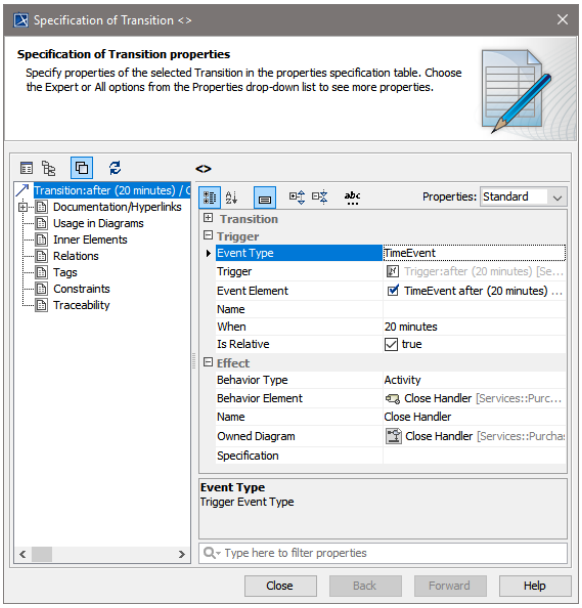
Transitions

States are connected by directed **transitions**. Each transition has a **trigger** and an **effect** compartment in the transition specification - only exception: the creating transition.

On this Page:

Figure: Defining Trigger and Effect of a Transition



The creating transition starts from the initial state. It is implicitly triggered by the creation of the object. For all other, non-initial states, the trigger is either a **time** or a **signal** event. The effect of each transition is twofold: First, an event handler is executed, which must always be a method of the <<PersistentState>> class. After executing the event handler successfully, the object goes into the target state of the transition. If an unhandled error occurs, the object stays in the transition's source state.

Transition Rules

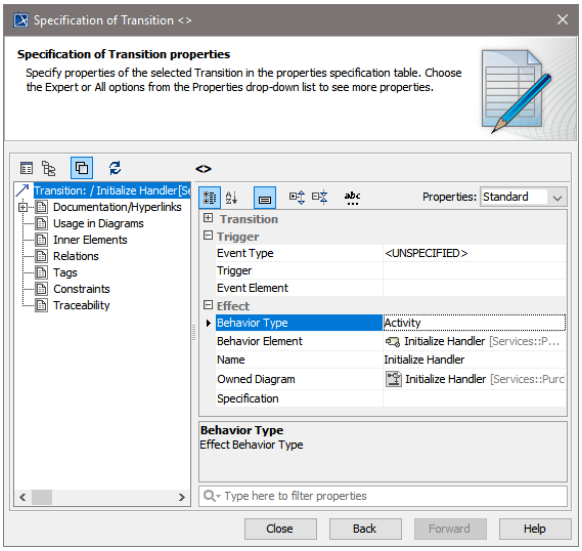
The xUML Runtime that executes and controls transitions complies with the following rules:

Rule	Description
Within one object transition, execution is serialized.	Only one transition of a given object can be in execution at any time. This is required to avoid conflicts of multiple transition activities manipulating the same object.
Objects are treated concurrently.	Transitions on different objects can be executed simultaneously.
A transition takes time to execute.	In state machine diagrams without parallelism, this is not an issue. For models that use parallel states, the serialization of transitions may pose a problem: use Do Activities in these cases.
New transitions are processed only after completion of the previous transition (Run-To-Completion).	Once initiated, a transition must complete before the same object can process another transition. It is in the modeler's responsibility to ensure that the procedure will complete ("Run-To-Completion" semantic).
Transitions are consistent.	When a transition completes, it must leave the system consistent.

Initial Transition

The entry point of a newly persisted object into its life-cycle is defined by the transition starting at the initial state (filled circle). This is triggered by creating a persistent state object as depicted in the activity diagram **Create Purchase Order** (see further below). Each state machine diagram must have exactly one initial state, which must have exactly one outgoing transition.

