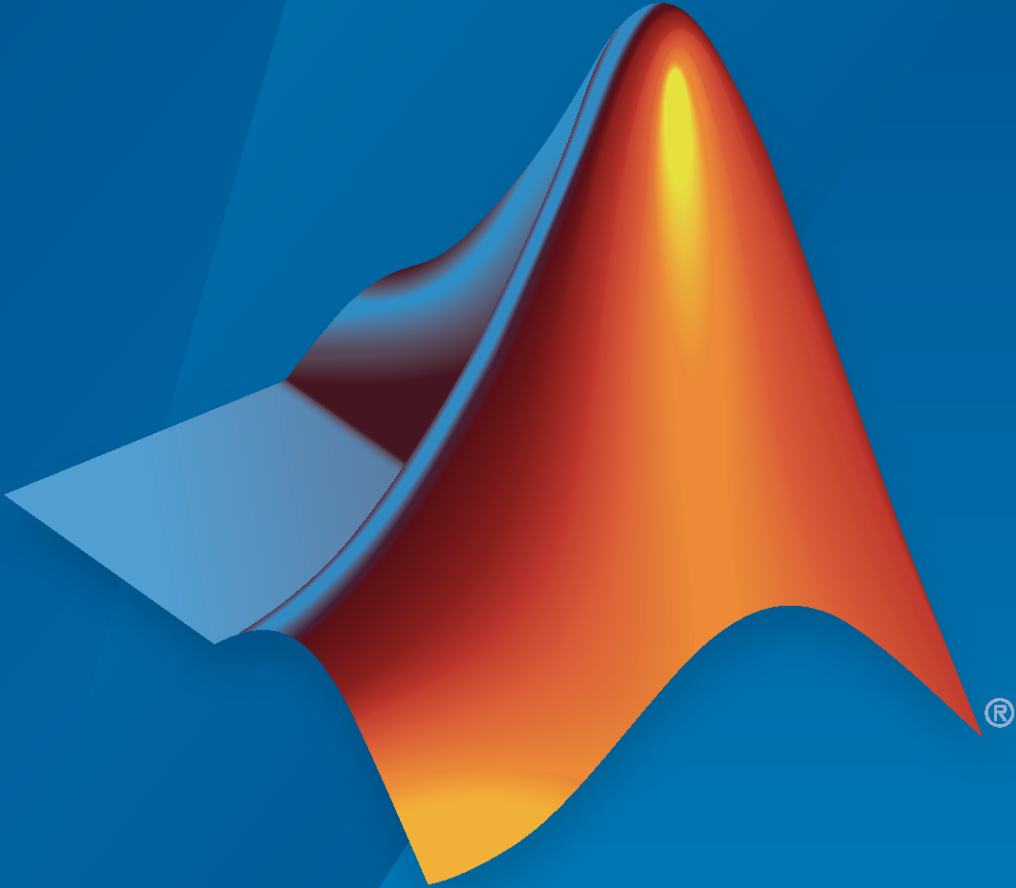


# RoadRunner Scenario Release Notes



## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



The MathWorks, Inc.  
1 Apple Hill Drive  
Natick, MA 01760-2098

### *RoadRunner Scenario Release Notes*

© COPYRIGHT 2022 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

### **Trademarks**

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### **Patents**

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

<b>Pedestrian Actors: Design scenarios with pedestrian actors</b> .....	<b>1-2</b>
<b>Actor Groups: Attach child actors to a parent actor</b> .....	<b>1-2</b>
<b>Actor Groups: Model actor group behavior in MATLAB</b> .....	<b>1-2</b>
<b>User-Defined Actions: Process custom actions created in Roadrunner Scenario using MATLAB and Simulink</b> .....	<b>1-2</b>
<b>Longitudinal Distance Action and Condition: Configure the motion of an actor based on longitudinal distance to a reference actor</b> .....	<b>1-3</b>
<b>AND Condition: Combine multiple singular conditions with AND logic</b> .....	<b>1-4</b>
<b>Phase State Conditions: Check the runtime state of a phase</b> .....	<b>1-4</b>
<b>Global Parameters: Adjust conditions for the entire scenario</b> .....	<b>1-5</b>
<b>Parameter Data Types: Specify the data type of parameter values</b> .....	<b>1-6</b>
<b>Action Types: Change speed and acceleration units</b> .....	<b>1-9</b>
<b>remapAnchor Function: Create and Remap Anchors Programmatically</b> .....	<b>1-11</b>
<b>Import CSV Trajectories: Import trajectory data interactively and programmatically</b> .....	<b>1-11</b>
<b>ASAM OpenSCENARIO Export Updates</b> .....	<b>1-11</b>
<b>Application Examples: Using MATLAB and Simulink with RoadRunner Scenario</b> .....	<b>1-12</b>
<b>Functionality being removed or changed</b> .....	<b>1-12</b>
Behavior Changes for Interrupted and Continuous Actions .....	<b>1-12</b>

<b>Introducing RoadRunner Scenario</b> .....	<b>2-2</b>
<b>Scenario Editor: Design scenarios for simulating and testing automated driving systems</b> .....	<b>2-2</b>
Define Actors and Paths .....	2-2
Define Scenario Logic .....	2-3
Simulate Scenarios .....	2-4
Validate Scenarios .....	2-5
<b>ASAM OpenSCENARIO Support: Import and export scenarios using ASAM OpenSCENARIO file format</b> .....	<b>2-5</b>
<b>Programmatic Scenario Interfaces: Generate scenario variations, import and export scenarios, and simulate scenarios using MATLAB functions or a gRPC API</b> .....	<b>2-5</b>
<b>Simulate RoadRunner scenarios with actors modeled in MATLAB and Simulink</b> .....	<b>2-6</b>
<b>CARLA Actor Cosimulation: Visualize actors designed in CARLA within RoadRunner Scenario using the cosimulation platform</b> .....	<b>2-7</b>

# R2022b

---

**Version: 1.1**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

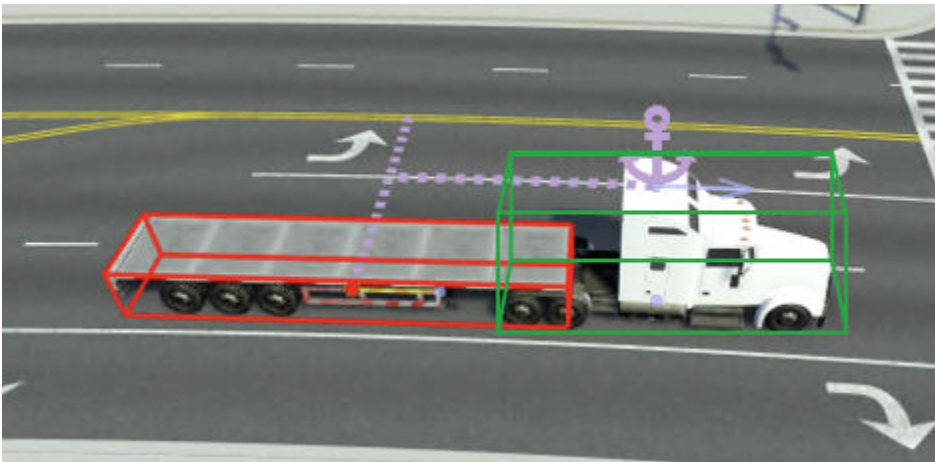
## Pedestrian Actors: Design scenarios with pedestrian actors

You can add pedestrian actors to scenarios in RoadRunner Scenario. You can assign pedestrian paths and behaviors independently or relative to vehicle actors in the scene.

