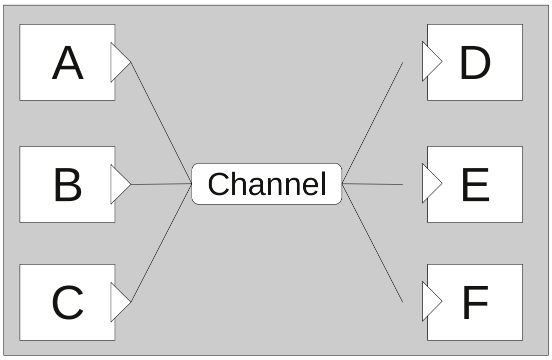
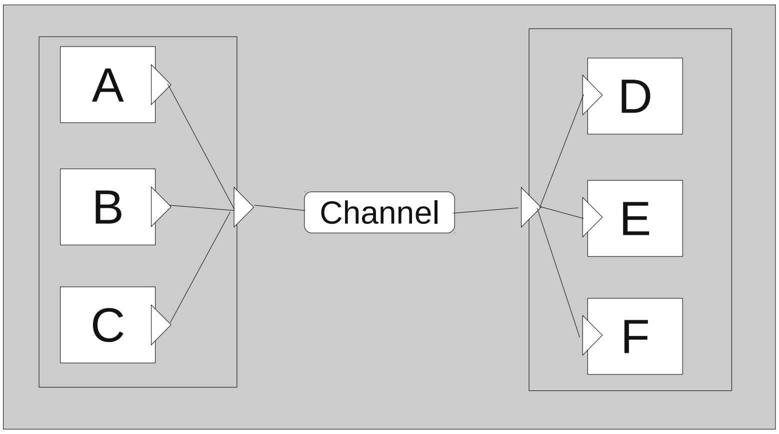
Simple Case – showing overall intent



Question: How can we *compose* the same result by building up an application from a number of composites, rather than building just one composite?

Easy scenario – all publishers grouped, and consumers grouped



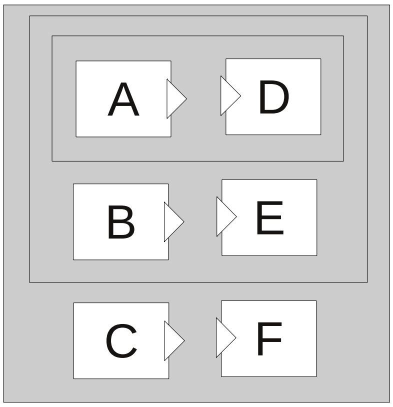
There are several not-so easy scenarios explored in the following pages, showing what they look like with the current working draft, and then alternate ways of thinking about and presenting them:

- Only promote from channels (discarded – you'll see why)
- “direct wires” - no “channels” necessary, although they can still be used.
- Promote channels
- Channels as just another aspect of “component type”, but tying producer and consumer

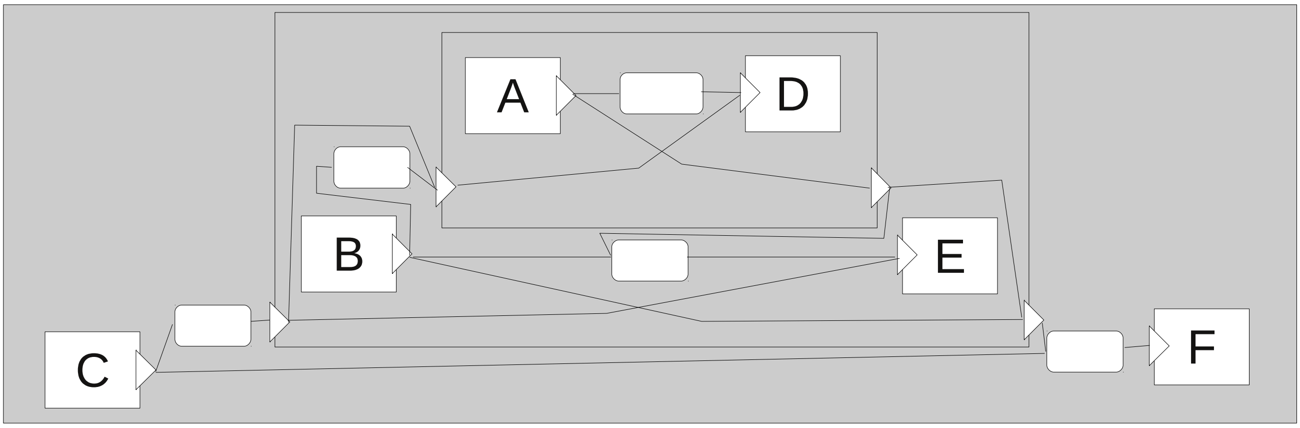
Pathological Case #1: (PC #1)

Composite containing A & D, contained inside a composite that holds B & E, which is contained inside of a composite holding C & F. Picture to the right shows without wiring.

