Connectors

Connectors are used to branch or reconnect the process flow. Two connectors are available in EPCs for modeling: **AND** and **XOR**.

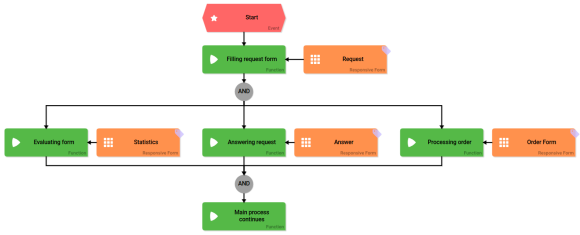
Further information and case examples regarding the use of each connector can be found at AND Branching and XOR Branching.

On this Page:

Related Pages:

- Modeling EPCs
  - Configuring EPC Branching

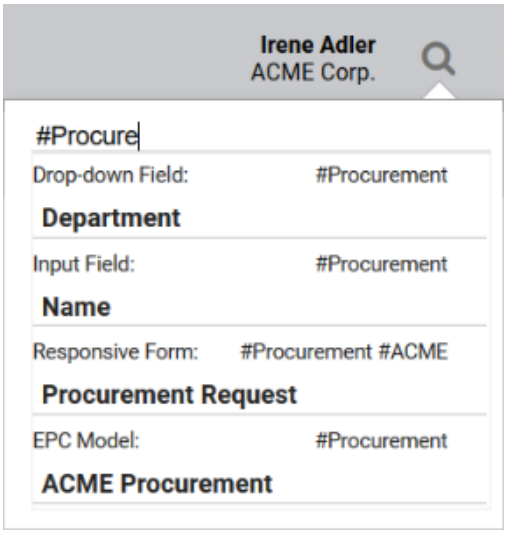
The AND Connector

Element	AND
Function	During modeling, the **AND** Connector is used to document that multiple process branches are run through simultaneously. This means: All branches between the AND brackets have to be completed before the process can continue in the main branch.
Connection	Predecessor: **Function** Successor: **Function, Event**
Example	A business process intends that completion of a reuqest form triggers three different process branches: The form is evaluated statistically, the buyer receives an answer and the order is processed. The first AND Connector opens the parallel execution: The three process chains Evaluating form, Answering request and Processing order are started simultaneously and run independently from each other. The closing AND defines the end of the parallel execution. The process continues with the next step of the main process branch. 

Configuration Options

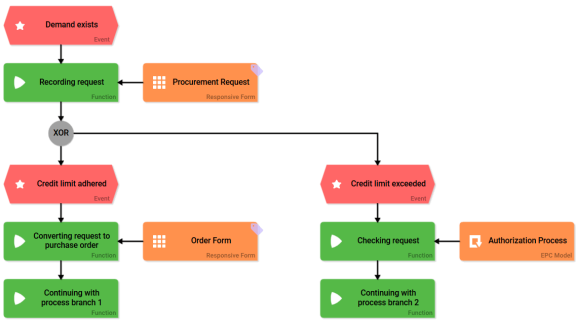
The form element may be edited via the **Edit Sidebar**.

Name AND	**Name** Field name shown in the form.
ID rp02727f33-b7cf-40ef-886e-b4da441295ff	**ID ( )** The read-only field contains the model ID of the element. Designers can use the ID for example in the search to link the element in other models.

	**Description** A commentary field for the modeler. Its content is invisible to users and its sole purpose is internal documentation.
	**Hashtags** Possibility to insert your own key terms . The leading hashtag sign **# (hash)** will be inserted automatically. Use the spacebar to insert multiple hashtags:  One hashtag may be issued for multiple elements: 

The XOR Connector

Element	
Meaning	The connector XOR (= Either-Or) indicates that one of several conditions must be fulfilled. This means that after the XOR, exactly one of several possible process strings is processed.

Connection	Predecessor: **Function** Successor: **Event**
Example	A business process intends that a Procurement Request can be placed. After the action Recording request two possible events can occur: **Either** the Credit limit adhered **or** the Credit limit exceeded. Depending on which event occured, a different process string is started. The constraints, that need to be met for either process sring are defined in the element Event. 

Configuration Options

Edit the element via the **Edit Sidebar**.

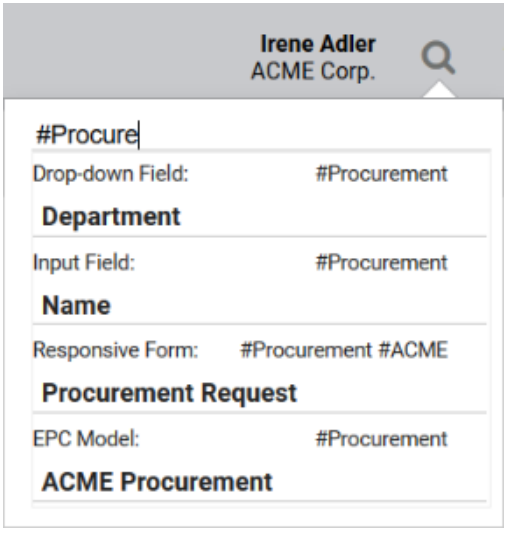
Name XOR	**Name** Name of the element within modeling.
ID rp02727f33-b7cf-40ef-886e-b4da441295ff	**ID ( )** The read-only field contains the model ID of the element. Designers can use the ID for example in the search to link the element in other models.
Description	**Description** A commentary field for the modeler. Its content is invisible to users and its sole purpose is internal documentation.

Hashtags

Possibility to insert your own key terms . The leading hashtag sign **# (hash)** will be inserted automatically.

Use the spacebar to insert multiple hashtags:

One hashtag may be issued for multiple elements:



Branching and Constraints

The AND Connector is tied to a single constraint: All process branches within the AND bracket have to be executed. The main process can only continue once all process branches between the opening and closing AND Connector are processed. Therefore no further constraints need to be defined when using the AND Connector.

The XOR Connector enables issuing multiple constraints. Therefore the XOR Connector is always followed by an Event in which the corresponding constraints for the varying process strings need to be defined.

In order to prevent an EPC from being ineffective always verify **event** and **counterevent** when designating constraints.

Example: A form offers the answers **YES** a nd **NO** . In the process, yes-answers shall follow a different path than no-answers. Therefore it has to be verified which of both paths needs to be followed after the form has been saved. Instead of checking if **YES** **or** **NO** was specified, check if **YES** **or** **NOT YES** was entered. This enables you to also pick up cases where neither answer was specified.

The counterevent to YES is not NO, but NOT YES!