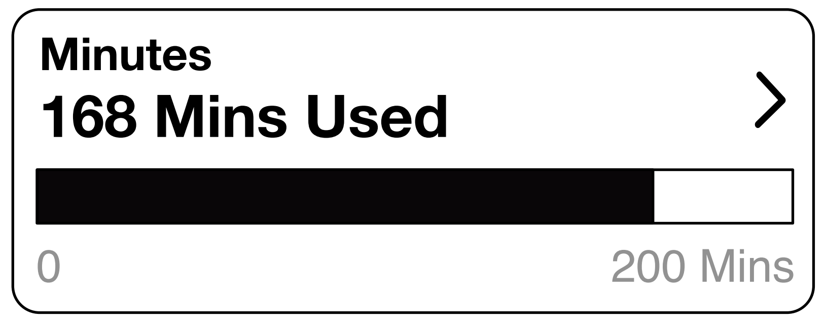


Verbose bound

1 bound label is used as a statement, not just to display the bound value.

NOTE: making both bounds verbose is not recommended as there is a risk of overlapping that is hard to control.
Also, the verbose bound layout begins to make the details view less useful, which is not ideal when we need it anyway.

4



Inverse

This example shows how the pattern can be used to emphasise Used amounts rather than Remaining amounts.

5 Gauges

To keep the design simple and prevent confusion, the gauge component, when used, should always behave the same way throughout the application. There are a couple of parameters to decide to define the behaviour:

Grow or shrink

The gauge can either:

- grow as the user consumes (this is what it does in the examples in the present document)

- shrink as the user consumes

Anchor point

The gauge can be anchored either:

- to the left hand side of the block (that's the case in the present document)

- to the right hand side of the block

Toggle or not

You may either :

- let the user choose whether the balance should put emphasis on remaining amounts rather than used amounts, or

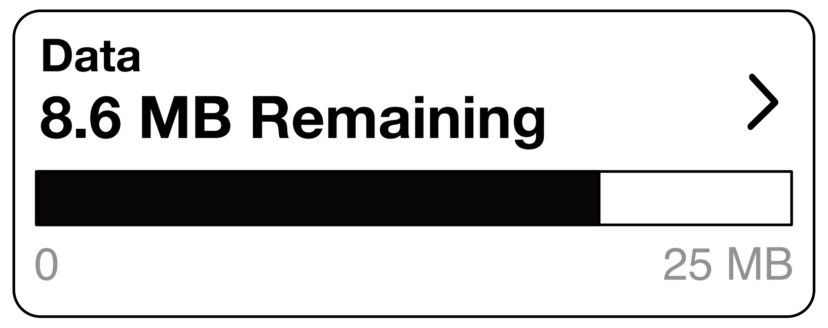
- decide and impose a fixed emphasis.

Most examples in the present document emphasise the remaining amounts and understate the used ones.

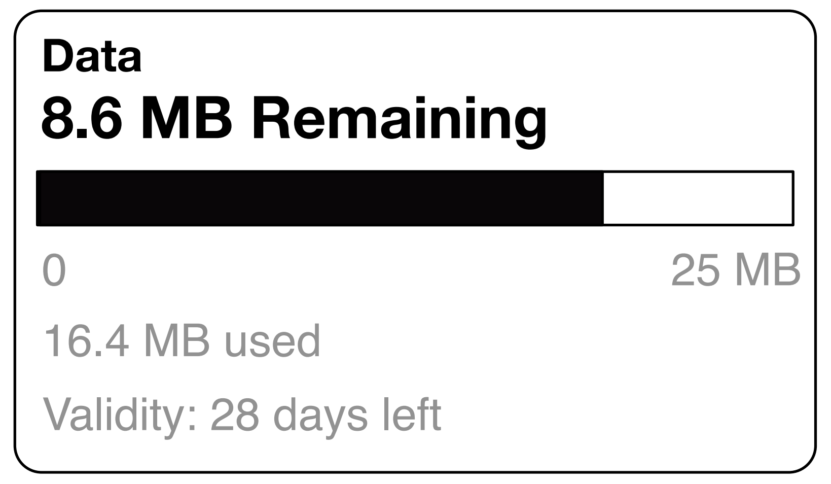
Recommendation if you're going to offer a toggle control to users:

- Keep the toggle control understated, because it is not something people are likely to use all the time. Do not let it clutter the main UI. You may put in the app's settings.

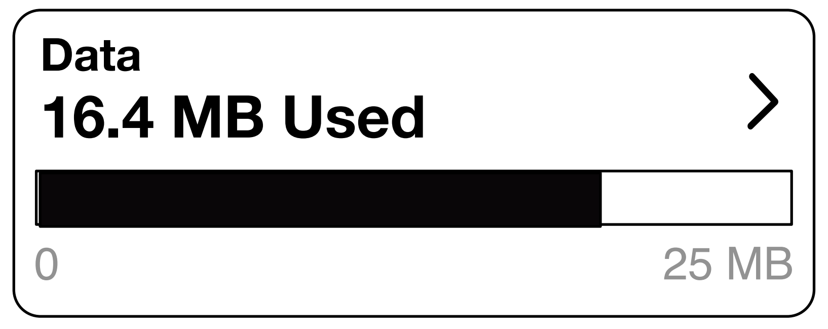
- The toggle should only swap the Remaining and Used textual statements. It is simpler if the gauge component itself is NOT affected by the toggle.



