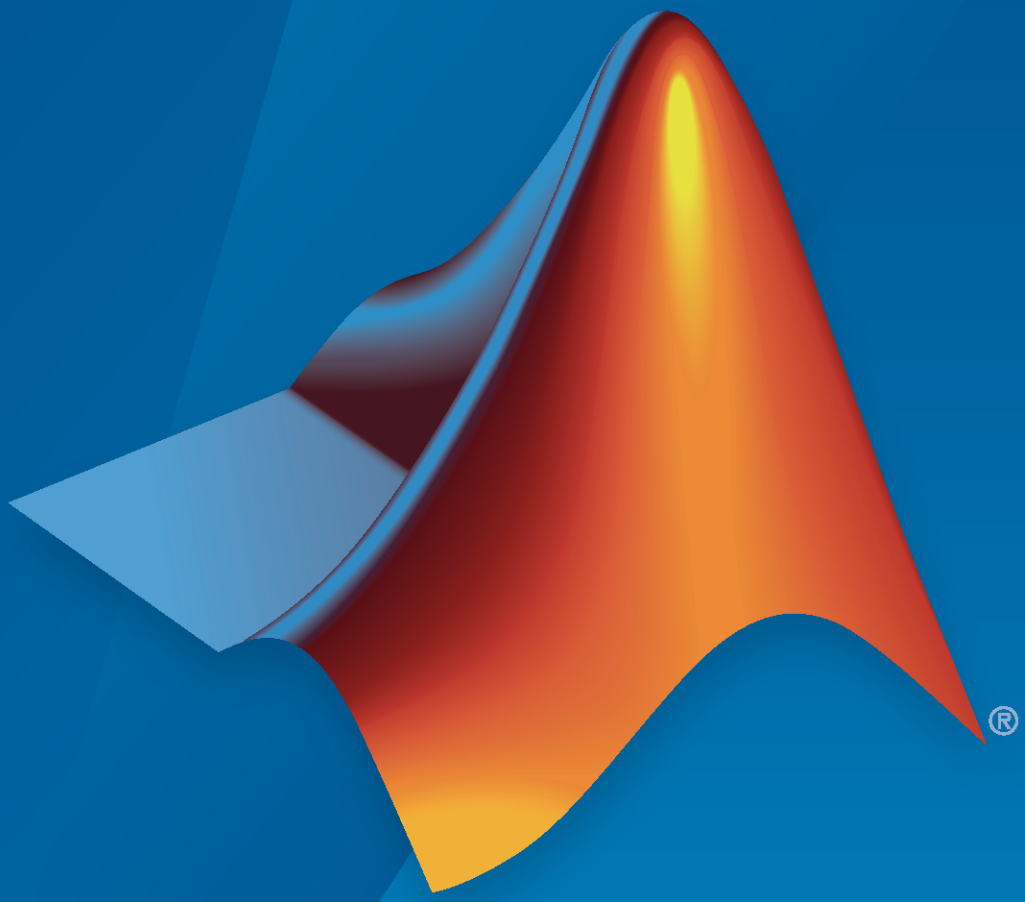
RoadRunner Scenario Release Notes



MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `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

Contents

R2023a

R2022b

R2022a

R2023a

Version: 1.2

New Features

Bug Fixes

R2023a General Release

Sensor Simulation for RoadRunner Scenario: Simulate RoadRunner Scenario with sensor models defined in MATLAB

Use the new `SensorSimulation` object to view and control the sensors in a RoadRunner scenario. Configure the sensor models in MATLAB®, and then use the `addSensors` function to add them to vehicles in your RoadRunner scenario. You can then obtain ground truth measurements by using the `targetPoses` and `laneBoundaries` functions, and process them to generate detections. You can also visualize the detections by using the `birdsEyePlot` object. The `SensorSimulation` object supports these sensor model System objects:

- `visionDetectionGenerator`
- `ultrasonicDetectionGenerator`
- `drivingRadarDataGenerator`

Sensor Simulation for RoadRunner Scenario: Simulate RoadRunner Scenario with sensor models defined in Simulink

Use the new `SensorSimulation` object to view and control the sensors in a RoadRunner scenario. You can now add sensor model blocks to a Simulink® model that defines custom behavior for a RoadRunner actor. To add the sensors to vehicles in your RoadRunner scenario, use the `addSensors` function and specify the block paths to the sensor models. You can then obtain ground truth measurements by using the RoadRunner Scenario Reader block, and process them to generate detections. The `SensorSimulation` object supports these sensor model blocks:

- Vision Detection Generator
- Ultrasonic Detection Generator
- Driving Radar Data Generator

Offline Playback of RoadRunner Scenario Simulation: Replay simulation from saved file

In R2023a, you can save a RoadRunner Scenario simulation log into a file and replay the corresponding simulation from the file at any time. The simulation playback runs in the same Roadrunner Scenario environment as the original simulation and does not involve any computation from an associated cosimulation client like MATLAB or Simulink.

Save the scenario log of a `Simulink.ScenarioSimulation` simulation into a file using the `save` command. The `set` command allows you to replay the simulation.

You can inspect the saved simulation log file from which a scenario is replayed using the `load` command.

User Defined Events: Use custom events to control a RoadRunner Scenario simulation or enable communication between actors

Starting in R2023a, you can create custom events in MATLAB and Simulink. An actor dispatches a custom event, carrying some data, to RoadRunner Scenario to control a simulation in a specific way,

or to communicate with other actors. For example, you can arrange for all vehicles in a simulation to stop at the same time using event parameters. User-defined events can be sent from an actor to a scenario, or from an actor to all other actors in a scenario.

In MATLAB, use `sendEvent` and `receiveEvent` functions in the System object™ implementation of an actor model to send and receive events from a scenario.

The Simulink blocks RoadRunner Scenario Reader and RoadRunner Scenario Writer can now send and receive events from a RoadRunner Scenario simulation.

For more information, see “Design Vehicle Following User-Defined Events Scenario”.