Figure: Defining the Effect of Creating a Persistent State Object in MagicDraw

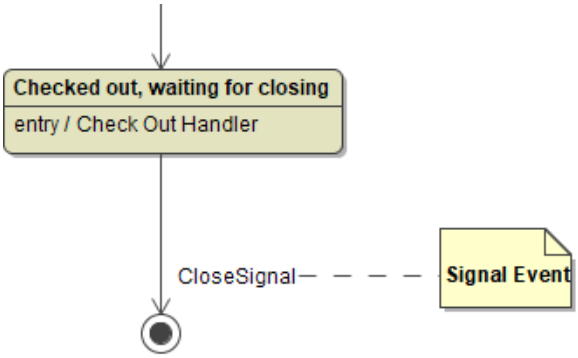


Unlike other transitions of a state machine diagram, this transition is executed synchronous with the activity that creates the object. This means that the activity creating an object will pause until this transition, including the behaviors defined on it, is completed. Any error during this transition will abort the object creation and the calling activity will receive an error.
The effect of each transition must be of type **Activity**. This means that an operation is called after creating the persistent state object. In the example above, the xUML Runtime calls the activity diagram **Initialize Handler** of the **PurchaseOrder** class.

Signal Events

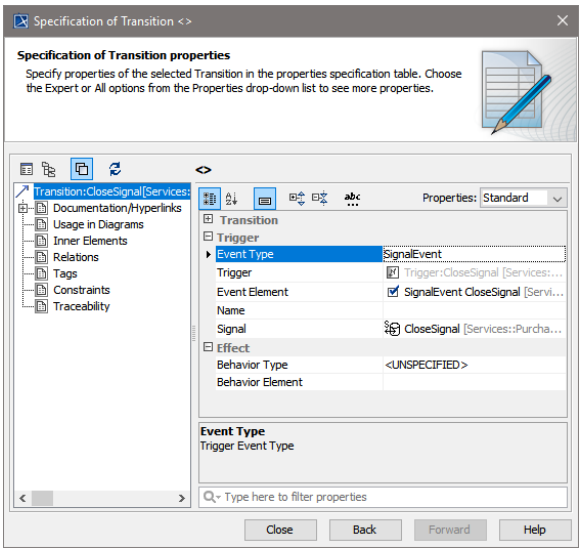
Transitions starting from states can be triggered by signals. The example below shows a transition that is triggered by the **CloseSignal**.

Figure: Simple Example of a Transition Triggered by a Signal



Signals are message structures being sent asynchronously to objects (see also chapter Signals). If the object is in a state being able to handle the signal, the transition triggered by the signal executes the associated signal handler.

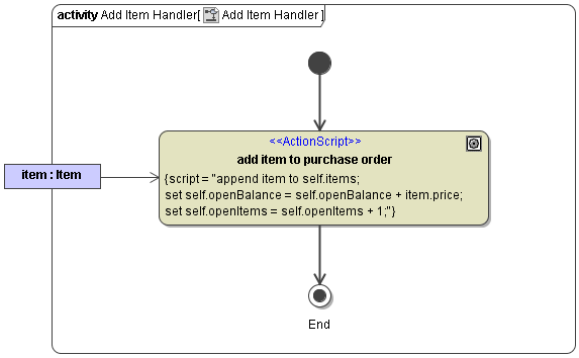
Figure: Using a Signal to Trigger a Transition



Signals processed by a transition are defined in the **Trigger** compartment of the transition specification.

The **Effect** compartment of the transition specification defines which activity is used to handle a signal. However, it is allowed to define transitions not having any signal handler. Such transitions just change the state of the object and do not have a signal handler. If a signal handler is given, it is defined as depicted in figure Defining the Effect of Creating a Persistent State Object in MagicDraw.
If no error occurs executing the signal handler, the object changes its state afterwards. Otherwise, it stays in the source state. While executing a signal handler, no other signal is delivered. This behavior is known as "Run-To-Completion" semantics and is part of a set of rules that governs the asynchronous behavior of signals. More rules about the behavior of signals are described in chapter Signals.
Frequently, signals have attributes to pass data from an activity to an object. The following figure shows the activity **Add Item Handler** triggered by **AddItemSignal** (as defined in figure State Machine Diagram of a Purchase Order).

Figure: Changing a Persistent State Object in a Signal Event Handler



The signal parameter **item**, which was defined in the signal declaration, is mapped to the input parameter **item** of the **Add Item Handler** activity, which is the event handler defined on the transition in the state machine diagram for **PurchaseOrder**.
Inside the activity diagram of the event handler, the attributes of the persistent state object can be modified using the **self** keyword. The changes will be persisted automatically after successful completion of the transition.
Optionally a local object **currentEvent** can be used to access meta data about the delivered event. **curre ntEvent** is an instance of the **Event** class and the most commonly used attribute is **eventSource**, containing a unique conversation ID shared by the sender and receiver of the signal. See chapter Conver sations for more information.