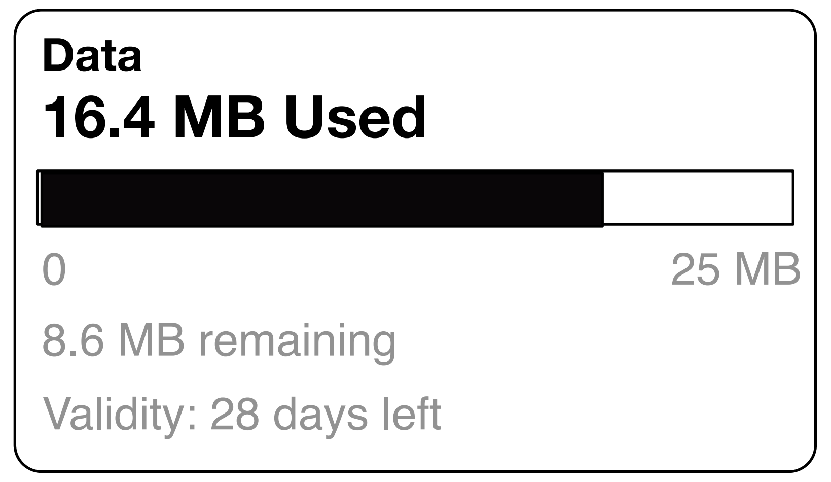
Emphasis on remaining amounts
(compact view)



Emphasis on remaining amounts
(detailed view)



Emphasis on used amounts
(compact view)



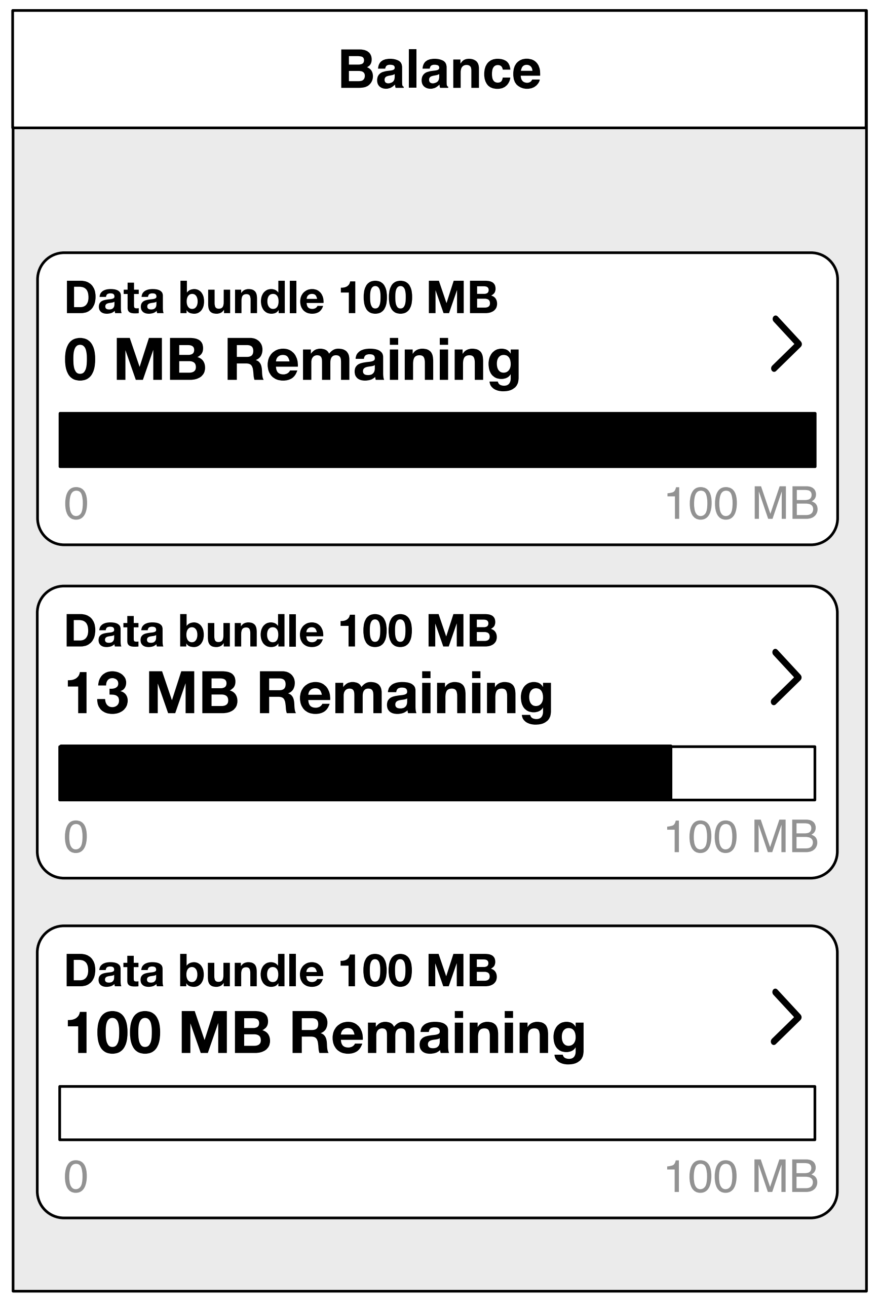
Emphasis on used amounts
(detailed view)