Additional Data Type Support for User Defined Actions Created in MATLAB and Simulink

In R2023a, RoadRunner Scenario now supports `string`, `double`, `uint16`, `int32`, `uint32`, and `boolean` data types for user-defined action parameters created in MATLAB and Simulink. Previously, RoadRunner Scenario supported only the `string` data type for user-defined action parameters.

User-defined action parameters defined in MATLAB or Simulink and imported into RoadRunner Scenario become initial values for their action asset and must have a data type supported by RoadRunner Scenario.

You cannot edit the initial values of imported parameter assets within RoadRunner Scenario. To change the initial values of imported parameters, you must edit them in MATLAB or Simulink. You can still use actions phases in the **Logic** editor to alter the values of imported parameters over the course of the simulation.

Prop Reference Actors: Use prop models to design scenario logic

You can convert the prop models in a scenario into prop reference actors and use them to design scenario logic. To create a prop reference actor, in **Scenario Editing** mode, select the Prop Reference Creation Tool. Next, select the prop model in the scenario you want to convert to a prop reference actor, then, in the left toolbar, select Create Prop Reference.

Prop reference actors do not have initial action phases or default scenario logic and you cannot assign scenario logic to them. However, you can use prop reference actors to build your scenario logic by referencing them in the action phases and conditions of vehicle or pedestrian actors. For more information, see “Actors in RoadRunner Scenario”.

Application Examples: Simulate truck platooning, autonomous emergency braking, and lane-level path-planning applications

The “Truck Platooning with RoadRunner Scenario” (Automated Driving Toolbox) example shows how to cosimulate a platooning application with RoadRunner Scenario and Simulink. The platooning system contains vehicle-to-vehicle (V2V) communication, tractor-trailer dynamics, longitudinal controller, and lateral controller components. The lateral controller enables you to test the platooning application on curved road scenarios.

The “Autonomous Emergency Braking with High-Fidelity Vehicle Dynamics” (Automated Driving Toolbox) example shows how to design an autonomous emergency braking (AEB) system in Simulink

and cosimulate the system with RoadRunner Scenario using a scene that contains elevated roads. This example models an AEB system using high-fidelity vehicle dynamics with 14 degrees-of-freedom and uses camera, radar, and terrain sensors in the 3D simulation environment.

The “Lane-Level Path Planning with RoadRunner Scenario” (Automated Driving Toolbox) example shows how to plan a vehicle path between the specified start and goal points using MATLAB and follow the planned path in RoadRunner Scenario.

Reverse Motion: Simulate reverse motion of actor in lane-following mode

You can now simulate reverse motion of an actor in the lane-following mode. To configure the reverse motion for an actor, specify a negative speed value for the `Change Speed` action. For more information, see “Reverse Motion Along Lane”.

R2022b

R2022b Update 5

Remove Actor Actions: Remove actors from the scenario during run time

The `Remove Actor` action removes an actor from the scenario during simulation run time. The rest of the simulation continues until it reaches a specified simulation end condition. For more information, see Remove Actor Actions.

You can export `Remove Actor` action to the ASAM OpenSCENARIO® 1.x and 2.0 file formats. For more information, see Remove Actor Action (1.x) and Remove Actor Action (2.0).

Add Action Phase Before: Add action phases prior to existing phases in the logic editor

As of R2022b Update 5, you can add action phases that precede existing phases in the **Logic** editor, including the initial phase for each actor. To add an action phase before an existing phase, right-click an action phase in the **Logic** editor, then, in the context menu, select **Add Action Phase Before**. By default, RoadRunner Scenario creates a `Wait` action with a `Duration` condition of `5.00s`.

If you add an action phase before the initial phase of an actor, the default `Wait` action and accompanying `Duration` condition delay the initialization of the actor by 5 seconds during simulation.

The first action phase for an actor must be an initial action phase or a `Wait` action. If the first action phase is a `Wait` action, the next action phase must be the initial action phase. Otherwise, the simulation stops and RoadRunner Scenario displays an error in the **Output** pane. To change an action phase to an initial phase, right-click the action phase in the **Logic** editor and select **Set as Initial Phase**.

Additional Actor Type Support for Action Phases and Conditions: Use any action phase with pedestrian actors, set pedestrians as reference actors

Previously, RoadRunner Scenario supported only vehicle actors as reference actors. As of R2022b Update 5 you can use pedestrian actors as reference actors for action phases and conditions that have a **Reference Actor** field. To add an actor as a reference actor for an action phase or condition, select the action phase or condition in the **Logic** editor. Then, in the **Attributes** pane, set **Reference Actor** to an actor in your scenario.

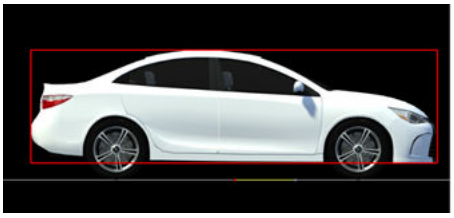
Additionally, you can now use any action phase type when building scenario logic for pedestrian actors. However, because pedestrian actors require paths, if you select an action type that uses lane data, such as the `Change Longitudinal Distance` or `Change Lane` action, RoadRunner Scenario displays a warning in the **Attributes** pane: "*Action* action cannot be used when actor follows path". *Action* is the name of the selected action type. This is because action types that use lane data do not support path-following actors. For more information about adding action phases to actors and action phase types, see Define Scenario Logic.

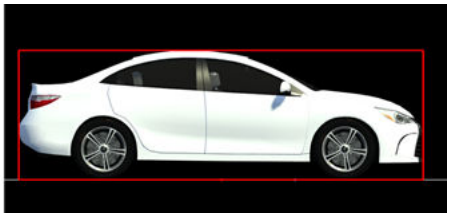
Functionality being removed or changed

Changes to Vehicle Bounding Boxes: Bounding boxes of vehicle actors now include the tires
Behavior change

Previously, the vehicle bounding boxes only surrounded the bodies of vehicle actors. As of R2022b Update 5, the bounding box, which determines collision for actors in the scenario, now includes the tires of vehicle actors. This change affects only the height of the bounding box.

These images show the difference between the old and new vehicle bounding boxes.





