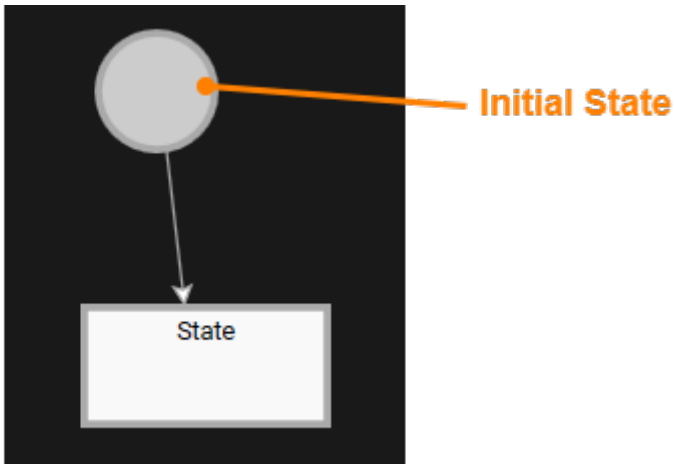
Components in State Machine

A State Machine diagram is built by combining core components such as **States** and **Transitions.** Typically, a set of states are interconnected through transitions to represent the flow of logic.

The following eight types of elements can be used as components of a State Machine in the Scene Composer. These components are available in the toolbar at the top of the State Machine Editor panel and can be added to the diagram using a simple drag-and-drop action.

Initial State

The **Initial State** is a special state that marks the starting point of the State Machine. Every State Machine contains exactly one Initial State. When the State Machine is executed, processing always begins from this Initial State.

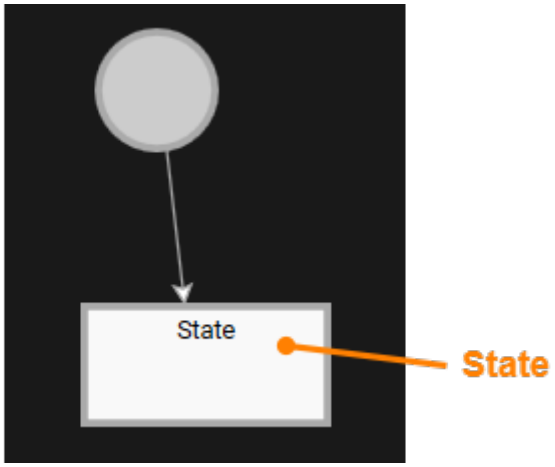


State

A **State** represents a specific logical condition within a State Machine. It reflects the current operation and context of the system. Specific actions can be executed when a State is entered or exited:

- **OnEntry Actions** - Executed when the State starts.
- **OnExit Actions** - Executed when the State ends.

These actions can be configured in the Properties Panel or the Fusion Editor after selecting the target State.

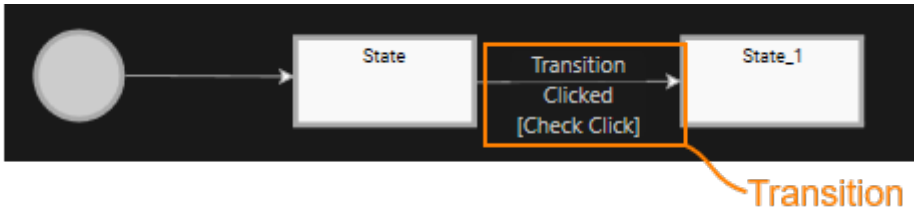


Transition

A Transition represents a connection between two States and is executed when specific conditions are met.Each transition consists of:

- **Sender** - The source that triggers the **Event**, which in turn initiates the **Transition.**
- **Event** - The event that initiates the Transition.
- **Condition** - A logical expression that must evaluate to true for the transition to occur.
- **Actions** - The operations executed when the condition is satisfied.

Transitions are evaluated in the logical order defined within the State Machine. If multiple transitions originate from the same State, you an assign priorities to determine which transition should be executed first. For more details, see here.