RoadRunner provides a set of standard pedestrian actors with their own skeletons, meshes, and animations that you can use to quickly populate a scene. You can also import custom pedestrian actors from third-party sources, such as Blender® or Unreal Engine®. For more information, see “Import Custom Character Meshes”.

## Actor Groups: Attach child actors to a parent actor

You can combine multiple actors into a single actor group, such as a truck with a trailer. Create an actor group by dragging and actor onto another actor in the scenario canvas. RoadRunner Scenario automatically designates the actor you drag as a child actor of the actor you drag it to (the parent actor), and assigns it to the attachment point of the parent actor. The child actor can rotate at the attachment point, along all three axes of the scenario, constrained by its regular interactions with the environment and other actors, including the parent vehicle.



## Actor Groups: Model actor group behavior in MATLAB

Automated Driving Toolbox™ provides a MATLAB® interface to model custom behavior for an actor group.

Retrieve a child actor using the `getAttribute` object function of a `Simulink.ActorSimulation` object that represents a parent vehicle. Specify the behavior of the child relative to its parent by updating its attributes, such as pose and velocity, within a MATLAB System object™.

To apply the new behavior to an actor group, associate the System object to the parent actor of that group.

## User-Defined Actions: Process custom actions created in Roadrunner Scenario using MATLAB and Simulink

In R2022b, you can define an action that contains custom parameters and include it in the scenario simulation logic. You can then use a MATLAB System object or Simulink® behavior model to read and

---

process this user-defined action. For example, you can use the custom parameters of a user-defined action to perform computations on actor attributes such as `Pose` and `Velocity`.

Because the RoadRunner Scenario built-in actor behavior model ignores user-defined actions, you must send a user-defined action to a cosimulation actor configured in MATLAB or Simulink for further processing. Use MATLAB functions, such as `getAction`, or Simulink blocks, such as RoadRunner Scenario Reader and RoadRunner Scenario Writer, to retrieve and process user-defined actions.

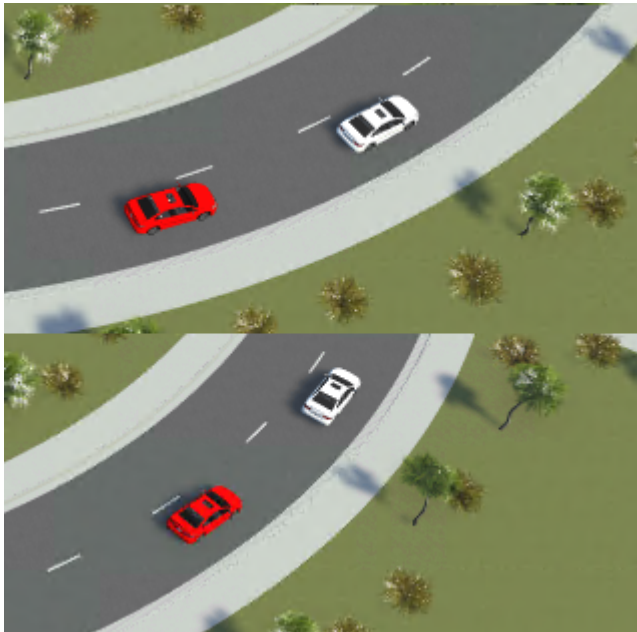
For more information about processing user-defined actions in MATLAB or Simulink, see “Design Vehicle Following User-Defined Actions Scenario”.

## Longitudinal Distance Action and Condition: Configure the motion of an actor based on longitudinal distance to a reference actor

Longitudinal distance is the distance between two vehicles projected onto the curvature of the lane in which they are traveling. You can configure an actor to establish and maintain a specified longitudinal distance from a reference actor by using the `Change Longitudinal Distance` action.

For example, if you want a vehicle to maintain a constant longitudinal distance of 6 meters behind a reference vehicle, you can use a `Change Longitudinal Distance` action with a **Distance Type** of `Space` and **Space Distance Offset** of 6.

For more information, see “Change Longitudinal Distance Actions”.



You can now specify the `Longitudinal Distance to Actor` end condition to action phases in the **Logic Editor**. Use this condition to end the associated action phase when the actor reaches the specified longitudinal distance from the reference actor.

For example, if you want a vehicle to perform a lane change action when it reaches a longitudinal distance of less than 10 meters behind the reference vehicle, you can use a `Longitudinal`

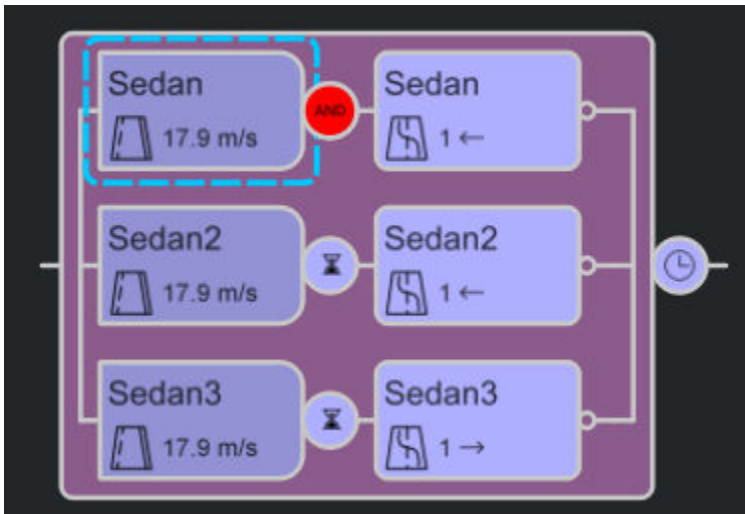
Distance to Actor condition with a **Relative Position** of Behind, a **Rule** of Less than and a **Threshold** of 10.

For more information, see “Longitudinal Distance to Actor Condition”.

## AND Condition: Combine multiple singular conditions with AND logic

You can combine multiple singular conditions using the AND logic condition. Using this condition, the actor transitions to the next action only when all preceding conditions joined by an AND condition have been completed. If any of the actions connected by an AND condition is incomplete, then the actor does not transition to the next action in the scenario logic. You can use the AND condition to check the completion of other conditions.

For example, consider a scenario that consists of three actors: Sedan, Sedan2, and Sedan3. The Sedan2 and Sedan3 actors each have Change Lane actions with associated Duration conditions. The Sedan2 and Sedan3 actors change lanes after their respective specified durations. The Sedan actor also has a Change Lane action with an AND condition assigned to it. Because the AND condition specifies the completion of the lane change actions for both Sedan2 and Sedan3, the Sedan actor performs the lane change action only when Sedan2 and Sedan3 have completed their lane change actions. After Sedan changes lanes, the simulation is complete.



