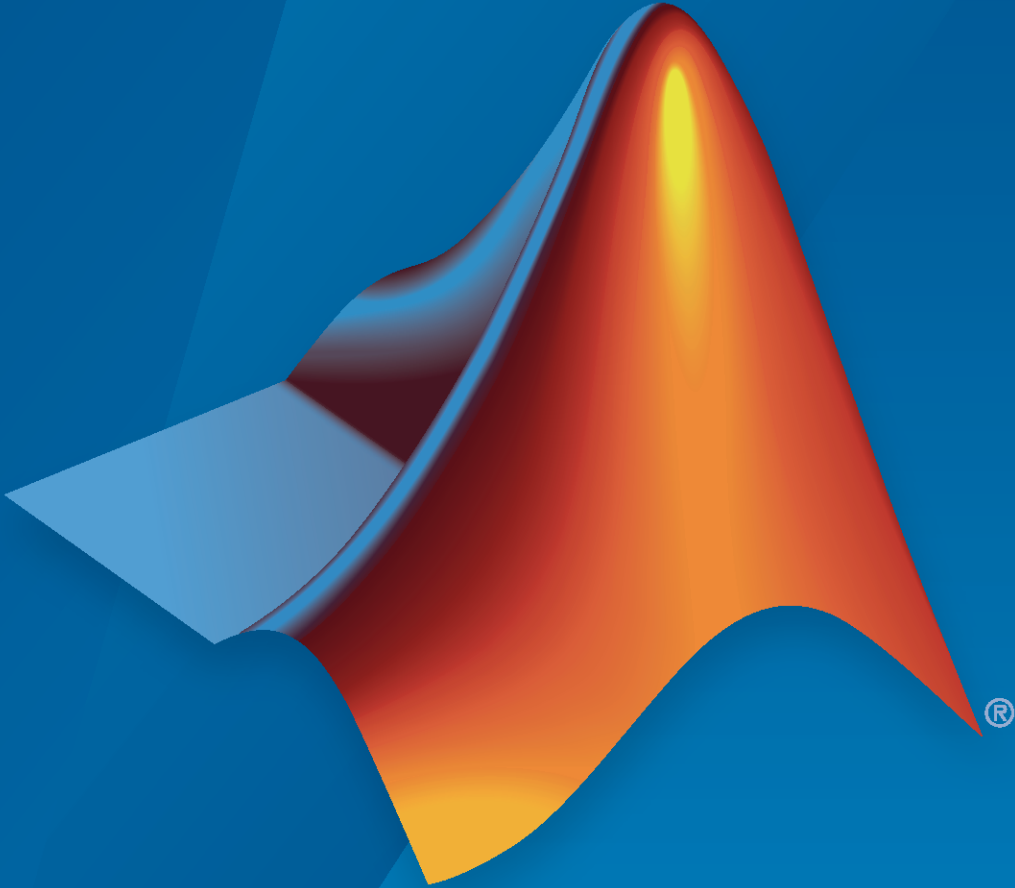


RoadRunner Scenario

Reference



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 Reference

© 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 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 for more information.

Revision History

March 2022	Online only	New for Version 1.0 (Release 2022a)
September 2022	Online only	Revised for Version 1.1 (Release 2022b)

1	Tools
2	Assets
3	Functions
4	Objects
5	Configurations

Tools

Scenario Edit Tool

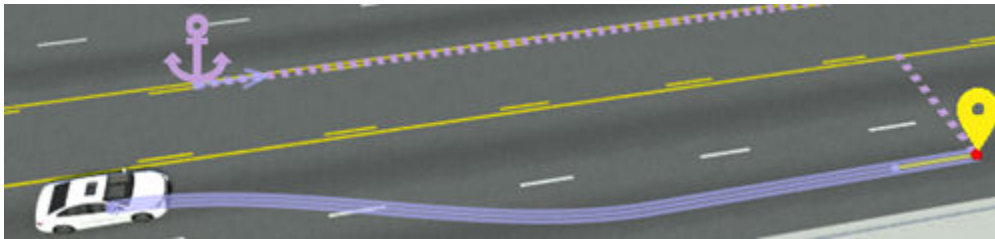
Add actors and paths, modify road anchors, and define scenario logic

Description

The **Scenario Edit Tool** enables the scenario and logic editing mode in RoadRunner Scenario. The **Scenario Edit Tool** is selected by default when you open RoadRunner Scenario.