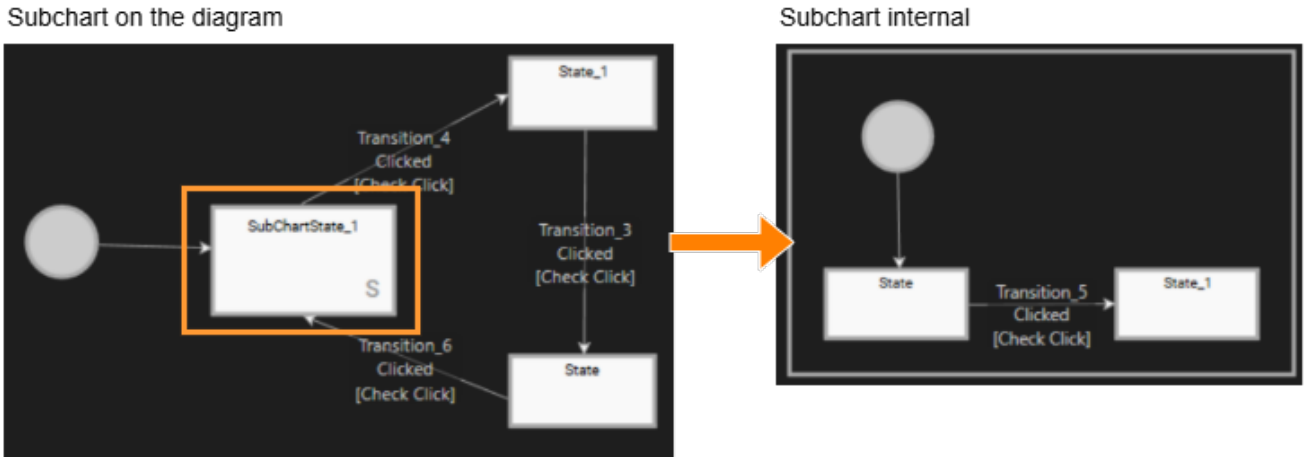
Subchart

A Subchart is a structure that groups one or more **States** and **Transitions** into a single unit. In the diagram, the lower-right corner of a Subchart is marked with an "**S**" to distinguish it from a normal State.



Subcharts can be connected to other States or Subcharts, enabling the creation of flexible and modular flow structures. **Double-clicking** a Subchart opens its internal details, allowing you to view and edit its contents.

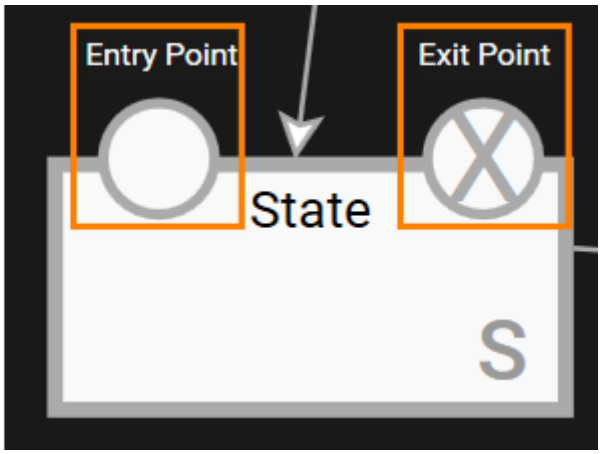
Because Subcharts can be edited independently of the main diagram, they are ideal for organizing complex logic into smaller, easy-to-understand sections.



Entry/Exit Point

Entry Points and Exit Points are pseudo-states that enable controlled entry into and exit from a **Subchart**. They serve as clear starting and ending positions for navigating a Subchart.