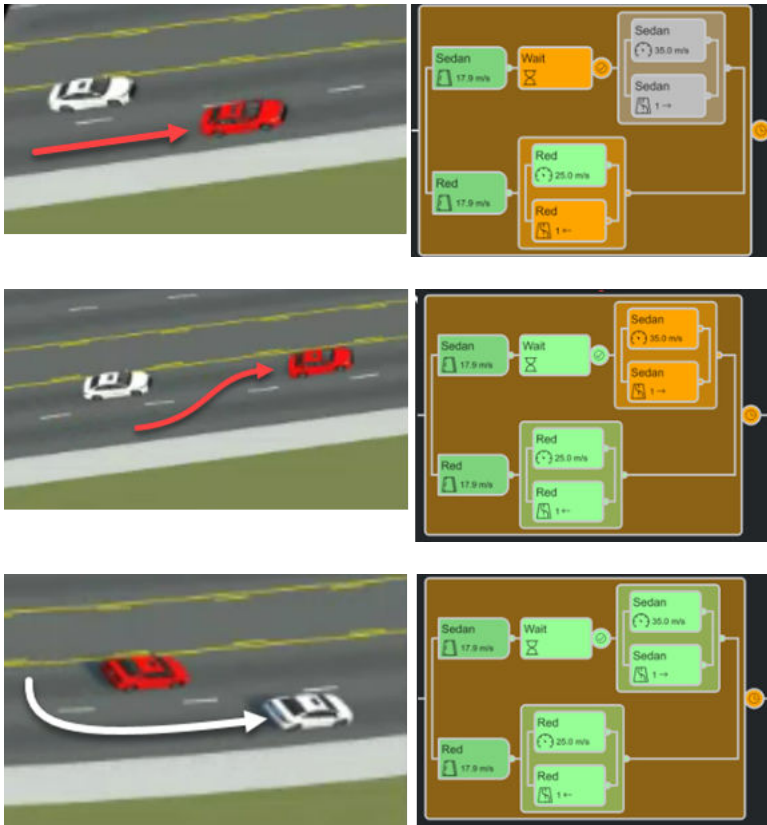
## Phase State Conditions: Check the runtime state of a phase

You can now check the runtime state of a phase in the **Logic Editor** by using phase state conditions. To model a phase state condition, create a condition in the **Logic Editor** and, in the **Attributes** pane, under **Conditions**, select **Phase State**. The phase state condition checks the status of a phase during simulation to determine whether its state is running or complete.

Phase state conditions enable more complex synchronization between actor actions, beyond serial and parallel execution.

In this example, the white sedan has parallel Change Lane and Change Speed actions after a Wait action with a phase state condition linked to the Change Lane and Change Speed actions of the red sedan. As a result, the white sedan waits for the red sedan to finish its lane transition, then changes lanes and accelerates once the lane transition is complete.

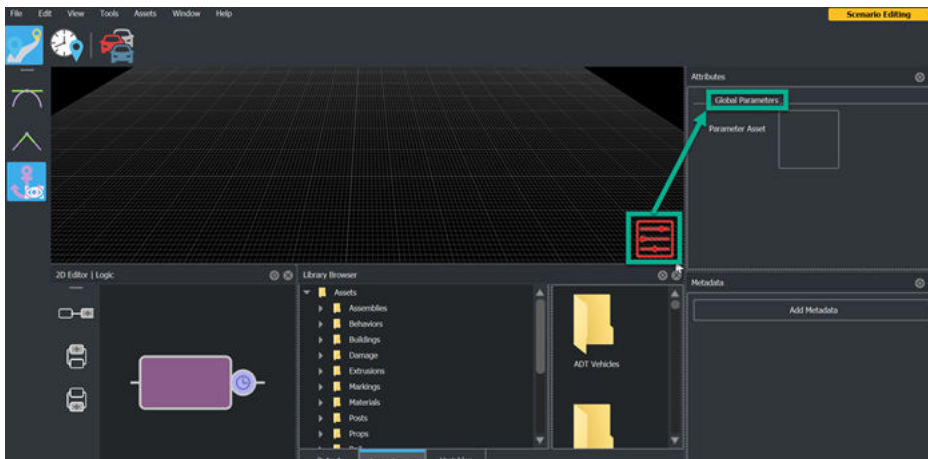


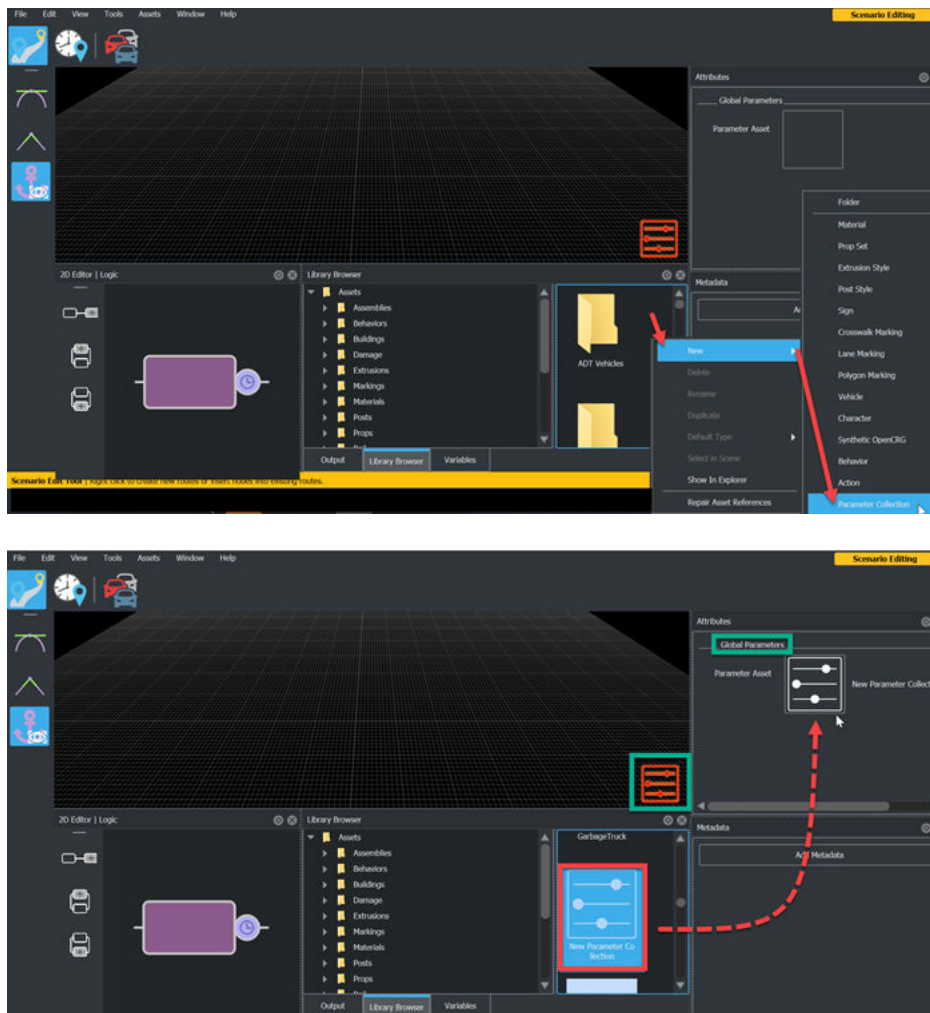


## Global Parameters: Adjust conditions for the entire scenario

As of R2022b, you can set global parameters in RoadRunner Scenario. Unlike behavior parameters, which apply to individual actors, global parameters adjust conditions for the entire scenario. Global parameters are flexible, and you can use them to represent different types of scenario parameters such as speed limit, time of day, and weather.

