# Accessing Event data

Transitions on State Machines are triggered by Events. Events may have associated data and, for certain kinds of Events, this data may be accessed within the Alf code used within the State Machine. Specifically:

- For a Signal Event, the values of the attributes of the received Signal instance are the Event data.
- For a Call Event, the values of the Parameters of the called Operation are the Event data.

This event data may be accessed within guard Expressions and effect Behaviors on Transitions and within entry, do-activity and exit Behaviors on States. For entry and do-activity Behaviors, the relevant Events are those on the *incoming* Transitions of a State, but, for an exit Behavior, the relevant Events are on the *outgoing* Transitions of the State.

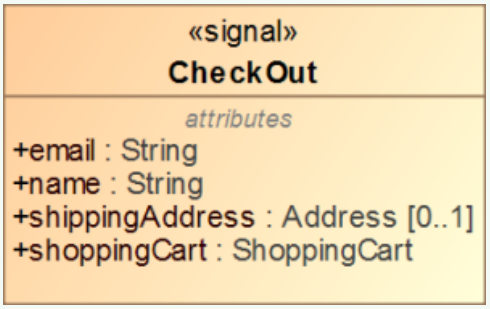
To access Signal Event data in Alf

---

- Use the special Parameter name *evt*, which has the Signal being received as its type. Use the usual dot notation to access Signal attributes.

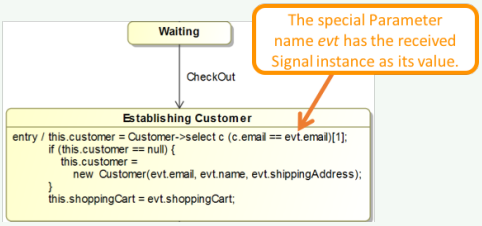
> ⚠ Signal Properties Visibility needs to be set to **public** to access them with evt Parameter.

Consider an e-commerce application that allows customers to place items in a shopping cart and then check out and pay for them. The *CheckOut* Event can be modeled using a Signal with relevant customer information, as shown below.



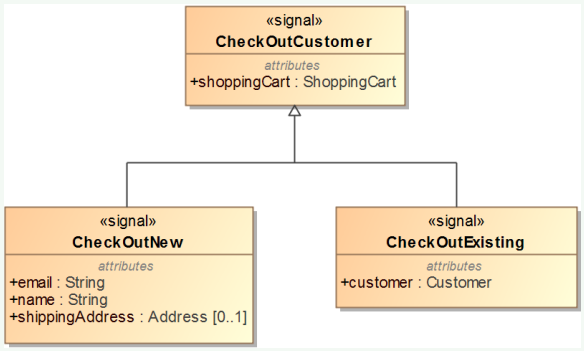

CheckOut Signal

The StateMachine diagram snippet below shows a Transition triggered by the *CheckOut* Signal, targeting a State with an entry Behavior to handle the Signal Event by initializing a new order for the customer.



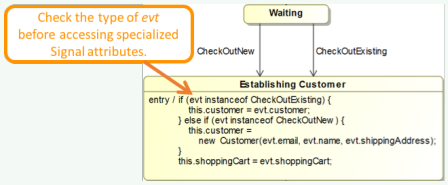

Handling the CheckOut Signal Event

The Behavior above first checks if there is an existing customer with the given email and, if not, creates a new customer record using the *CheckOut* data. An alternative approach would be to using different Signals for checking out a new customer and checking out an existing customer, as shown below.




CheckOutNew and CheckOutExisting Signals

In this case, the *Establishing Customer* State will have two incoming Transitions, one for each Signal. By first checking which kind of Signal was actually received, the Alf code for the entry Behavior can then access the appropriate Event data in order to either create a new customer record or link to an existing customer, as shown below. Note that, because both Signals are specializations of the general Signal *CheckOutCustomer,* it is possible to access the common *shoppingCart* attribute without having to check which Signal was received.




Handling multiple Signal Events

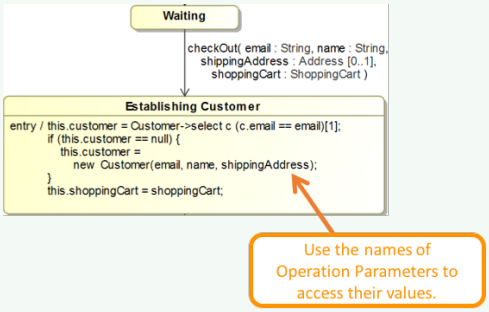
To access Call Event data in Alf

---

- Use the names of the Parameters of the Operation to access their values. Output values for *out* and inout Parameters may be set using assignment, and those values are returned at the end of the run-to-completion step for the Call Event.

> ⚠ Output Parameters are available in entry and exit Behaviors, but *not* in do-activity Behaviors, since do-activity Behaviors can continue to run asynchronously after the end of the run-to-completion step.

An alternative model would be to give the Order class that owns this StateMachine a *checkOut* Operation, with the StateMachine then handling a call to this Operation using a Call Event, as shown below. The entry Behavior in this case does the same thing as in the case above using a Signal Event, but note that the Parameters of the *checkOut* Operation are now used directly, instead of the property-access notation used for Signal Event data.




Handling the checkOut Call Event

Unlike Signal Events, access to Call Event data is only allowed if all relevant Transitions for a State Behavior (incoming for entry and do-activity, outgoing for exit) have Call Events for the same Operation. It is also not possible to access the Event data if Signal Events and Call Events are mixed, or are mixed with other kinds of Events (such as Time Events and Change Events).