Question: How can I do the wiring to achieve the same result as the simple case, where a message from A goes to D, E, & F, and likewise for B & C?

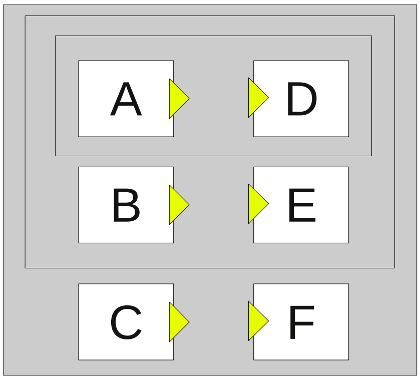


PC #1, following current WD, but avoiding global channels and bindings

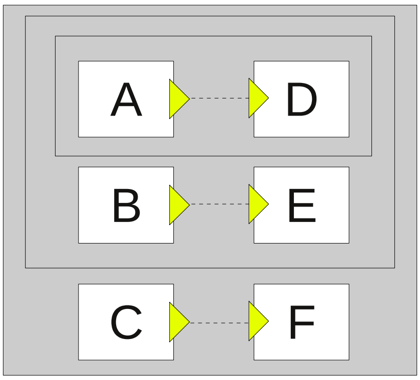


Problems: So many wires – confusing. At deployment, may be impossible to infer that a single channel is appropriate → may create at least three channels, and likewise increase the costs of listening and sending. Up-side – different policies and bindings could be employed, also completely composable.

PC #1, with global channel or concrete bindings



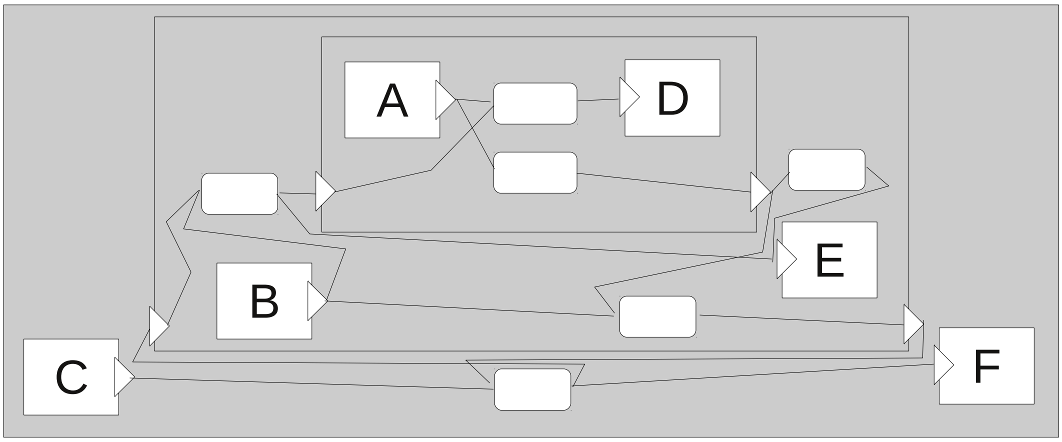
Or, if graphical editor notices @source & @target (or bindings) are the same:



Much simpler! But – little or no visibility to what is actually happening. Easy to make binding and/or @source/@target value mistakes. New components can come along a use the same global channel or binding destination without noticing, yielding surprising results. Problematic composability. What can we do to improve this?

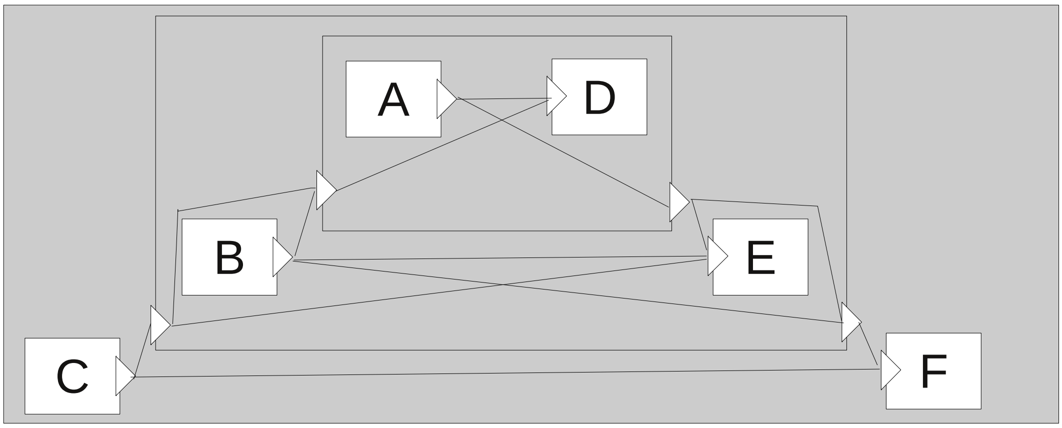
Challenge – How to Simplify But Allow Composition?