If you load a project created before R2022b Update 5 into this version of RoadRunner Scenario, the actors automatically update to use the new bounding boxes. Note that this change to vehicle bounding box dimensions can cause collisions during simulation where there previously were not any.

The new bounding box dimensions are also reflected in vehicle actors exported to ASAM OpenSCENARIO formats.

R2022b Update 3

Time to Actor Condition: Control actor behavior based on the time to reach the relative position of a reference actor

The time to actor condition calculates the time to the reference actor by taking the longitudinal distance between the two actors and dividing it by the relative velocity of one actor compared to the other. If you set **Condition** to `Time To Actor`, then the actor performs the previous action until it is a certain number of seconds from reaching the relative position of another actor. For more information, see Time to Actor Condition.

Additional Speed Comparisons for Actor Speed Condition: Set rules for comparing actor speed

As of R2022b, you can specify additional speed comparisons when using the `Actor Speed` condition: `Equal to`, `Greater or equal`, `Greater than`, `Less or equal`, `Less than`, and `Not equal to`.

When you set **Relative to** to `Absolute`, RoadRunner Scenario uses the method specified by **Rule** to compare the actor speed to the specified value of **Speed**. When you set **Relative to** to `Actor`, RoadRunner Scenario uses the method specified by **Rule** to compare the actor speed to the speed defined by the **Reference Actor**, **Direction**, and **Speed Offset** attributes.

Default Color: Specify default color for vehicle assets

As of R2022b, vehicle assets now have a **Default Color** attribute that you can change, in the **Attribute** pane of the vehicle asset, to specify the color for all vehicle assets of that type when placed in a scene or scenario. Some vehicle assets with initial default colors that are not white, like the `Ambulance` or `GarbageTruck` assets, do not visually reflect any changes to their default color. For more information on default color limitations, see the Limitations section of the Vehicle Assets page.

Reverse Motion: Simulate reverse motion of actor in path-following mode

Starting from R2022b, you can simulate reverse motion of an actor in the path-following mode. To configure the reverse motion for an actor, specify a negative speed value for the `Change Speed` action. For more information, see Path-Following Behavior and Negative Vehicle Speed.

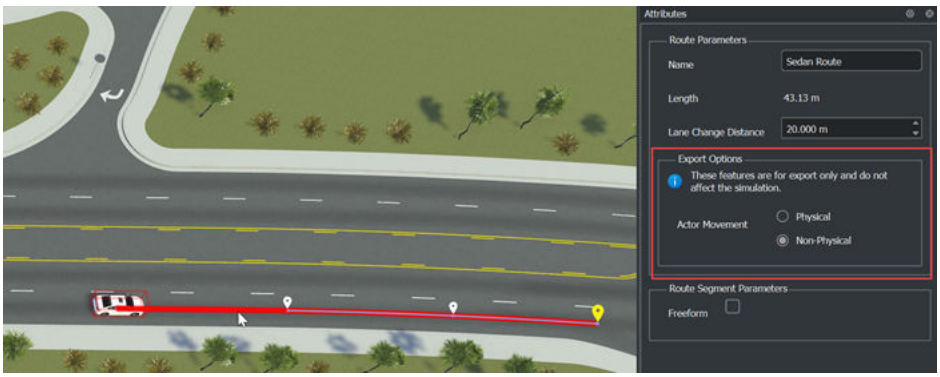
Bidirectional Lanes: Simulate lanes that support simultaneous travel in opposing directions

With bidirectional lanes, you can simulate lanes that support simultaneous travel in opposing directions. To create bidirectional lanes, you must activate **Scene Editing** mode and use the Lane Tool. To learn more about the Lane Tool and road and lane editing in RoadRunner, see Lane Tool and Roads, Lanes, and Markings.

Using **Scenario Editing** mode, you can place a vehicle directly in the bidirectional lane, or use a `Change Lane` action phase to simulate the vehicle changing from a unidirectional lane to a bidirectional lane. For more information, see Lane and Actor Direction in Scenarios.

Export Options for Paths: Specify physical and non-physical actor movement on paths

When exporting a path to the ASAM OpenSCENARIO format, you can specify whether the actor following the path does so with physical or non-physical motion. First, select a path segment, and then, in the **Attributes** pane, in the **Export Options** section, set **Actor Movement** to **Physical** or **Non-Physical**.



The **Actor Movement** setting does not affect the simulation in RoadRunner Scenario. Physical and non-physical motion attributes on paths affect only files exported to the ASAM OpenSCENARIO format, which you can use in simulators that consider the dynamic constraints of the actor.

GetAllScenarioVariables Method: Retrieve All Variables in a Scenario Programmatically

The `GetAllScenarioVariables` method retrieves all variable names and values in the current scenario. For information, see Generate Scenario Variations Using gRPC API.

R2022b General Release

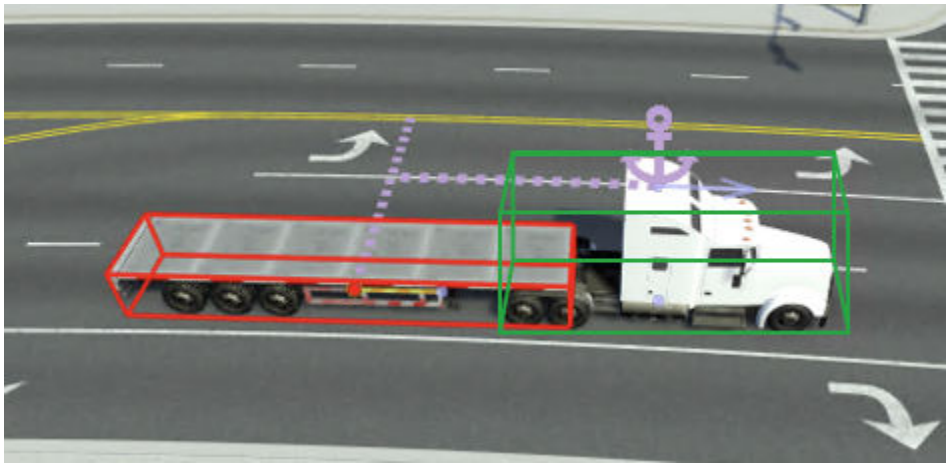
Pedestrian Actors: Design scenarios with pedestrian actors

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 standard pedestrian actor, named `Citizen_Male`, with its own skeleton, mesh, 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` (Automated Driving Toolbox) object function of a `Simulink.ActorSimulation` (Automated Driving Toolbox) 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` (Automated Driving Toolbox), or Simulink blocks, such as RoadRunner Scenario Reader (Automated Driving Toolbox) and RoadRunner Scenario Writer (Automated Driving Toolbox), to retrieve and process user-defined actions.