The **Scenario Edit Tool** enables you to perform these actions:

- Add vehicles as actors to the scenario. For more details on vehicles, see **Vehicle Assets**.
- Add or edit paths for actors to follow. For more details, see “Path Editing”.
- Move or modify the road anchors that determine how actors derive their positions from the scene. For more details, see “Scenario Anchoring System”.
- Define the scenario logic for how actors interact with each other. For more details, see “Define Scenario Logic”.
- Assign behavior assets to actors to determine what agent model the actors use during simulation. For more details, see “Specify and Assign Actor Behaviors”.



Open the Scenario Edit Tool

On the RoadRunner Scenario toolbar, click the **Scenario Edit Tool** button:



Examples

- “Design Lane Following Scenario”
- “Design Lane Change Scenario”
- “Design Lane Swerve Scenario”
- “Design Path Following Scenario”

Version History

Introduced in R2022a

See Also

Simulation Tool | Vehicle Assets

Topics

“Design Lane Following Scenario”

“Design Lane Change Scenario”

“Design Lane Swerve Scenario”

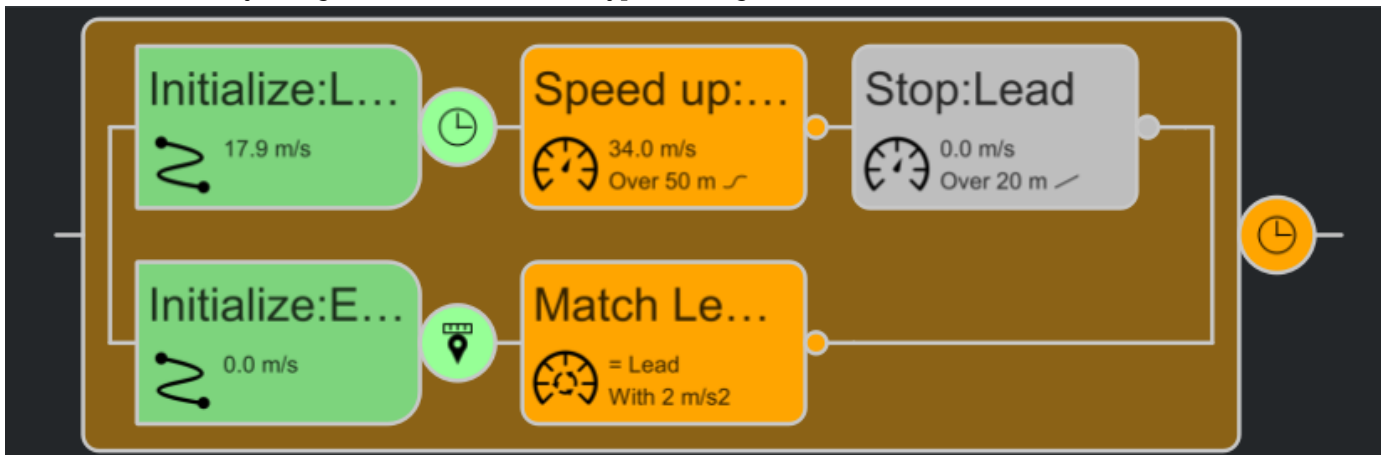
“Design Path Following Scenario”

Simulation Tool

Simulate scenario

Description

The **Simulation Tool** enables you to simulate the scenarios you design using RoadRunner Scenario. Using the controls in the **Simulation** pane, you can play back, pause, and restart a scenario. You can also control the pacing and step size of the simulation. You can navigate the scenario quickly and effectively using the different camera types during simulation.



Open the Simulation Tool

On the RoadRunner Scenario toolbar, click the **Simulation Tool** button:



When you click the **Simulation Tool** button, all actors and paths disappear from the scenario until you start the simulation by clicking the **Play** button.