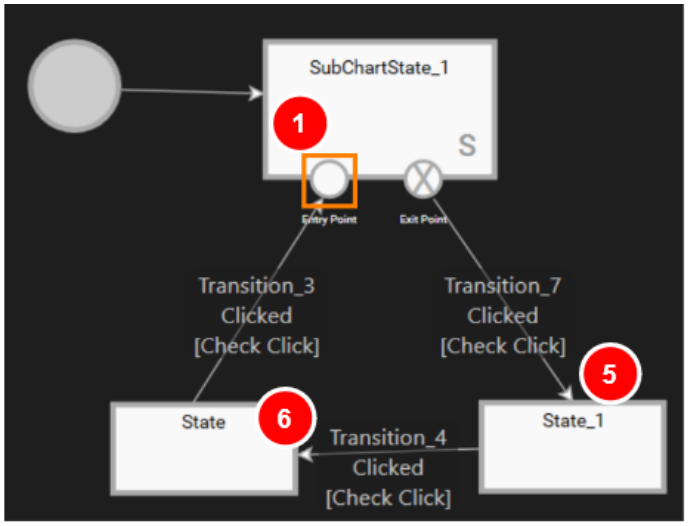
- **Entry Points**: Allow the Subchart to start from different positions based on specific conditions. Multiple Entry Points can be defined to handle various scenarios.
- **Exit Points**: Provide controlled exits from the Subchart. Multiple Exit points can be configured to return to different states depending on the conditions



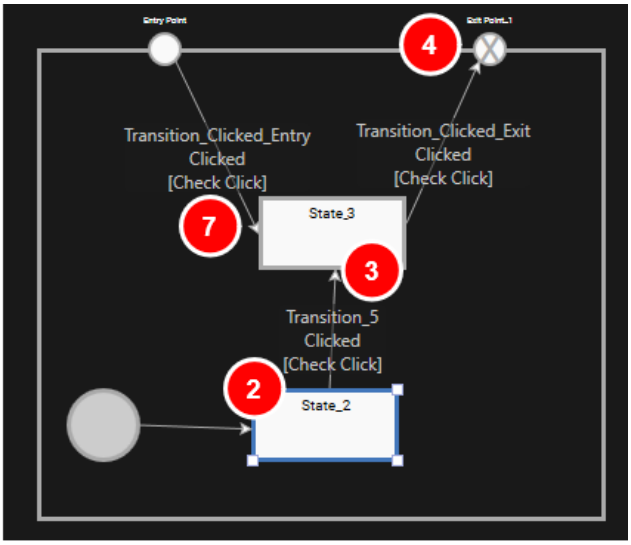
Using Entry and Exit Points enables the creation of modular and condition-driven flows within complex State Machine diagrams.

Example of Subchart Configuration Using Entry/Exit Point

The following example shows a simple configuration that incorporates a **Subchart** along with **Entry Points** and **Exit Points**.



Base Diagram



Subchart Diagram

In the figure above, the Subchart named "SubChartState_1" is placed within the State Machine diagram.

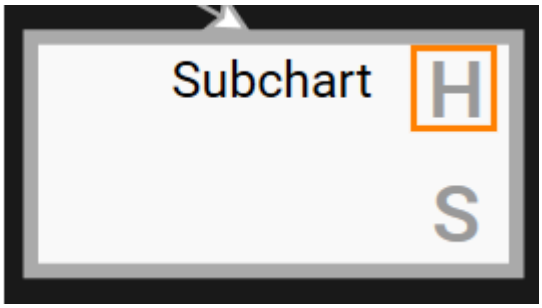
1. The state machine starts at the Initial State in the base diagram (see left figure above). From there it transitions into the Subchart state "SubChartState_1".
2. Inside the Subchart (see right figure above), the flow begins at the SubChart's Initial State and proceeds to State_2.
3. A mouse click triggers "Transition_5", moving the flow to State_3 within the Subchart.
4. Another mouse click initiates a transition via the Subchart's Exit Point.
5. And returns to the base diagram at State_1.
6. A subsequent mouse click transitions to the base diagram's "State".
7. With yet another mouse click, "Transition_3" in the base diagram is triggered, leading to the Entry Point of SubChartState_1 (orange frame in the left diagram). This re-enters the

Subchart via the Entry Point and returns to State_3 inside the Subchart..