PC #1: Option: Promote only from channels



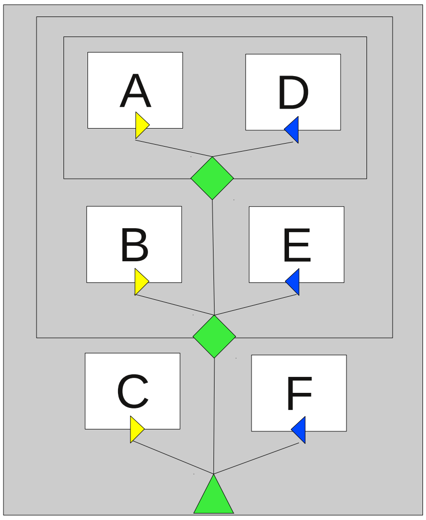
Thought this would be simpler, but it isn't. To follow intuition that messages flow along wires, in order to avoid duplicates, requires a channel for internally bound messages, and one for externally bound messages. It might be possible to simplify the above, but I struggled for some time. Discarding approach.

PC #1 option: “direct” wires, implied channels



Simplifies: Still cluttered, but less so. Fewer “wires”, if only because you can drop the ones that come “from” a channel. Still difficult to see that a single JMS Topic might be possible at deployment. Note that channels may still be a useful graphical shorthand for M-to-N mappings, but only that.

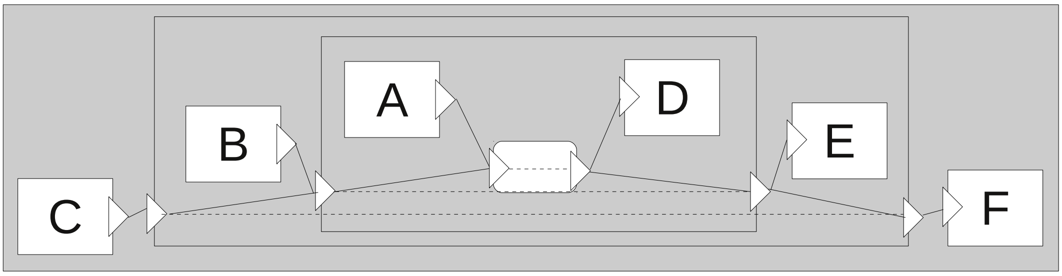
PC #1 option: Promote “channels” (a.k.a “the fir tree”)



Simplifies: Dramatically less cluttered. Easy to see that a single JMS Topic is possible at deployment (assuming no changines in binding/policy). Downsides: The common case at the individual component level – either producing (yellow) or consuming (blue) might appear to be second-fiddle to assumption of bi-directional flow by declaring of channels everywhere. For more complicated pictures (channel bridging, additional components), presentation could lead to lots of crossed wires.

PC #1: Option: Channels as components, coupled Producer & Consumer

- Linking producers and consumers in component type
- Direct wiring from component to component
- Channels just a special characteristic of a ComponentType, where there's some indication of coupling between consumers and producers.

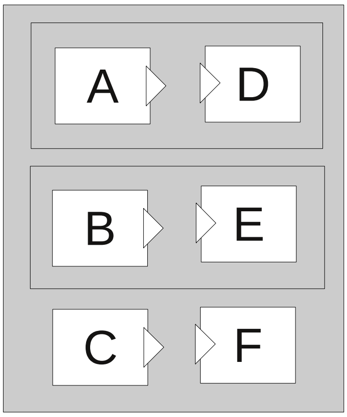


The above is conceptually similar to the “fir-tree” from the preceding page, in that there's some sort of explicitly established relationship between producers and consumers. However, it emphasizes the common case, where producers and consumers are disconnected, while allowing for the scenario where they are connected. It assumes the the composer of B-E composite, knows from the information in the componentType for composite A-D, that events delivered to the consumer on composite A-D will appear at the producer composite A-D, and likewise the composer of C-F will have to know about the relationship of the consumer and producer of composite B-E. Graphically, this is represented by a line across the component coupling producer and consumer.