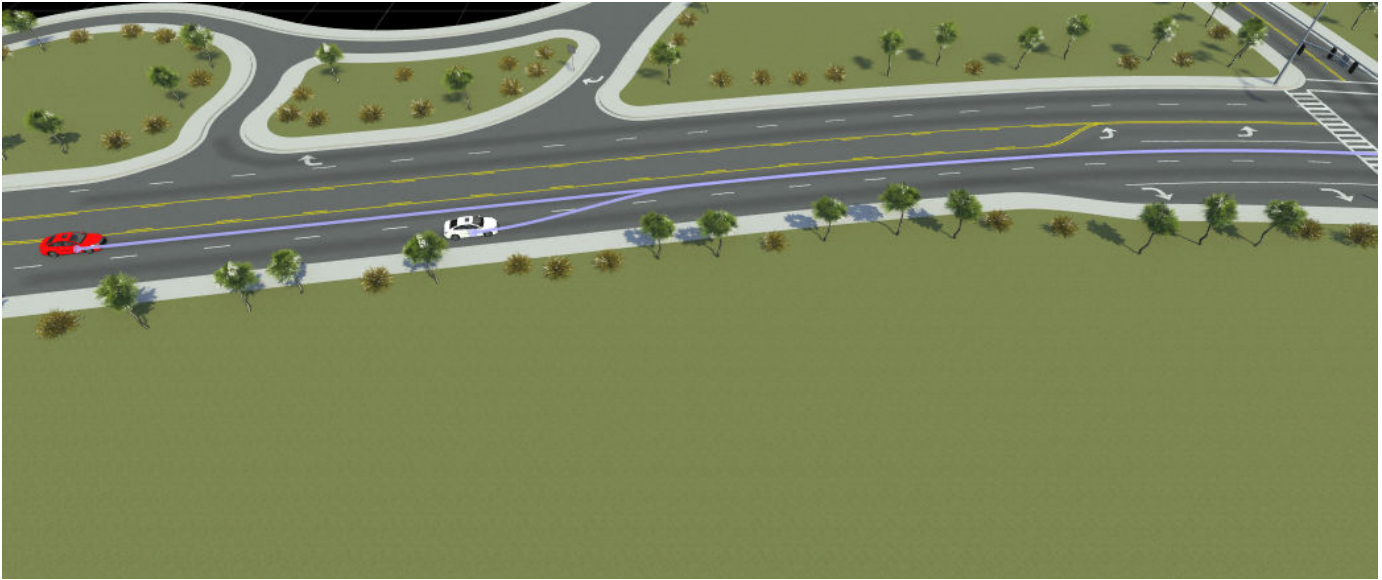
Examples

Simulate Scenario Using Simulation Tool

Open a prebuilt scenario, and then use the **Simulation Tool** to simulate the scenario.

From the **File** menu, select **Open Scenario**. Then, select the TrajectoryCutIn scenario, which is one of the prebuilt scenarios included by default in the Scenarios folder of RoadRunner Scenario.

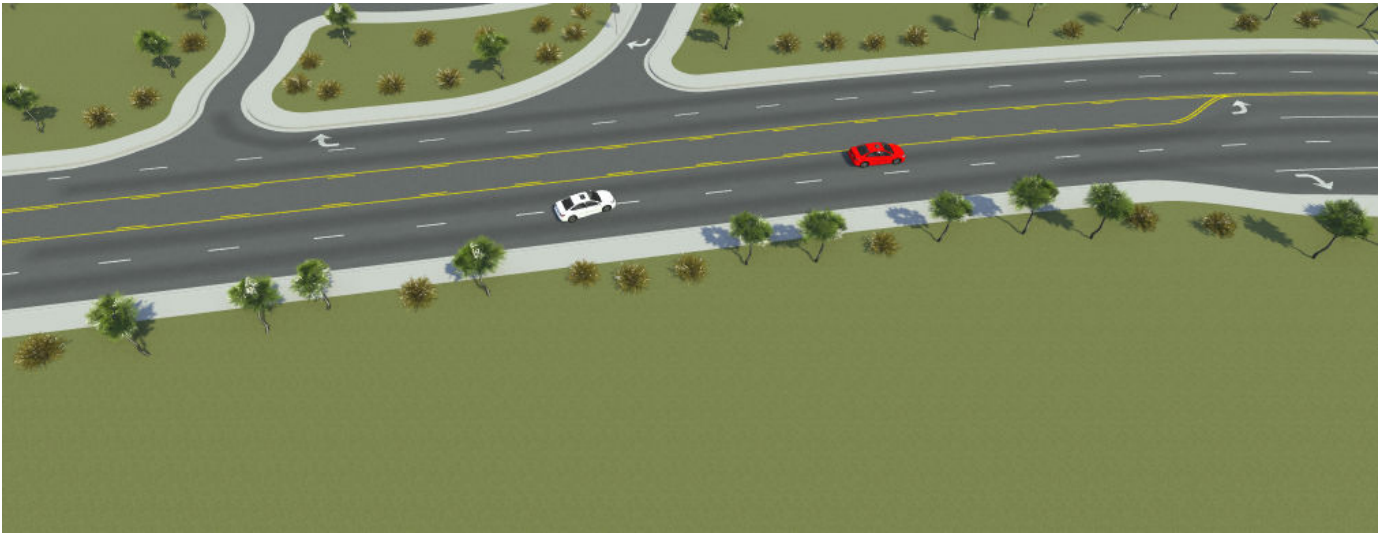
In this scenario, one vehicle cuts in front of another vehicle. The vehicles follow predefined paths.