History State

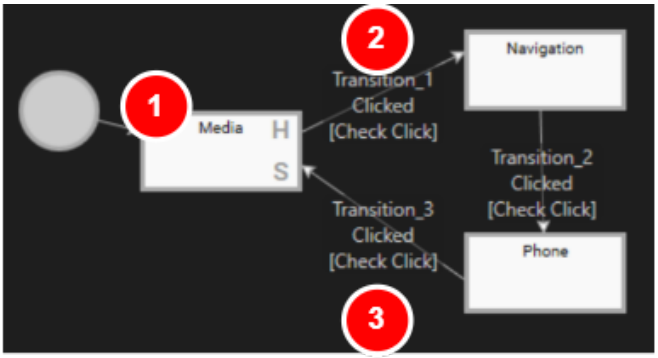
A **History State** is a special State that can be placed inside a **Subchart**. Its purpose is to remember the last active state within the Subchart. When the flow transitions back into the Subchart, processing resumes from the state that was active immediately before exiting, rather than starting from the Subchart's Initial State.

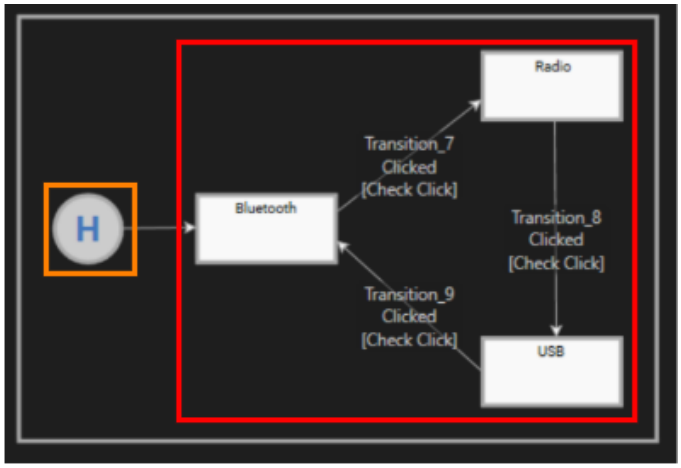
Subcharts that contain a History State are marked with an "**H**" in the top-right corner of the Subchart in the diagram.



Example of History State Configuration

The following example demonstrates a simple configuration where a History State is placed inside a Subchart. This setup allows the State Machine to resume from the last active state within the Subchart when re-entering, rather than starting from the Initial State.





In the figure above, the State Machine diagram contains a Subchart "Media". The Subchart itself includes a History State instead of an Initial State.

- The State Machine starts at the Initial State in the base diagram and transitions to the Subchart "Media" (see the left diagram above).

- Upon entering the Subchart, it first passes through the History State (orange boxed area) and then enters the **State loop** (red boxed area), starting with the "Bluetooth" state.
- The state machine remains in this Subchart as long as transitions between **Bluetooth**, **Radio**, and **USB** are triggered.
- When Transition_1 (marked as 2 in the figure) is triggered, the Subchart exits.
- At this point, the **History State** records the last active state within the Subchart before the exit.
- If the State Machine transitions back to the Subchart "Media" via Transition_3 (marked as 3 in the figure), it re-enters the Subchart.
- Thanks to the History State, the Subchart resumes from the last active state (e.g. Bluetooth, Radio or USB) instead of starting from the Initial State.