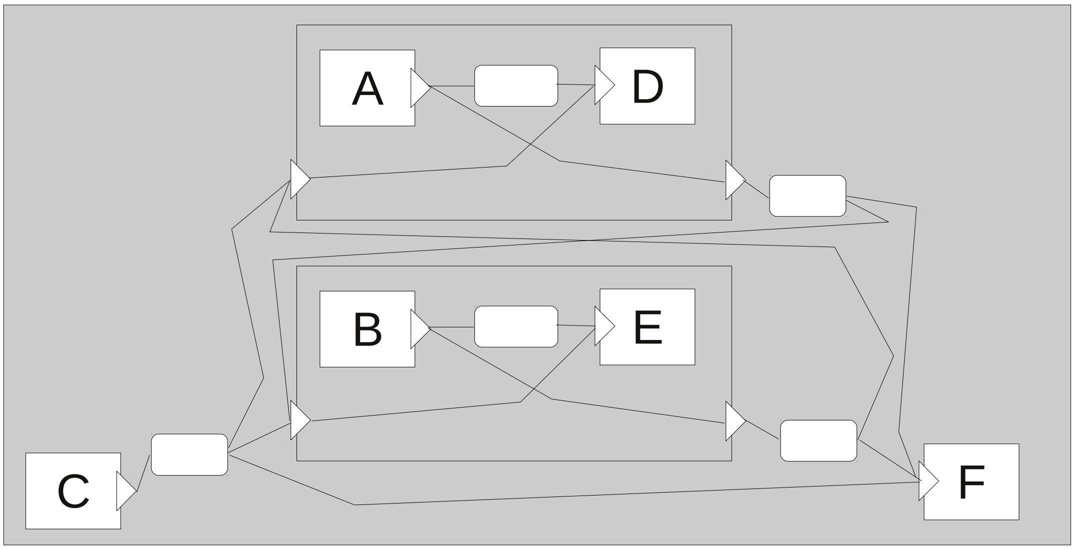
Pathological Case #2: (PC #2)

Composite containing A & D, another composite containing B & E, both of which are contained inside of a composite holding C & F. Picture below shows without wiring.

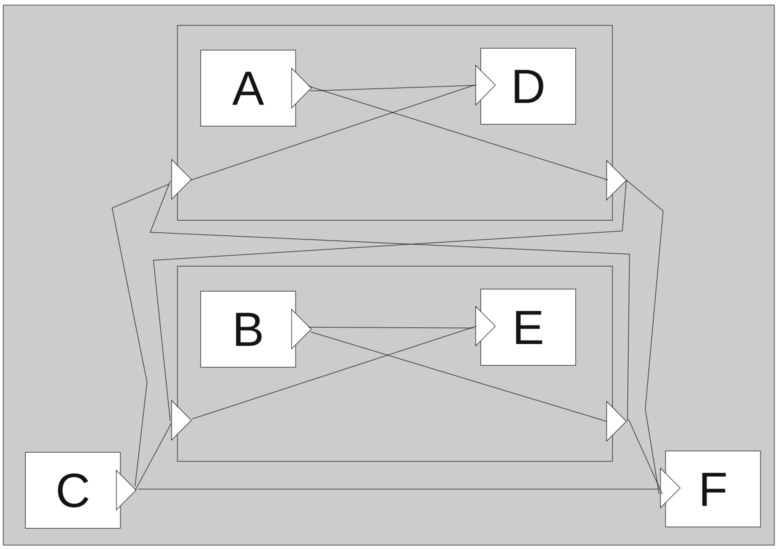
Question: How can I do the wiring to achieve the same result as the simple case, where a message from A goes to D, E, & F, and likewise for B & C?