On the RoadRunner Scenario toolbar, click the **Simulation Tool** .

In the **Simulation** pane, under **Simulation Controls** click **Play** to simulate the scenario.

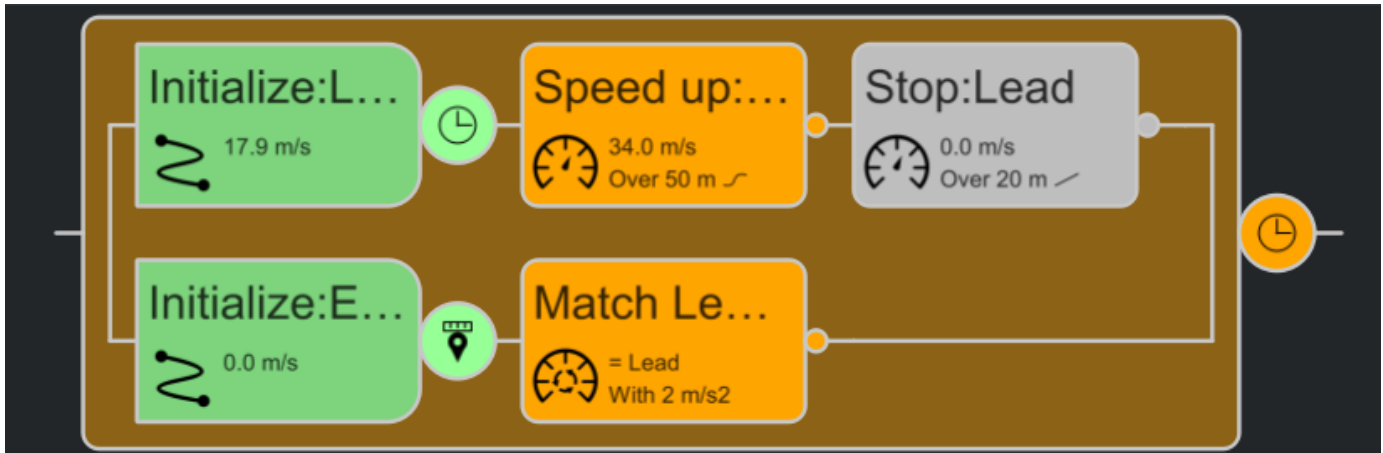
The scenario locks for editing and the simulation plays back in the scenario editing canvas.



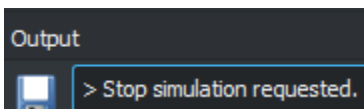
The **Logic** editor displays the status of the actions and conditions of the simulation. The colors indicate the status of actions and conditions:

- Green — Completed actions and conditions
- Orange — Active actions and conditions
- Gray — Actions and conditions not started or not reached during simulation
- Green and Red — Actions or conditions interrupted during simulation.

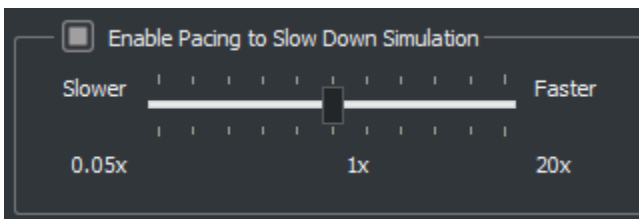
- Gray and Red — Actions or conditions skipped during simulation.