To create a new global parameter, right-click in the right pane of the **Library Browser**, click **New**, then click **Parameter Collection**. A parameter collection is a group of parameters that can be used in multiple scenarios.



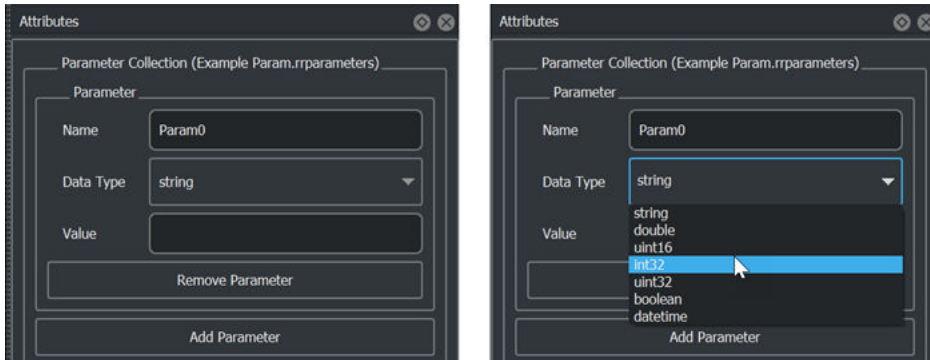


To assign a parameter collection to a scenario, drag the parameter collection asset from the **Library Browser** to the **Parameter Asset** field in the **Attributes** pane. You can set initial parameters by specifying their names in the corresponding **Name** fields and values in the corresponding **Value** fields.

To change global parameters during simulation, add an action in the **Logic Editor**, and, in the **Attributes** pane, set the **Action Type** to Change Global Parameter. To check the value of a global parameter during simulation, use a global parameter condition. To add a global parameter condition, create a condition in the **Logic Editor**, and, in the **Attributes** pane, set **Add Conditions** to Global Parameter. You can combine behavior and global parameters, as well as parameter actions and conditions, to design your scenario logic.

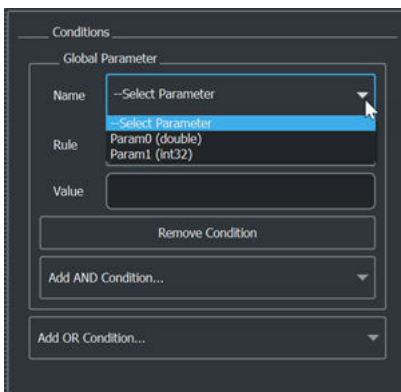
## Parameter Data Types: Specify the data type of parameter values

RoadRunner Scenario supports these data types for parameters: string, double, uint16, int32, uint32, boolean, and datetime. In parameter collections, you can specify the data type of each parameter value.

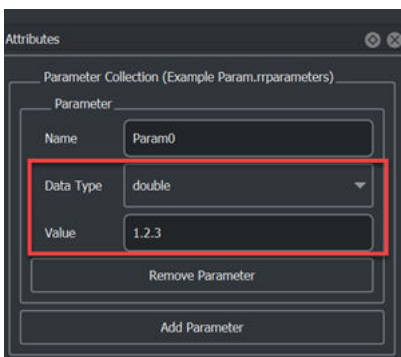


To define the data type for a parameter, select a parameter asset in the **Library Browser** and then, in the **Attributes** pane, select a data type from the **Data Type** list. You can then specify the value of this parameter in the **Value** field. Note that a parameter collection can contain parameters with various data types. Parameter assets imported from MATLAB use the `string` data type by default.

The names and data types of the parameters defined in a parameter collection asset are also displayed as elements of the **Name** list in the **Attributes** pane of conditions and actions.

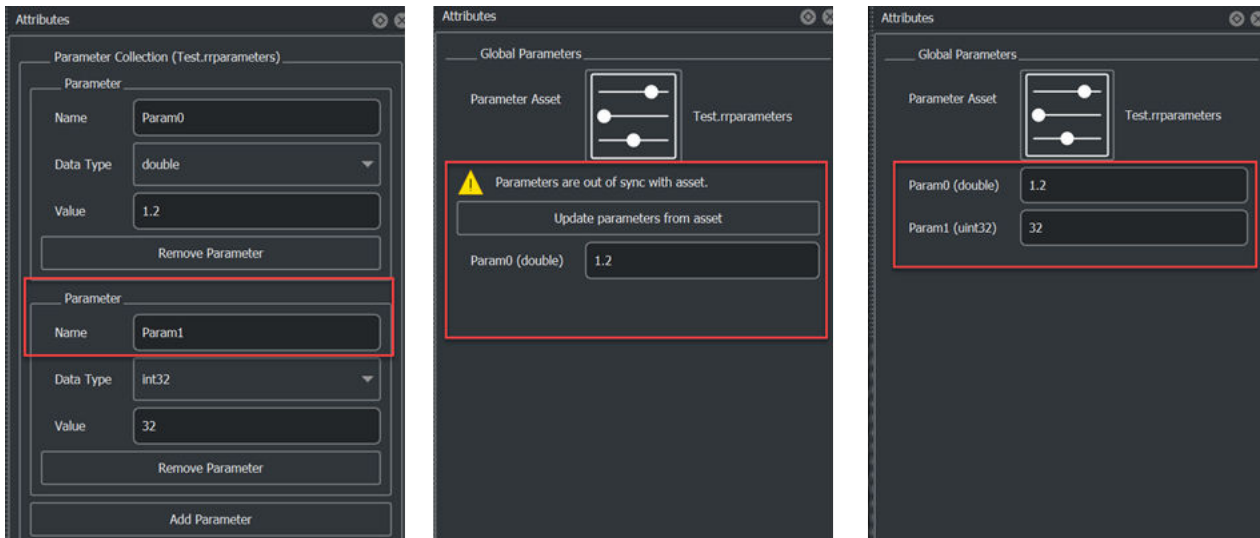


The **Value** field of a parameter must be a valid value for the selected data type. RoadRunner Scenario validates parameter values upon simulation or export, and if it cannot convert the value to the selected data type, then the validation fails and returns an error in the **Output** pane.