6 Order

The app itself does not associate any particular meaning to the order of blocks: it just displays them in the order it is told to display them.

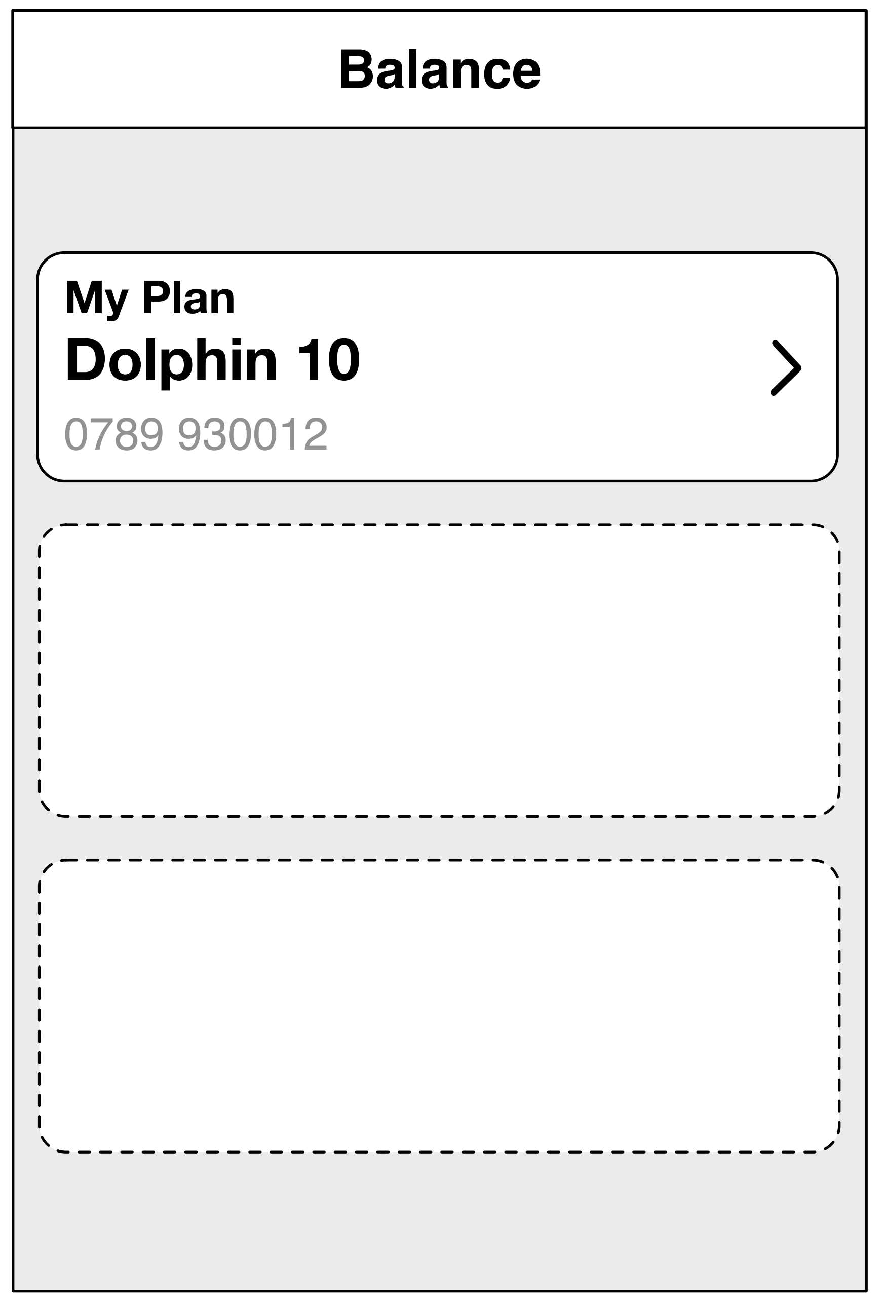
If the order has some importance, then it must be fixed at the backend level.

1



Example of meaning given to the order: the data bundles purchased by the user appear with the most recently purchased bundle at the bottom of the list. Meaning is: "bundles are consumed from top to bottom".

2



Example where the backend decides that the plan reminder block should be placed at the top of the list.