PC #2, per WD: avoiding global channels and bindings

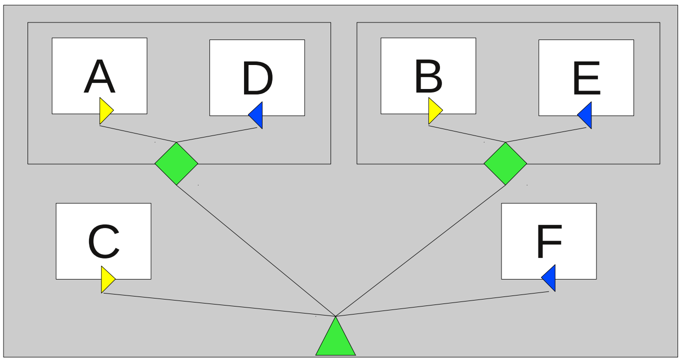


PC#2 option, using direct wires

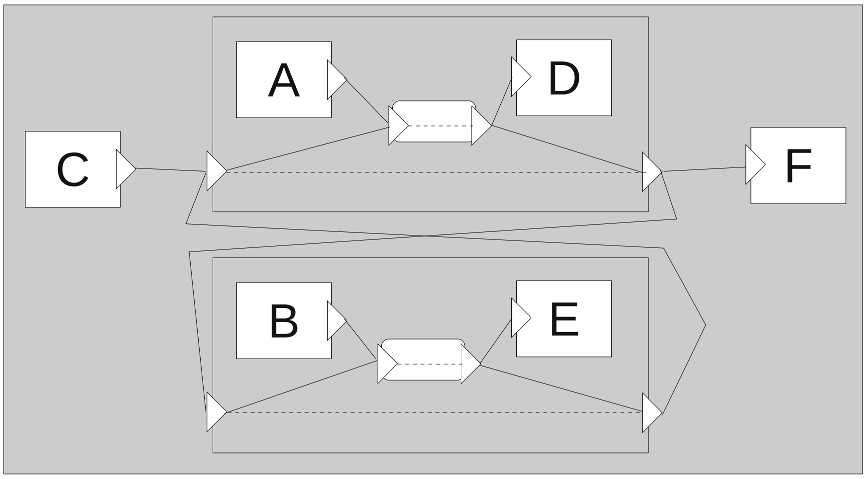


Eliminates a few wires.

PC#2 option: Promote “channels” (a.k.a avocado tree)



PC #2, Using channels as components, producer & consumer tied.

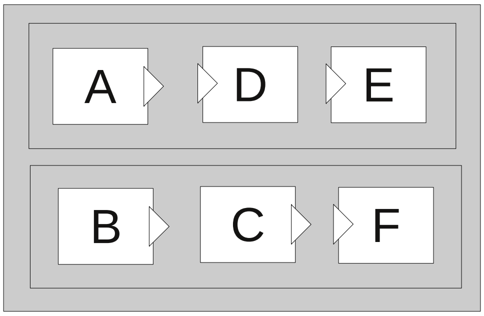


Note a subtlety above. It is likely that this *must* map to a single transport destination. A message sent from A leaves composite A-D, feeds into composite B-E, but then leaves composite B-E headed back to A-D. If these are unique “channels”/destinations, then some logic is needed to avoid resending to the original channel. Or the designer of A-D & B-E needs to provide two channels, one for internal use, one for external use

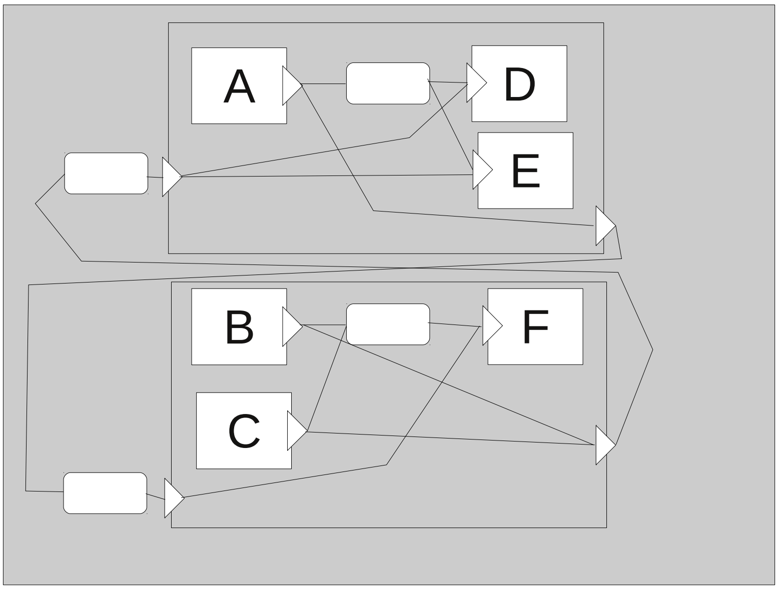
Pathological Case #3: (PC #3)

Composite containing A, D, & E, another composite containing B, C, & , both of which are contained inside of a composite wiring the two together. Picture to the right shows without wiring.

Question: How can I do the wiring to achieve the same result as the simple case, where a message from A goes to D, E, & F, and likewise for B & C?



PC #3, per WD: avoiding global channels and bindings



PC#3: Option: promote “channels” (a.k.a spreading ivy)