Contain Flows

Standard Link Mechanism

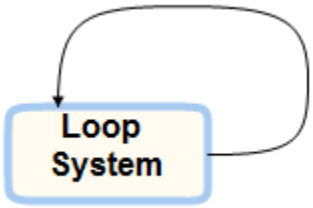
The most visible characteristic of flows are the **links** produced in diagrams. The principles applicable inside a filter are:

- A link appears between two objects *A* and *B* in the Internal Block Diagram if a flow *F* is produced by *A* or one of its descendants and consumed by *B* or one of its descendants.
- A link appears between an object *A* and a port if a flow is produced (resp. consumed) in *A* or one of its descendants and consumed (resp. produced) outside the focus object.
- A link appears between an object *A* and a port if a flow is produced (resp. consumed) in *A* or one of its descendants and not consumed (resp. produced) in the project. We call this flow unallocated or unlinked.

Sometime it is possible to see a link appearing as a loop that is produced and consumed by the same object.

- A link appears as a loop on an object *A* if it is produced (resp. consumed) by *A* and consumed (resp. produced) by *A* or one of its descendants.

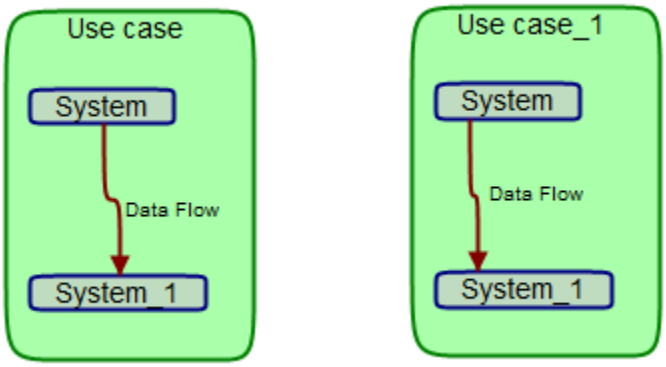
The **Contain Flows** feature is aimed at creating exceptions to these principles.



When Do I Need To Not See a Link?

There are two main circumstances in which you do not want to see links.

1. The first reason is to prevent from being submerged with unallocated flows in the Internal Block Diagram. Indeed, when you have a big project with a lot of unallocated flows, you may not want to see them at the top level if you are sure they will be consumed (resp. produced) at the same level as their producer (resp. consumer). For example, the internal flows of ECUs (Electronic Control Units) are absolutely not relevant at the top level.
2. The second case is that you want to have in your project several instances of a system. For example, if you want to model several use cases for the system you are designing, you could want to model it like in the picture below. There are a lot of flows inside the object *Syst em*. Each flow is produced inside *Use case* and consumed inside *Use case_1*; thus they should all be displayed and inundate the diagram. Happily, *Use case* contains (i.e. hides) each flow.

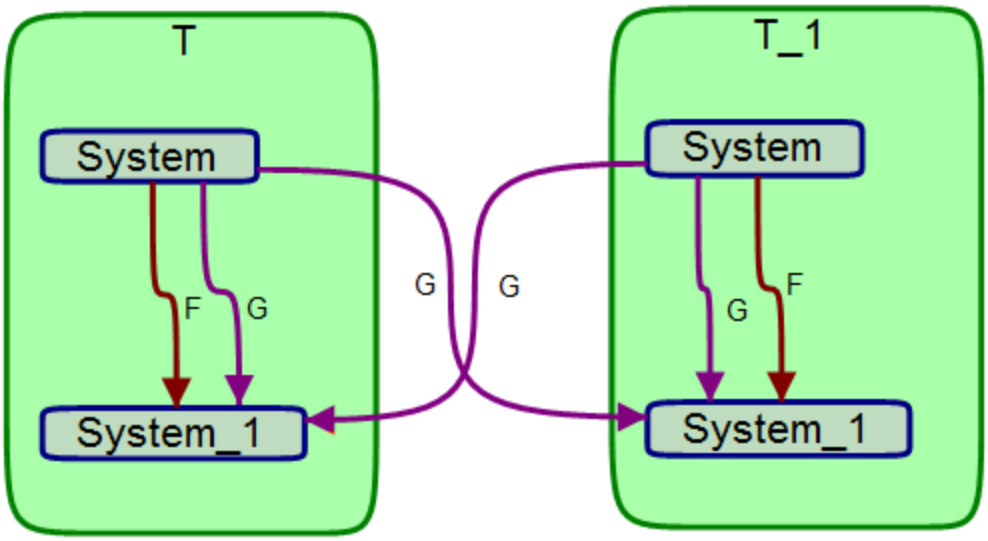


How Does Contain Flows Work?