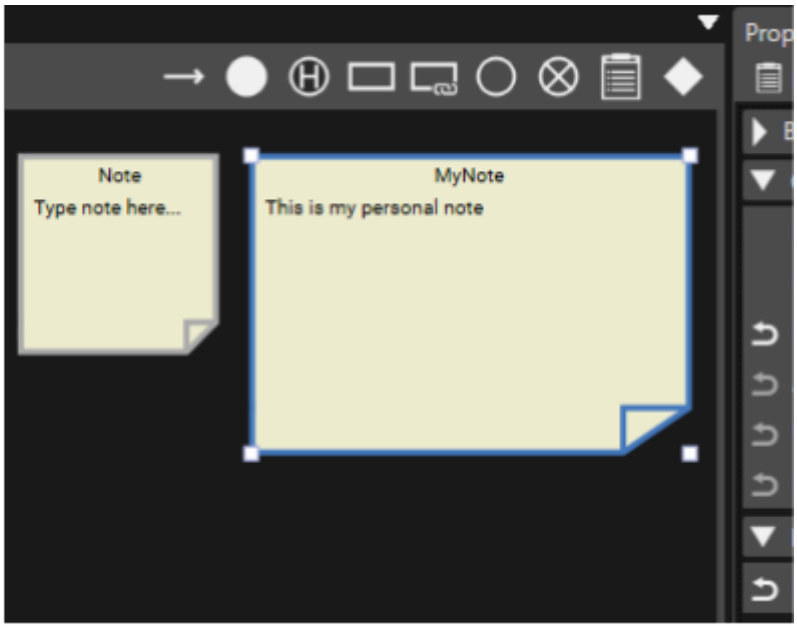
> When exiting the **Media** state and transitioning to the **Navigation** state, the system executes not only the **OnExit** actions of the **Media** state, but also the **OnExit** actions of the last active Media device state (e.g. **Bluetooth, USB** or **Radio**).

> When entering the **Media** state, the system executes both levels of OnEntry actions: the Media state's OnEntry actions and the OnEntry actions of the stored most recently selected Media device state (e.g. **Bluetooth**, **USB** or **Radio**).

For an example implementation that demonstrates the use of a **History State** within a Subchart, see State Machine Solution.

Note

In the **State Machine Editor**, you can add **Notes** to the diagram to provide additional information, such as TODOs, caveats, implementation details or design considerations. Notes help improve clarity and make complex diagrams easier to understand and maintain.