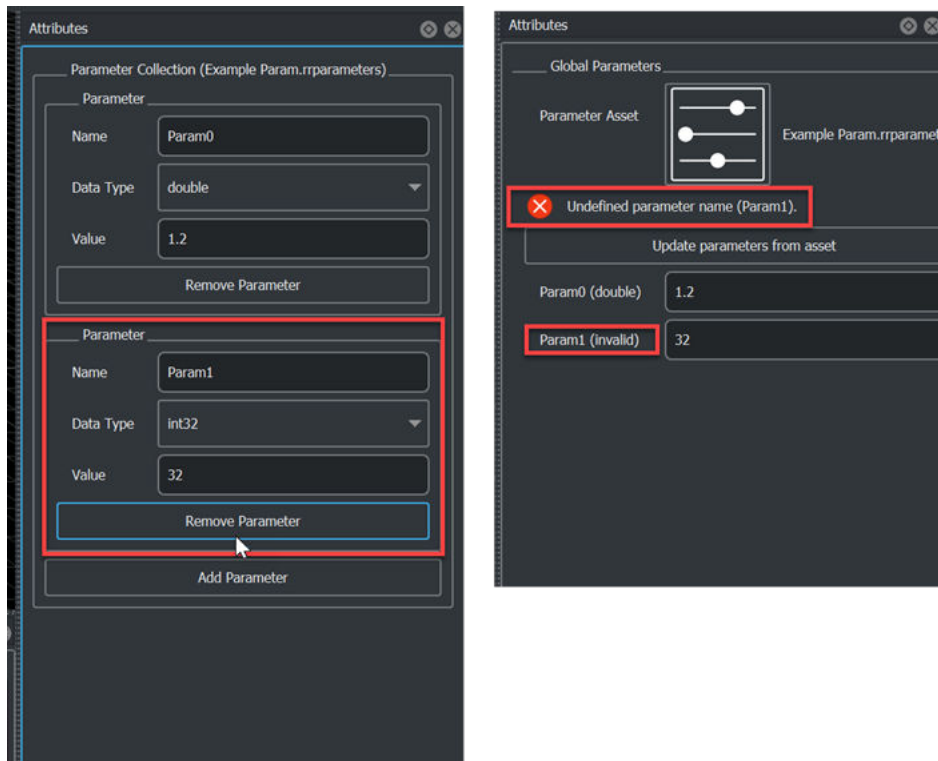


```
> ERROR: Scenario contains critical issues. Refer to the output panel for more details.
> WARNING: ----- Scenario Validation Report -----
> WARNING: Logic Issues:
> WARNING:          CRITICAL ERROR: Failed to convert '1.2.3'. Navigate to parameter collection asset to resolve this error.
> WARNING:          CRITICAL ERROR: Failed to convert '1.2.3'. Navigate to global parameters panel to resolve this error.
> WARNING: ----- End of Report -----
```

If you add a parameter to a parameter collection asset after the parameter collection has been added to the global parameters, RoadRunner Scenario displays an alert in the **Attributes** pane. Click **Update parameters from asset** in the global parameters **Attributes** pane to update the global parameters with the new parameter and clear the alert.



If you remove a parameter from a parameter collection asset after the parameter collection has been added to the global parameters, RoadRunner Scenario displays an error in the **Attributes** pane, and both simulation and export fail and return additional errors in the **Output** pane. Click **Update parameters from asset** in the global parameters **Attributes** pane to update global parameters to reflect the removed parameter. You must then update any actions and conditions that reference the removed parameter.



## Action Types: Change speed and acceleration units

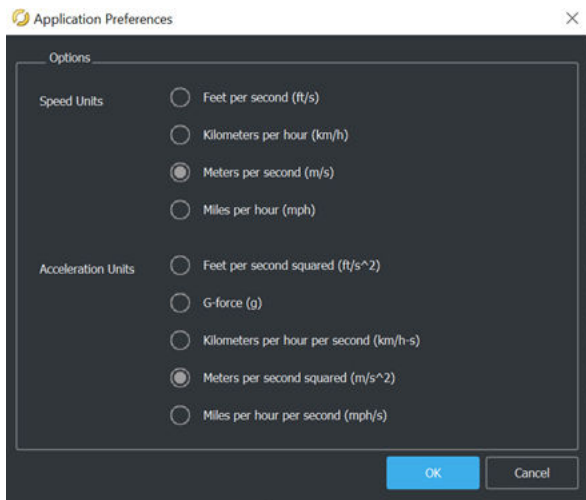
You can change the units of speed and acceleration used by the vehicle actors in RoadRunner Scenario. You can select from these units of speed:

- **Feet per second (ft/s)**
- **Kilometers per hour (km/h)**
- **Meters per second (m/s)**
- **Miles per hour (mph)**

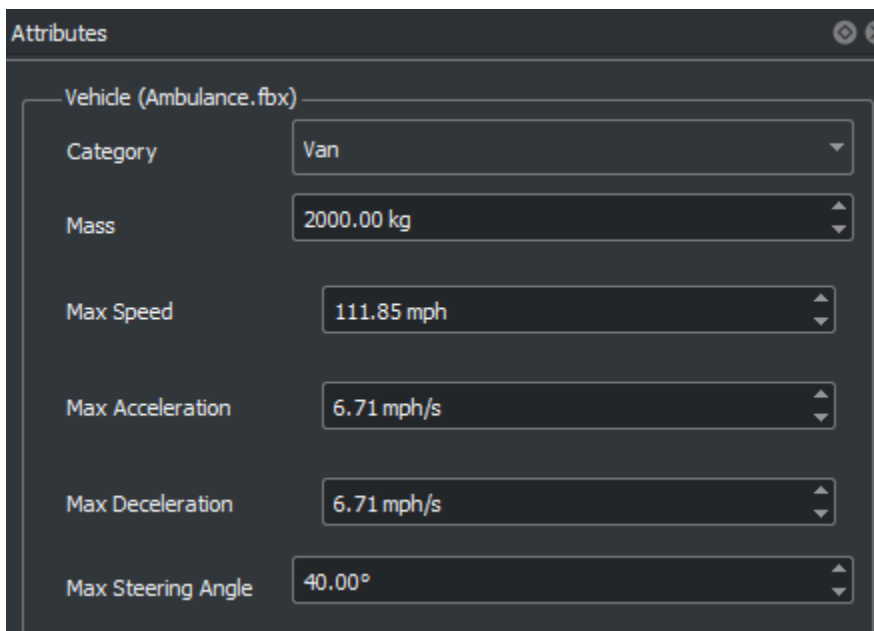
You can select from these units of acceleration:

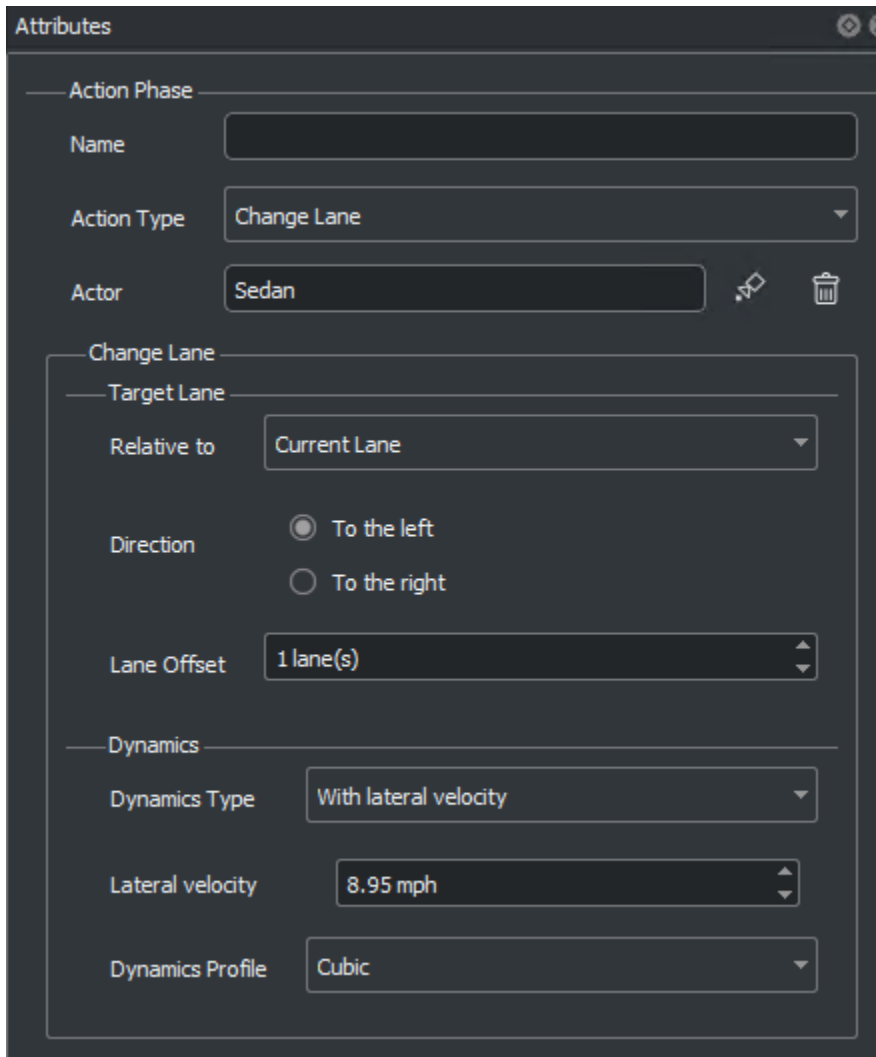
- **Feet per second squared (ft/s<sup>2</sup>)**
- **g-force (g)**
- **Kilometers per hour per second (km/h-s)**
- **Meters per second squared (m/s<sup>2</sup>)**
- **Miles per hour per second (mph/s)**

To select these units, in **Scenario Editing** mode, from the RoadRunner Scenario toolbar, select **Edit**, then **Preferences**.



The **Attributes** pane reflects the selected speed and acceleration units for the vehicle assets and action phases.





## remapAnchor Function: Create and Remap Anchors Programmatically

Use `remapAnchor` function to remap a source anchor to a target anchor. You can specify the name of the target anchor or specify a 3-D scene position at which to create a new target anchor.

## Import CSV Trajectories: Import trajectory data interactively and programmatically

Import trajectories from the CSV file format using RoadRunner Scenario. Alternatively, you can programmatically import scenario files using the `importScenario` MATLAB function or the `Import` method of the gRPC<sup>®</sup> API. For more information, see "Import Trajectories from CSV Files".

## ASAM OpenSCENARIO Export Updates

You can now export these scenario elements to files of the ASAM OpenSCENARIO<sup>®</sup> 1.x and 2.0 formats:

- Change Longitudinal Distance action
- Change Global Parameter action
- User Defined action
- Behavior Parameter condition
- Global Parameter condition
- Longitudinal Distance To Actor condition
- Phase State condition
- Pedestrian actor

RoadRunner now supports additional axles for vehicles that have more than two axles when exporting to ASAM OpenSCENARIO 1.x.

RoadRunner Scenario does not export actor groups to the ASAM OpenSCENARIO 1.x and 2.0 formats because ASAM OpenSCENARIO does not support actor groups. When you export a scenario that contains actor groups, RoadRunner Scenario exports the data of each actor independently.

When exporting to the ASAM OpenSCENARIO 1.x formats, RoadRunner Scenario validates the exported file against the corresponding XML schema to check that the resulting model conforms to the target language.

## **Application Examples: Using MATLAB and Simulink with RoadRunner Scenario**

The “Autonomous Emergency Braking with RoadRunner Scenario” (Automated Driving Toolbox) example shows how to simulate an autonomous emergency braking (AEB) system, designed in Simulink, with RoadRunner Scenario. The example demonstrates how to use vision and radar detection sensors and generate speed variations for both the vehicle under test and the global vehicle target. You can use these processes to test the AEB system per the European New Car Assessment Programme (Euro NCAP) test protocols.

The “Highway Lane Following with RoadRunner Scenario” (Automated Driving Toolbox) example shows how to cosimulate a highway lane-following application, designed in Simulink, with RoadRunner Scenario and Unreal Engine. The highway lane-following application uses an Unreal Engine simulation environment to model detections from camera and radar sensors. The highway lane-following application has controller, sensor fusion, and vision processing components that enable the ego vehicle to follow lanes and avoid collision with other vehicles.

## **Functionality being removed or changed**

### **Behavior Changes for Interrupted and Continuous Actions**

#### *Behavior change*

Starting in R2022b, to align with ASAM OpenSCENARIO execution guidelines, if an action is interrupted, actors no longer continue executing the previous action until an overriding action is sent. If an action is interrupted, the actor now invokes this default control behavior:

- If a longitudinal action is interrupted, the actor retains its current speed.
- If a lateral action is interrupted, the actor corrects its yaw to maintain the lateral offset from its last source.



---

For example, if a vehicle actor is interrupted in the middle of a lane change, the actor stops executing the lane change and maintains its current speed and position, instead of continuing into the destination lane.

Actions that are set to run continuously, and do not have a specified end condition, now continue indefinitely.



# R2022a

---

**Version: 1.0**

**New Features**

## Introducing RoadRunner Scenario

RoadRunner Scenario is an interactive editor that enables you to design scenarios for simulating and testing automated driving systems. You can place vehicles, define their paths and interactions in the scenario, and then simulate the scenario in the editor. You can also generate multiple variations of scenarios programmatically, export them to ASAM OpenSCENARIO, and simulate and visualize them in automated driving simulators and players.

If you design an actor with autonomous behavior in an external simulator such as CARLA, you can assign that actor behavior to a vehicle in your scenario. You can then cosimulate that actor in RoadRunner Scenario and the external simulator. You can also cosimulate actors designed in MATLAB and Simulink and analyze simulation results using MATLAB and Simulink tools.

For an overview of RoadRunner Scenario capabilities, see RoadRunner Scenario Fundamentals.