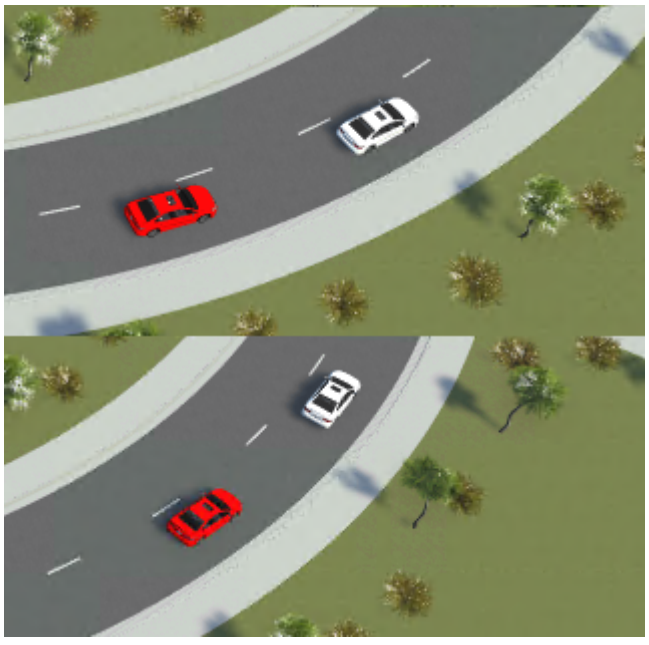
For more information about processing user-defined actions in MATLAB or Simulink, see Design Vehicle Following User-Defined Actions Scenario.

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.

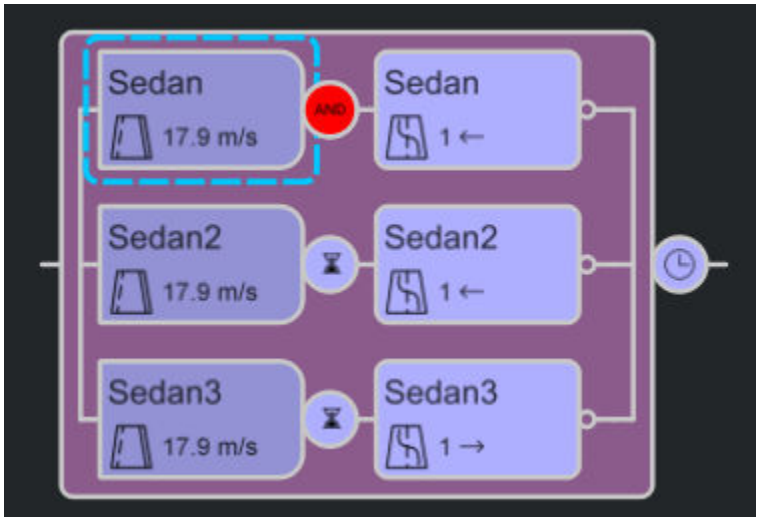
For example, if you want a vehicle to perform a lane change action when it reaches a longitudinal distance of less than 10 meters behind the reference vehicle, you can use a `Longitudinal Distance to Actor` condition with a **Relative Position** of `Behind`, a **Rule** of `Less than` and a **Threshold** 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.

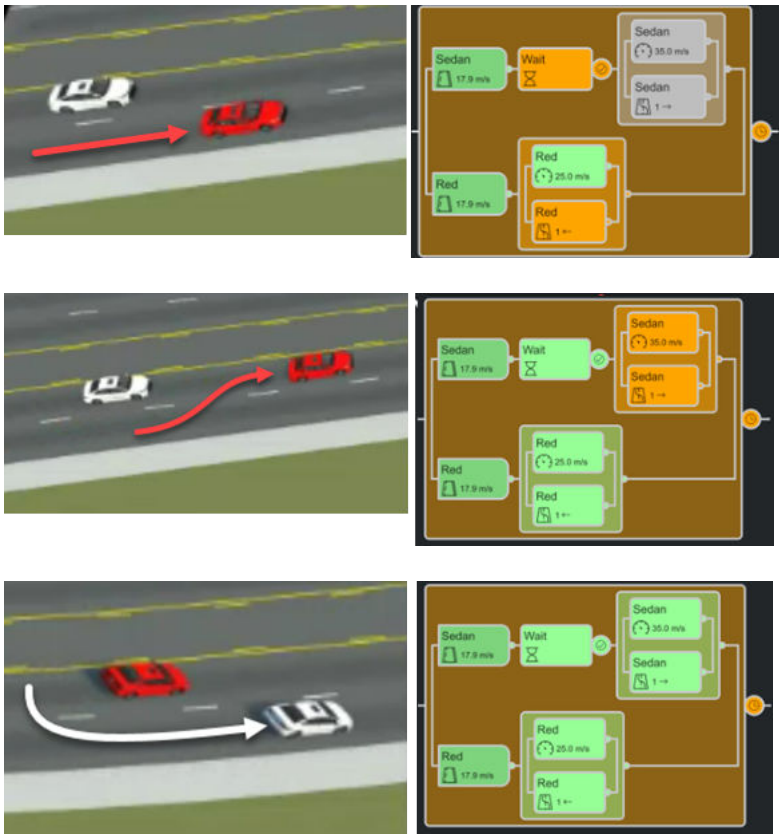


Phase State Conditions: Check the runtime state of a phase

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.