The principle is quite simple. For a given type T, we can set flows of type F to be contained. In this cases, an object of type T will not display incoming or outgoing links of type F. It is possible to contain several flow types at the same time. All flow types that are indirect children of the abstracting type can be contained.

⚠ Contain Flows does not work for first-level links (i.e. flow instances that are direct children of the abstracting object).

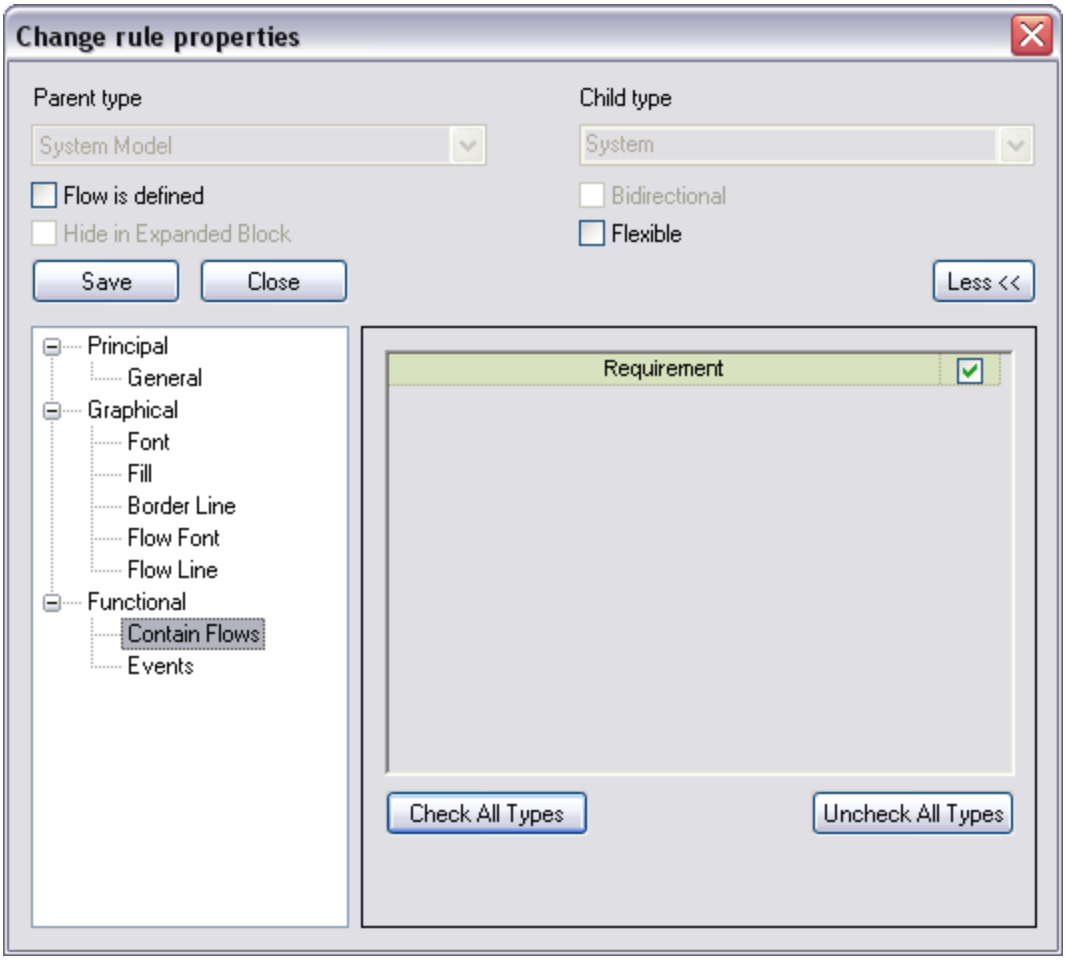
In the example below, T contains flows of type F but not of type G.



Activating Flow Filtering

In the Type Properties, the **Contain Flows** section is dedicated to defining which types of flows are contained by the current type. The proposed types are the relevant ones (the flow types that are descendants of the containing type). The types with **not checked** check-boxes are contained. The **Check All Types** and **Uncheck All Types** options can be used to help the checking/unchecking.

⚠ Click on **Save** to make the change in the Contain Flows settings effective.



It is also possible to activate the Contain Flows mechanism for a particular projection.