## Scenario Editor: Design scenarios for simulating and testing automated driving systems

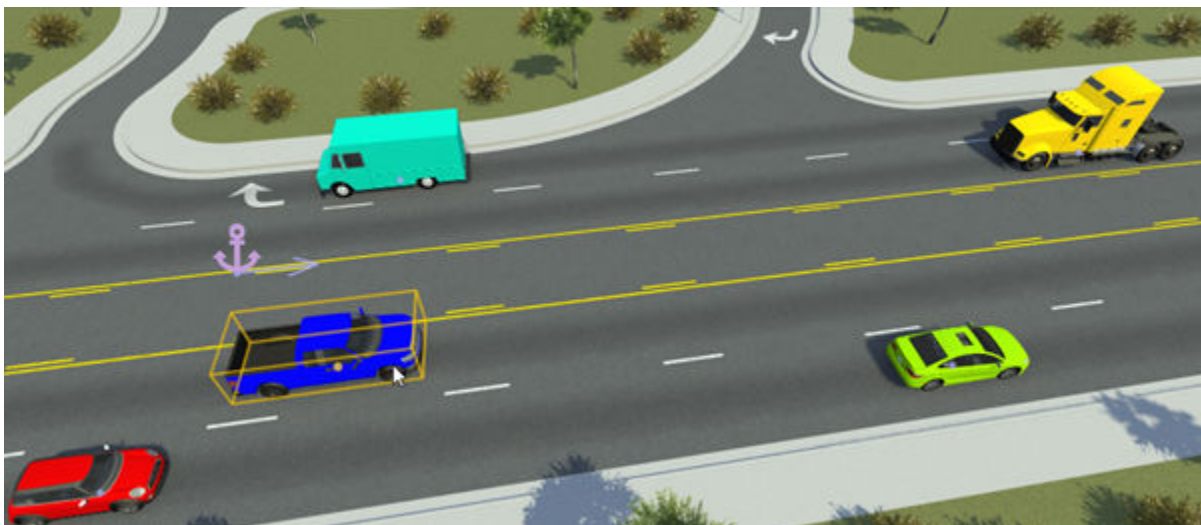
Use the RoadRunner Scenario interactive editor to modify imported scenarios or design your own scenarios based on scenario testing protocols such as ALKS or Euro NCAP®. To get started designing scenarios, see these examples:

- Explore and Simulate a Simple Scenario
- Design Lane Following Scenario
- Design Lane Change Scenario
- Design Lane Swerve Scenario
- Design Path Following Scenario

### Define Actors and Paths

#### Specify Vehicles

In the **Library Browser**, if your project includes RoadRunner Asset Library, then the **Vehicles** folder contains a variety of vehicles that you can add to your scenario.



---

You can modify attributes of vehicles, such as their color. By adding cuboid vehicles to the scenario, you can also modify vehicle dimensions. For more details on vehicles, see **Vehicle Assets**.

If you have your own vehicle meshes defined, you can import them into RoadRunner Scenario and use them in your scenarios. For more details, see Import Custom Vehicle Meshes.

### Specify Driving Paths

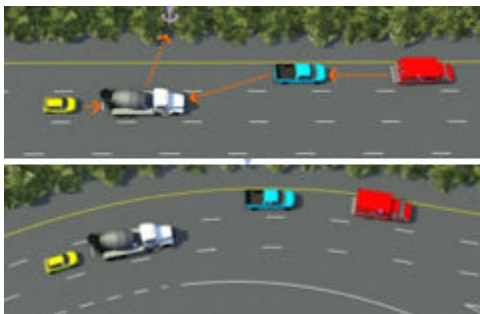
You can set predefined paths for vehicles to follow in a driving scenario. By default, paths snap to lanes, enabling you to quickly create complex paths such as turns and lane changes. For additional flexibility, you can change the shape of paths by modifying tangent waypoints, shift paths laterally within a lane, and specify free-form off-road paths.



For more details on specifying paths, see Path Editing.

### Relocate Scenarios

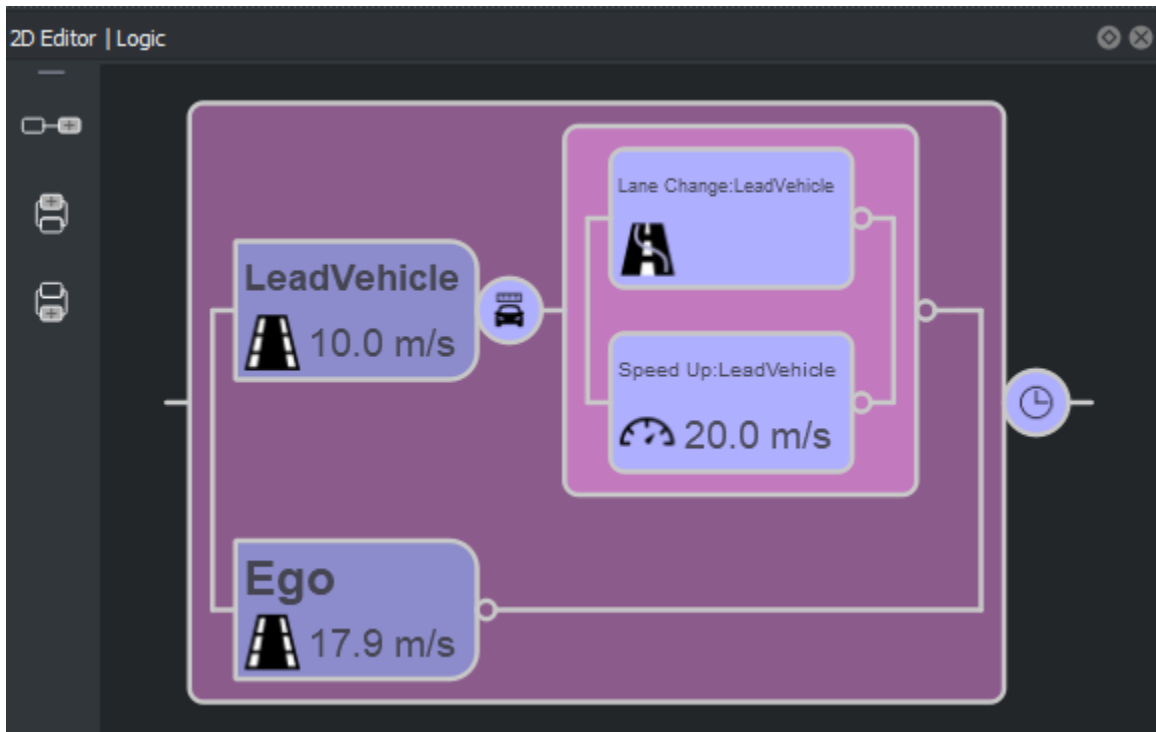
RoadRunner Scenario uses an anchoring system in which the actors and path waypoints are positioned relative to anchor points. By moving anchor points, you can quickly relocate a scenario within a scene or relocate a scenario to an entirely new scene.



Any vehicle or path waypoint can act as an anchor. For more details about anchoring, see Scenario Anchoring System.

### Define Scenario Logic

RoadRunner Scenario provides a graphical interface for defining the logic of a scenario. This graphical **Logic** editor is available from the **2D Editor** pane.