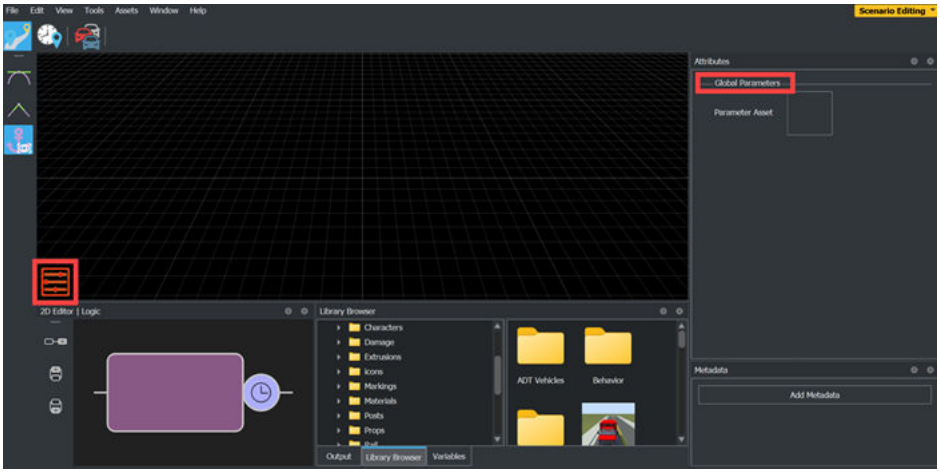
In this example, the white sedan has parallel `Change Lane` and `Change Speed` actions after a `Wait` action with a phase state condition linked to the `Change Lane` and `Change Speed` actions of the red sedan. As a result, the white sedan waits for the red sedan to finish its lane transition, then changes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

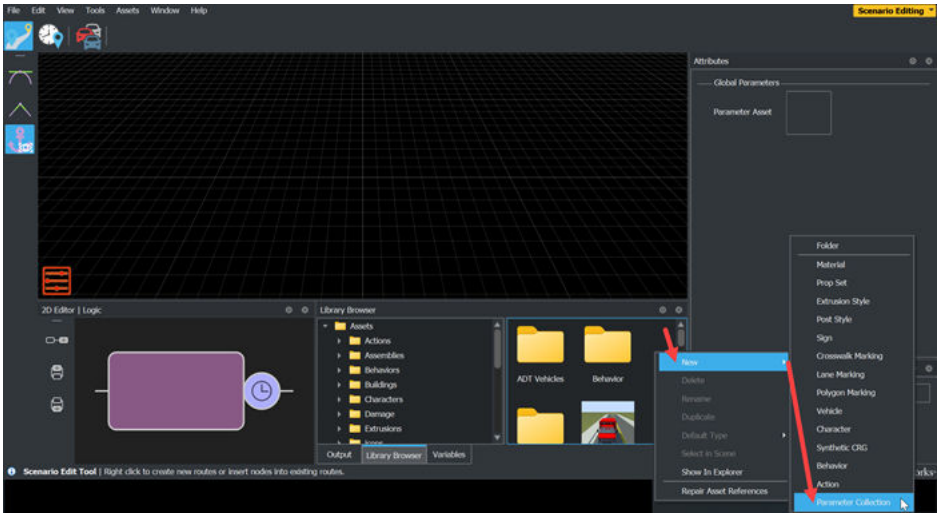


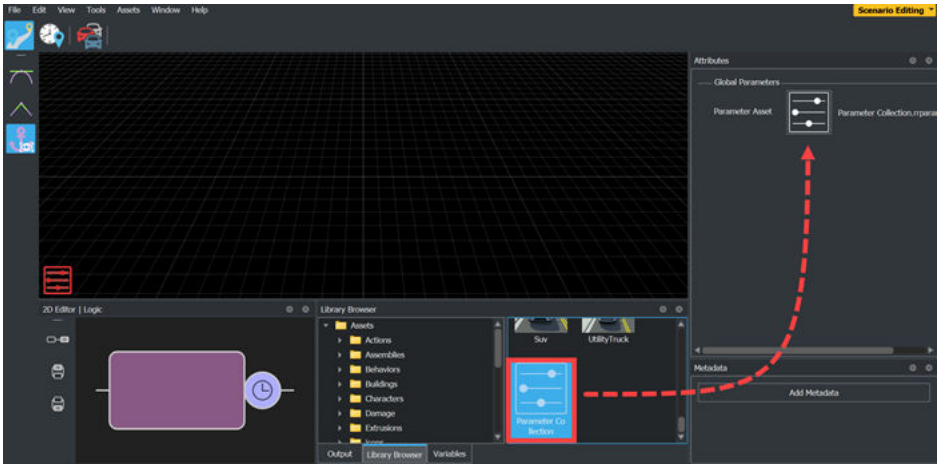
Global Parameters: Adjust conditions for the entire scenario

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 and time of day.

To create a new global parameter, right-click an empty space in the right pane of the **Library Browser**, select **New**, and then click **Parameter Collection**. A parameter collection is a group of parameters that you can use in multiple scenarios.







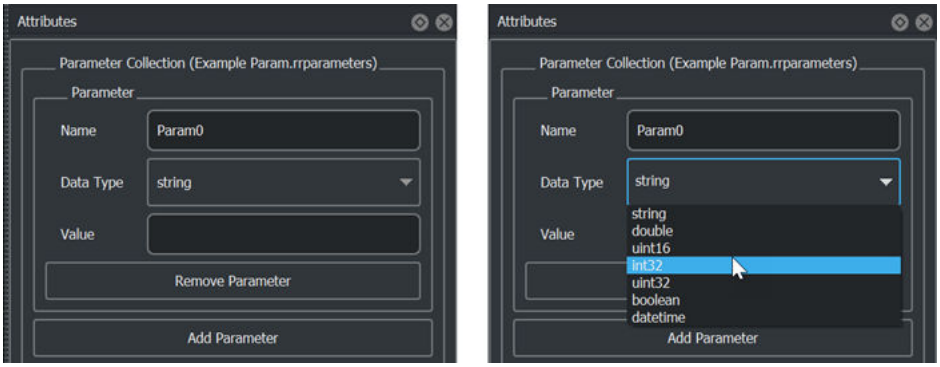
To assign a parameter collection to a scenario, select the global parameters icon to access the **Attributes** pane for **Global Parameters** and drag the parameter collection asset from the **Library Browser** to the **Parameter Asset** field. By default, RoadRunner Scenario references global parameters from the associated parameter collection asset. To initialize parameter values, in the

global parameters **Attributes** pane, under **Global Parameters**, select the check boxes next to the corresponding parameter names.