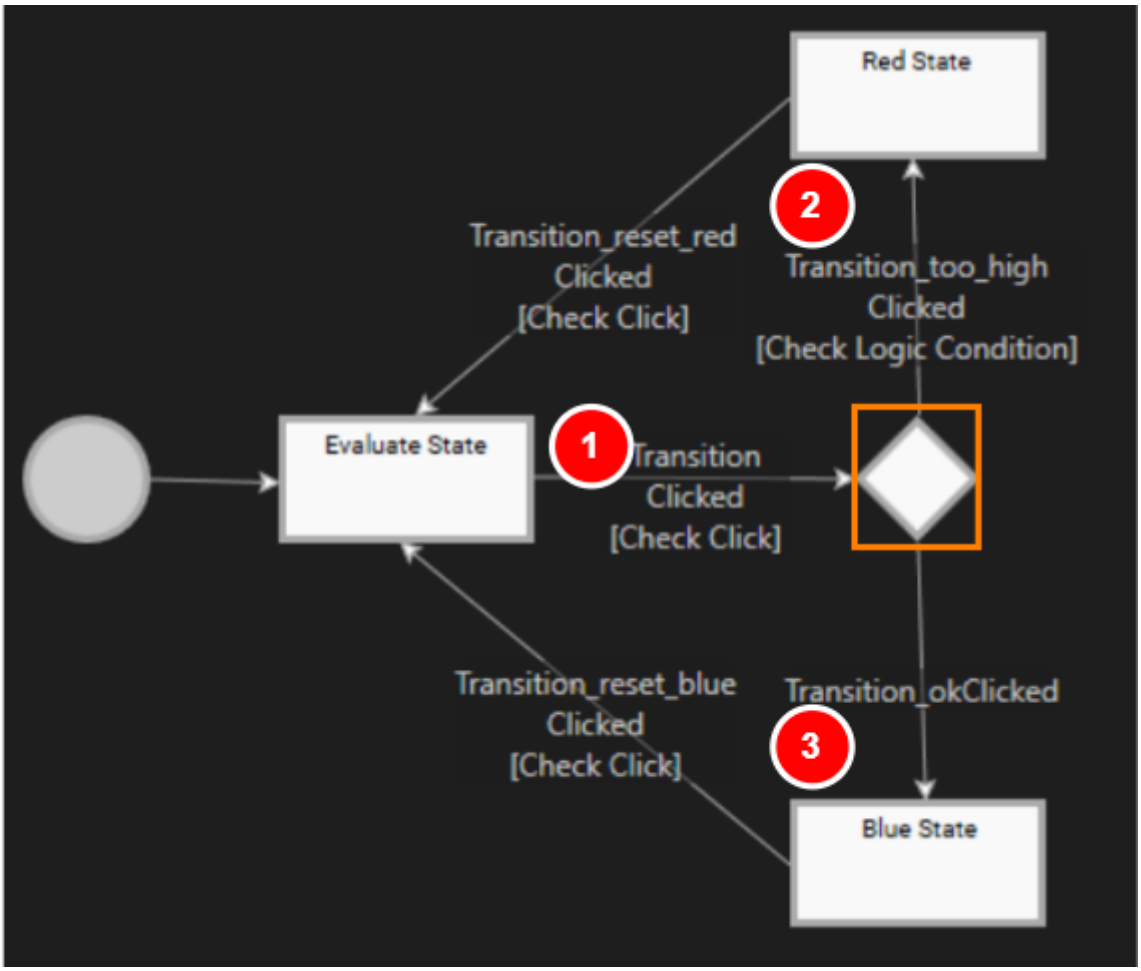


Choice

A **Choice** element allows you to branch **Transitions**, enabling the State Machine to enter a specific state based on the evaluation of multiple conditions. This feature simplifies the design and maintenance of complex logic where a set of conditions must be checked to determine the next state.

Example of Choice Usage

In the State Machine diagram below, the transition begins at the Initial State and proceeds to the **Evaluate State**.



When "Transition" (marked as 1 in the above figure) is triggered by a mouse click, the State Machine moves to the **Choice** element (orange-framed section in the figure). At this point, the transition branches based on the evaluated conditions:

- "**Transition_too_high**": The processed value is checked against a specific threshold.
  - **If the condition is met** (value exceeds the threshold): The State Machine transition to "**Red State**" (marked as 2 in the figure).
- "**Transition_okClicked**":
  - **If the condition is not met** (value is below the threshold): Transition_okClicked is executed, and the State Machine moves to "**Blue State**" (marked as 3 in the figure).

The priority of an **Exit Transition** connected to a **Choice** element is the same as the normal Transition priority change procedure.

> By utilizing **Choice** elements, multiple conditions can be clearly organized within the State Machine, making the branching structure easy to visualize. This makes it possible to design complex logic in a simple manner.

ViewState

ViewState is a specialized State used to control whether a specific **View** is displayed or hidden during the State Machine flow. It is linked to a corresponding View defined in the **View Definition Editor**.

There are two ways to created a ViewState:

1. **Dragging a View from Solution Explorer**
   - Drag a View directly from the Solution Explorer and drop it onto the State Machine Editor diagram
   - A corresponding ViewState is created automatically.
2. **Using the State Machine Editor toolbar**
   - Select the ViewState element from the toolbar and place it onto the diagram.
   - In the Properties panel, configure the ViewState's view.

> When configuring a State Machine that switches from displaying one scenario to displaying another, the configuration effort is significantly reduced when using **ViewStates** instead of **States** and **TransitionRequests**.

Please also see Jump To View Action.

Revision #18
Created 2023-02-15 08:29:39 UTC by Tsuyoshi.Kato
Updated 2026-01-30 13:31:37 UTC by Elisabeth Zauner