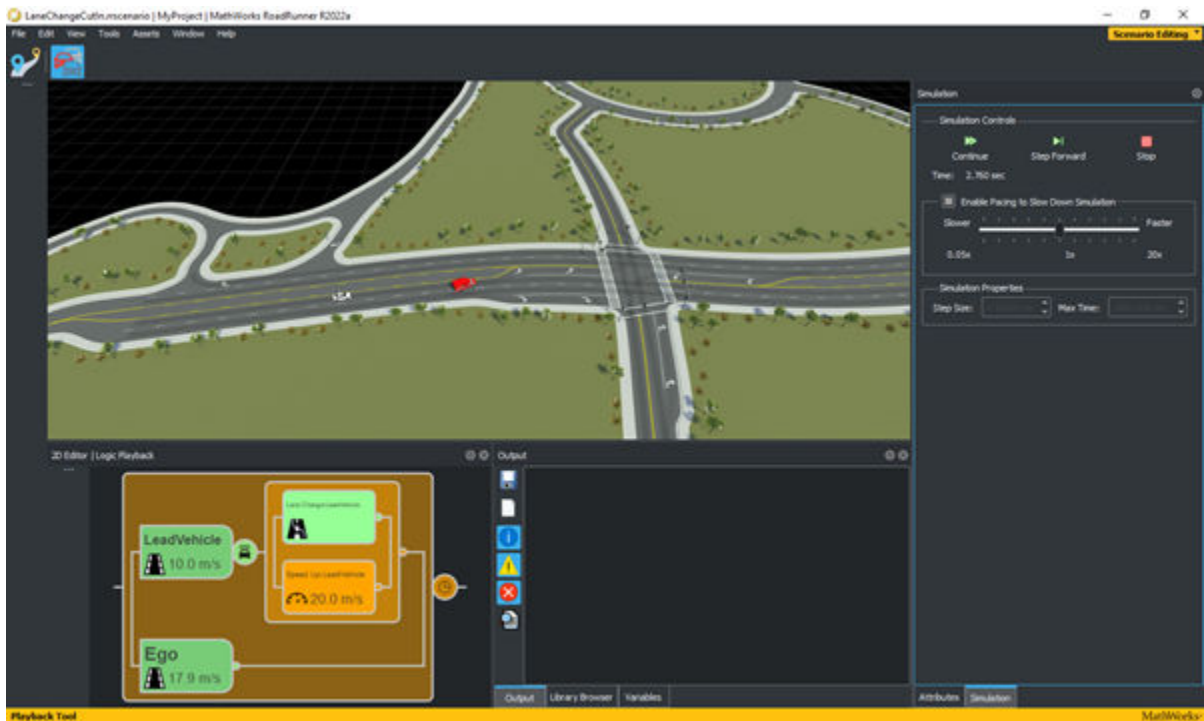


For details on all the actions and conditions you can set, see [Define Scenario Logic](#).

By default, vehicles have built-in behavior that enables them to follow lanes, change lanes, and swerve within a lane. For more details on these behaviors, see [Built-In Behavior for Vehicles](#).

### Simulate Scenarios

Use the **Simulation Tool** to simulate scenarios in the editing canvas. Using the controls in the **Simulation** pane, you can control the pacing and step size of the scenarios.



## Validate Scenarios

During the design, simulation, and export processes, RoadRunner Scenario provides validation feedback on your scenarios. This feedback comes in the form of visual indicators reports in the **Output** pane after simulation and upon export. Use this validation feedback to debug and resolve validation errors early. For more details, see [Validate Scenarios](#).

## ASAM OpenSCENARIO Support: Import and export scenarios using ASAM OpenSCENARIO file format

You can import scenarios with vehicle behaviors specified as predefined trajectories in ASAM OpenSCENARIO V1.0 into RoadRunner Scenario. You can modify these scenarios in the scenario editor. For more details on importing, see [Import Trajectories from ASAM OpenSCENARIO Files](#).

You can also export scenarios to ASAM OpenSCENARIO V1.x and V2.0. You can use these exported scenarios in other driving simulators and players. For more details on exporting scenarios, see [Export to ASAM OpenSCENARIO](#).

## Programmatic Scenario Interfaces: Generate scenario variations, import and export scenarios, and simulate scenarios using MATLAB functions or a gRPC API

RoadRunner Scenario provides programmatic interfaces for performing common workflow tasks, such as generating scenario variations, importing and exporting scenarios, and simulating scenarios.

<b>MATLAB Interface (Requires Automated Driving Toolbox)</b>	<b>External Programming Language Interface</b>
<p>Create a <code>roadrunner</code> (Automated Driving Toolbox) object to connect to an installed version of RoadRunner. Then call object functions to control RoadRunner programmatically.</p> <p>For more details on using these functions, see MATLAB Functions for Scenarios.</p>	<p>Compile the gRPC API into any language supported by gRPC, such as C++ or Python®. Then, call the compiled API from your language-specific clients. You can also use a precompiled version of the API to control RoadRunner from the command line.</p> <p>For more details on using the API, see gRPC API for Scenarios.</p>

You can modify scenario variables to create hundreds of scenario variations. For example, you can vary the distances between vehicles, or vehicle speeds, colors, or types. First, create a variable from an attribute in the application, then modify variables in the app or by using the RoadRunner gRPC API or MATLAB functions.

You can export multiple variations of a scenario programmatically. If you have multiple scenarios in one project, you can export them programmatically to speed up the export process.

You can import scenarios into different scenes programmatically. By loading a scenario into multiple scenes, you can test how your driving algorithm handles specific driving scenarios under varying conditions.

## Simulate RoadRunner scenarios with actors modeled in MATLAB and Simulink

Automated Driving Toolbox provides a simulation framework for simulating scenarios in RoadRunner with actors modeled in MATLAB and Simulink. For more information, see [Overview of Simulating RoadRunner Scenarios with MATLAB and Simulink \(Automated Driving Toolbox\)](#).

These examples demonstrate how to control ego vehicle behavior in RoadRunner simulations using MATLAB and Simulink:

- The Speed Action Follower with RoadRunner Scenario (Automated Driving Toolbox) example shows how to design speed action following behavior using MATLAB. You assign this behavior to the ego vehicle in the RoadRunner scenario and control the speed of the ego vehicle to avoid collision with a lead car. The example also shows how to visualize RoadRunner Scenario simulation data using MATLAB.
- The Trajectory Follower with RoadRunner Scenario (Automated Driving Toolbox) example shows how to control the motion of the ego vehicle in RoadRunner Scenario using Simulink to follow the specified trajectory. The example uses the Stanley controller and 3DOF vehicle dynamics to control the motion of the ego vehicle. The example also shows how to visualize RoadRunner Scenario simulation data using MATLAB.
- The Highway Lane Change Planner with RoadRunner Scenario (Automated Driving Toolbox) example shows how to simulate a lane change behavior for the ego vehicle in a RoadRunner scenario by using Simulink. The example uses a highway lane change planner Simulink model that finds an optimal collision-free trajectory to navigate the ego vehicle.



---

## **CARLA Actor Cosimulation: Visualize actors designed in CARLA within RoadRunner Scenario using the cosimulation platform**

RoadRunner Scenario and CARLA cosimulation technology enables the RoadRunner Scenario system to control an ego vehicle in a scene, with easy state and behavior control, while the CARLA software engine manages other vehicles in the scene. This cosimulation combines the ease of development and design provided by RoadRunner Scenario with the scalability provided by CARLA. For more information, see [Cosimulate Actors with CARLA](#).