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, select `Global Parameter` from the **Add Condition** drop-down. You can combine behavior and global parameters, as well as parameter actions and conditions, to design your scenario logic.

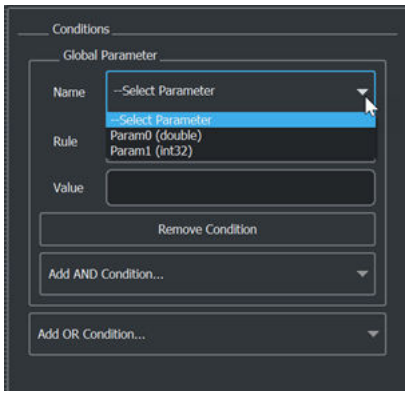
Parameter Data Types: Specify the data type of parameter values

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.

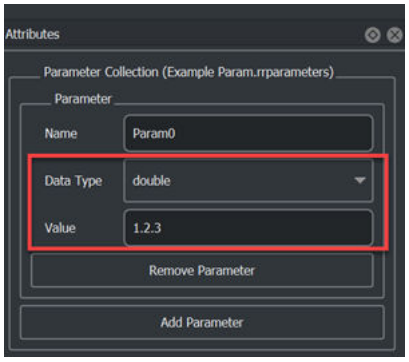


To define the data type for a parameter, select a parameter asset in the **Library Browser** and then, in the **Attributes** pane, select a data type from the **Data Type** list. You can then specify the value of this parameter in the **Value** field. Note that a parameter collection can contain parameters 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.



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:

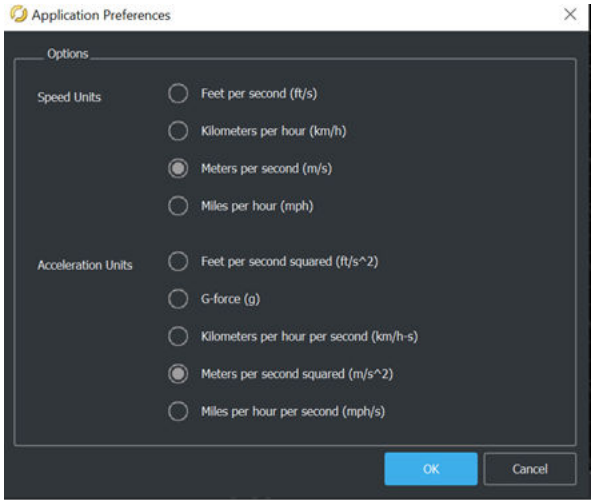
- **Feet per second (ft/s)**
- **Kilometers per hour (km/h)**
- **Meters per second (m/s)**
- **Miles per hour (mph)**

You can select from these units of acceleration:

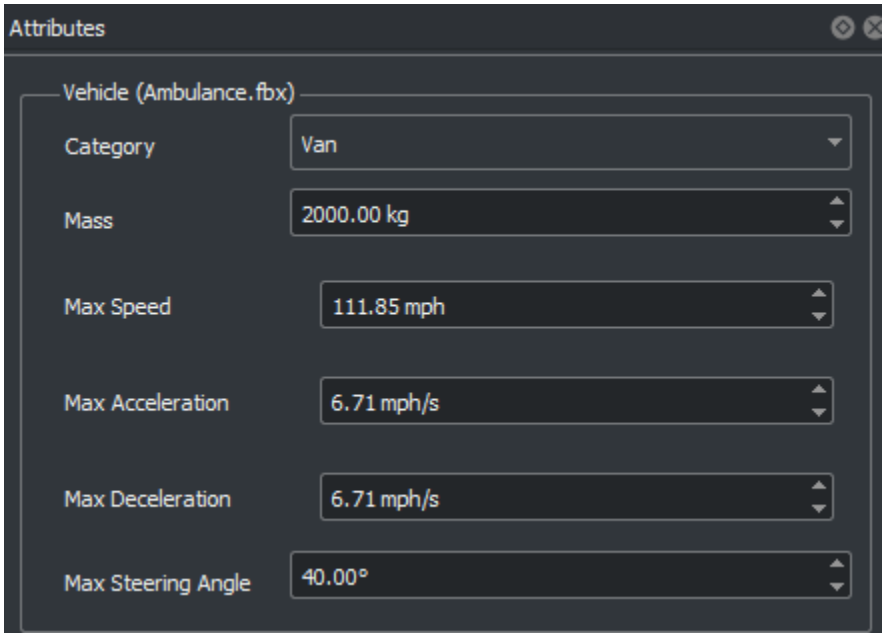
- **Feet per second squared (ft/s^2)**
- **g-force (g)**
- **Kilometers per hour per second (km/h-s)**
- **Meters per second squared (m/s^2)**