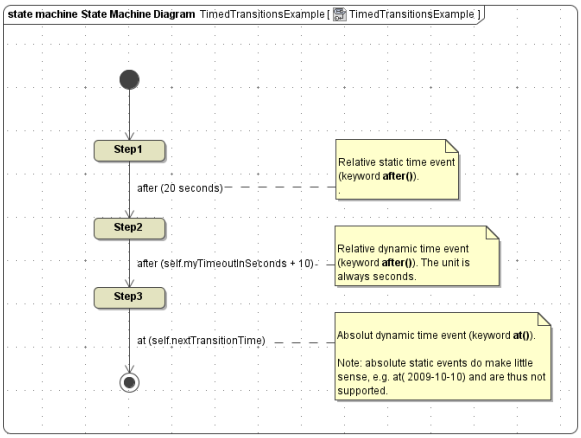
Time Events

Example File (Builder project Advanced Modeling/PState):

<your example path>\Advanced Modeling\PState\uml\pstateTimeTrigger.xml

Besides signals, time events can trigger transitions. For example, the following state machine diagram shows the different kinds of time event supported by the xUML Runtime:

Figure:Supported Time Events



Basically, three kinds of time events are supported:

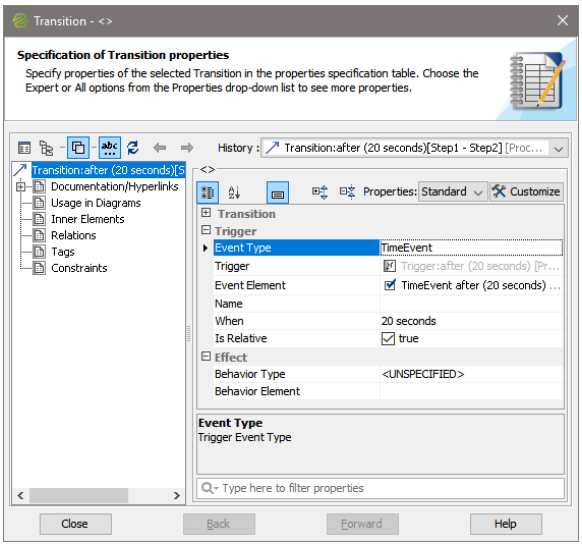
Event Type	Description
Relative time events using a literal expression	See for example the first transition in the above state chart. The **time expression** consist of a comma-separated list, like for example 2 days, 12 hours. The list elements or tokens consist of numbers and units. Allowed units are second, minute, hour and day. Units can be given in singular or plural form. Thus, 3 seconds, 1 minute is allowed. Furthermore, all units are case insensitive. Seconds can be specified as non-negative floats. All other units must be specified as positive integers. In order to tell the model compiler that the time expression defines a relative time duration, set the **Is Relative** flag to true. If this is not done, the compile will print a warning.
Relative time events using dynamic expressions	See for example the second transition in the above state chart. Return type of the expression must be **Integer**. Unit of measure is seconds. This is sensible for short timeouts, e.g. 30 seconds. Again, the **Is Relative** flag must be set to true.
Absolute time events using dynamic expressions	See for example the third transition in the above state chart. The return type of the expression must be **DateTime**. This is sensible for time events best calculated dynamically. For instance, an event triggering the transition in three years.

The prefix **after** or **at** is added by MagicDraw automatically for relative respectively absolute time triggers. You only need to add the time expression in the time event specification dialog.

A time event is defined in the **Trigger** compartment of the specification dialog of the transition:

1. **Event Type** is set to **TimeEvent**.
2. A time expression is filled into field **When**.
3. Like with signal events, the activity invoked during the time-out transition is defined in the **Effect** compartment.

Figure: Using a Time Event to Trigger a Transition



Transitions without an explicit trigger are handled like time events with a time-out of 0 seconds. These are also called **completion events** as the transition is triggered as soon as the source state completes its activity.
Similar to signal events, activities invoked by a time event can access and modify a persistent state object using the self keyword and use **currentEvent** to access meta data about the time event.