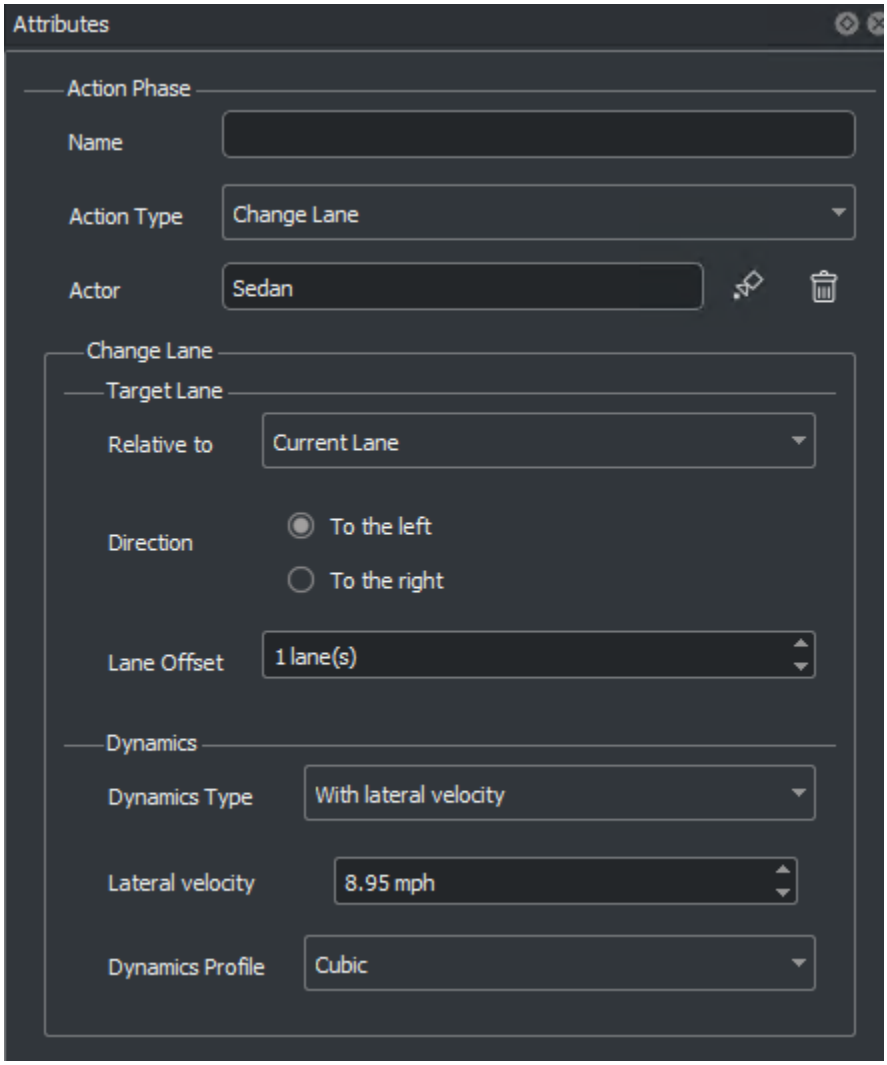
- **Miles per hour per second (mph/s)**

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` (Automated Driving Toolbox) MATLAB function or the `Import` method of the gRPC® 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 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.

Sample Scenarios: Open and explore sample files containing different scenario logic

As of R2022b, RoadRunner Scenario provides several prebuilt sample scenario files that you can open to observe different kinds of scenario logic and behavior. To learn more about the sample files and how to open them, see Open and Explore Sample Scenarios.

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

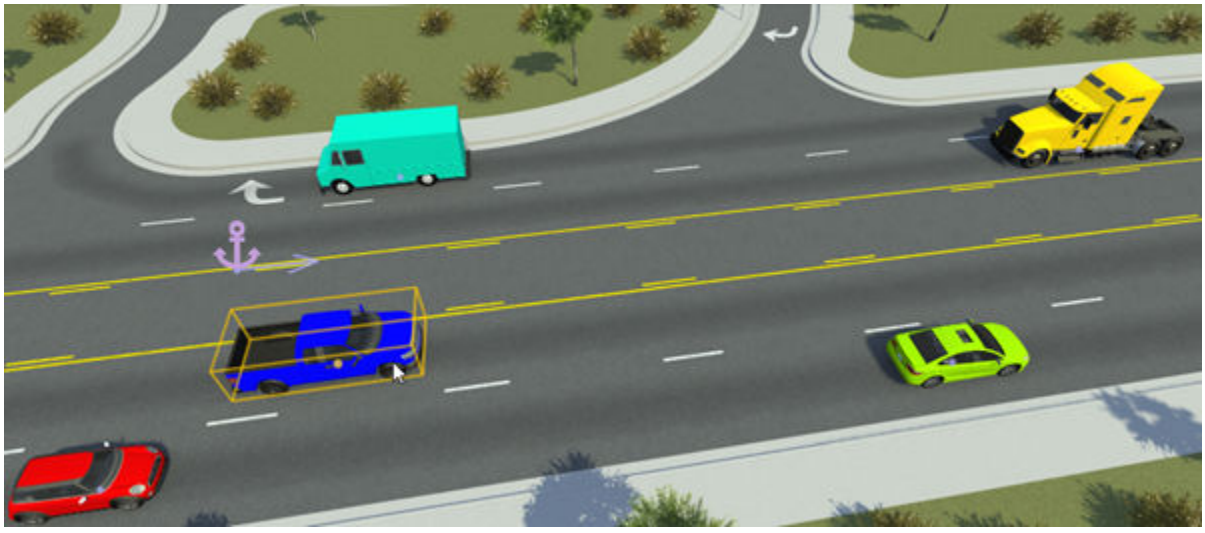
Use the RoadRunner Scenario interactive editor to modify imported scenarios or design your own scenarios based on scenario testing protocols such as ALKS or Euro NCAP®. To get started designing scenarios, see these examples:

- 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.

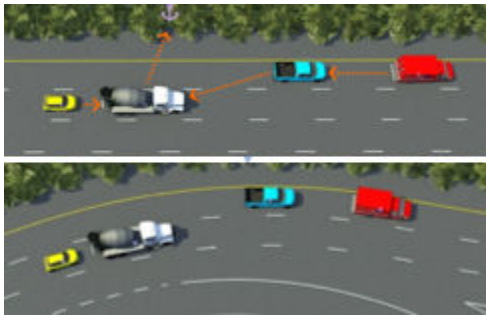
Specify Driving Paths

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

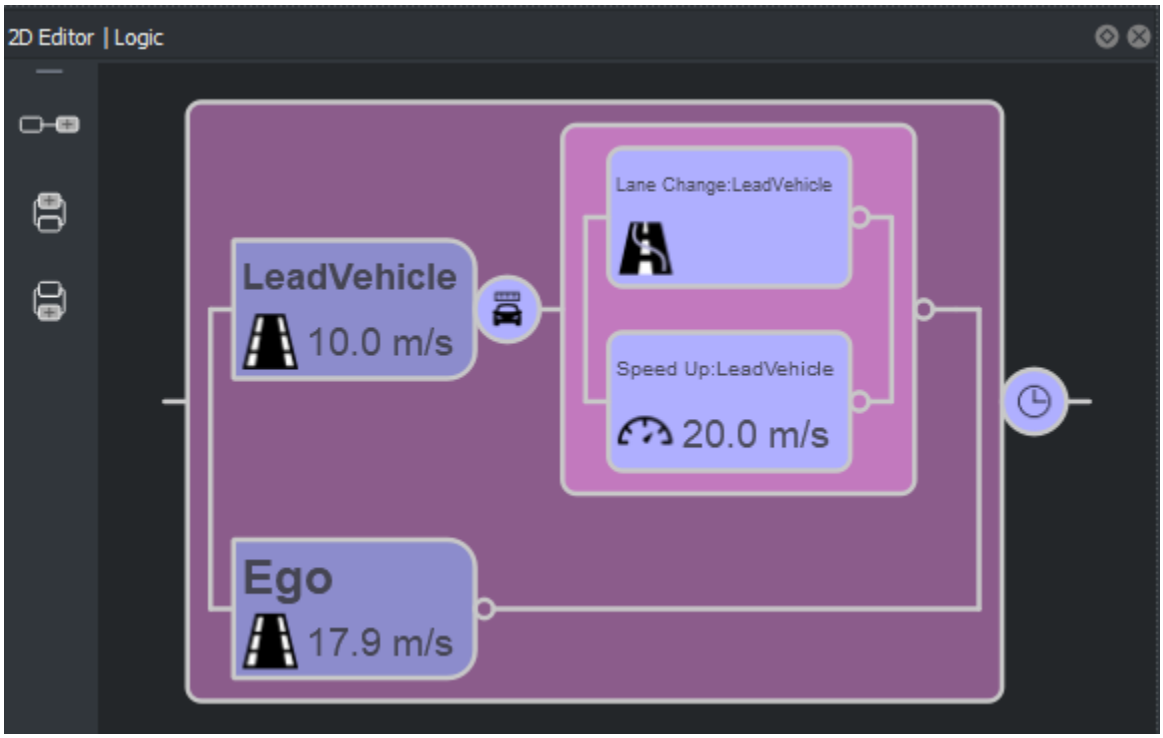
RoadRunner Scenario uses an anchoring system in which the actors and path waypoints are positioned relative to anchor points. By moving anchor points, you can quickly relocate a scenario within a scene or relocate a scenario to an entirely new scene.



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.

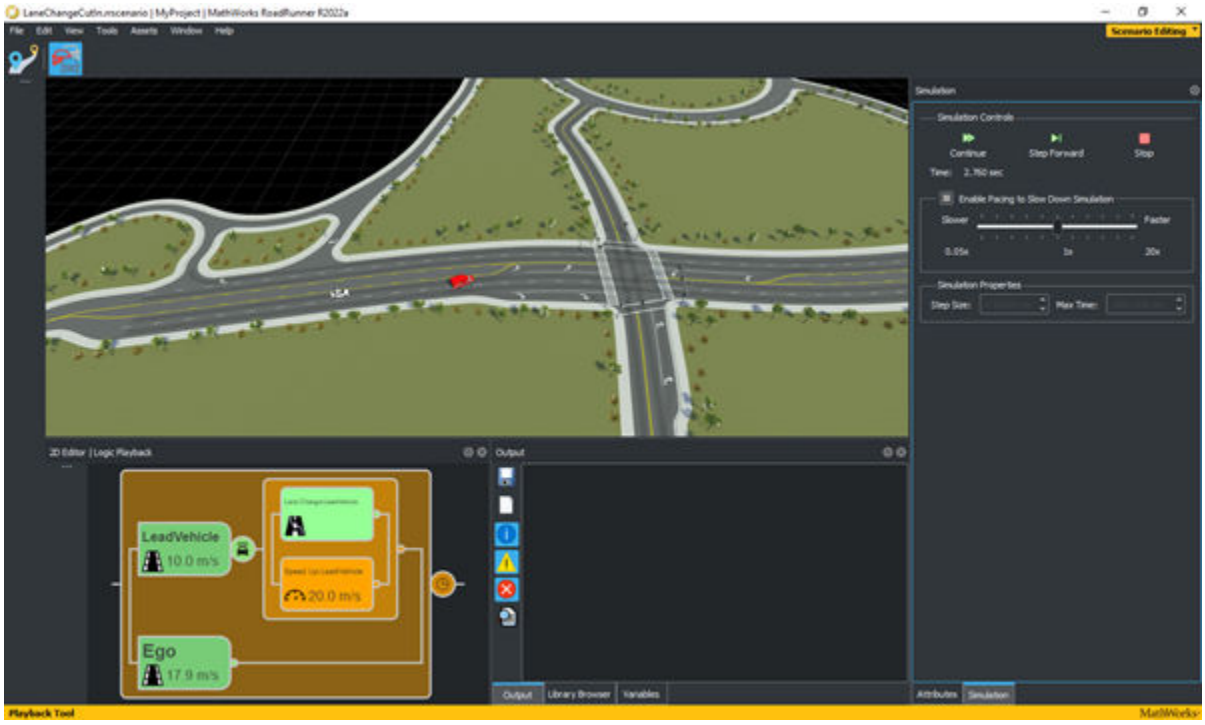


For details on all the actions and conditions you can set, see Define Scenario Logic.

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.

MATLAB Interface (Requires Automated Driving Toolbox)	External Programming Language Interface
Create a `roadrunner` (Automated Driving Toolbox) object to connect to an installed version of RoadRunner. Then call object functions to control RoadRunner programmatically. For more details on using these functions, see MATLAB Functions for Scenarios.	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. For more details on using the API, see gRPC API for Scenarios.

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.