Let the scenario play until the simulation time ends, or click **Stop** to end the simulation early. The **Output** pane displays the reason that the simulation ends. For example, if you stop the simulation early, the **Output** pane displays a Stop simulation requested message.




Restart the scenario by pressing **Play** again, or by using the **Ctrl+R** keyboard shortcut. During this simulation run, slow down the pace of the scenario by selecting **Enable Pacing to Slow Down Simulation**. Use the slider to adjust the pace of the scenario.



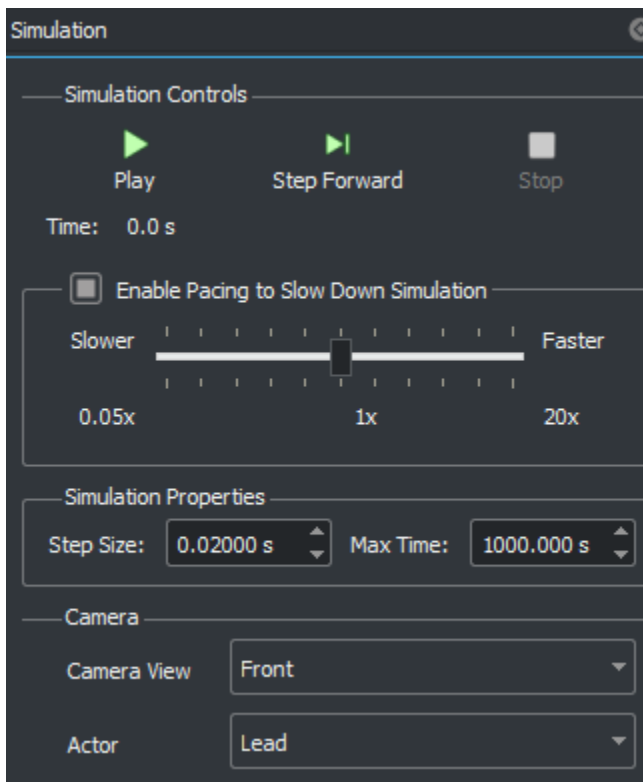
Alternatively, you can step through the simulation by selecting **Pause** and then clicking **Step Forward**.

Modifying the pacing does not change the results of the simulation. It changes only the rate at which the simulation is displayed on the screen.

Resume scenario editing by selecting the **Scenario Edit Tool**  from the toolbar. This enables scenario editing, and the actors revert back to their start positions.



You can visualize the scenario using different camera modes from the **Simulation** pane. Select a **Camera View** from the drop-down menu. Next, select the **Actor** for the camera view. In this example, the **Camera View** selected is **Front** and the **Actor** selected is **Lead**.



Click **Play** to start the simulation. In this example, the camera is placed at the front of the lead vehicle, the red sedan. This enables you to view the scenario from the front of the lead vehicle throughout the simulation. For more details about camera controls, see “Camera Control in RoadRunner Scenario”.



Limitations

- RoadRunner Scenario does not support simulation for very complex scenes. For example, a scene in a dense urban environment with numerous props and complex roads may be supported up to a size of 2000-by-2000 meters, whereas a scene in a more rural setting may support simulation up to a size of 5000-by-5000 meters.
- Sporadic failures to connect can occur on Linux® (Ubuntu® 16) operating systems. This issue can present itself as a failure to simulate, with the **Output** pane displaying **Failure to Connect** errors. As a workaround, in your Linux environment, set the `GRPC_DNS_RESOLVER` environment variable to `native`.

More About

Simulation End Conditions

When you create a new scenario, the **Logic** editor includes a default condition that causes the scenario to end.



By default, a simulation ends when either one of these conditions is met:

- Any actor collides with any other actor.
- The simulation reaches its defined end condition.

You can modify or delete either of these conditions. If you delete the collision condition, then collisions are not reported in the **Output** pane and actors appear to drive through each other during simulation. If you delete the end condition, the simulation continues indefinitely.

Tips

- To quickly restart a scenario simulation, press **Ctrl+R**.
- The **Output** pane displays information about previously run simulations. Use this pane to debug scenarios. For more details, see “Validate Scenarios”.

Version History

Introduced in R2022a

See Also

Scenario Edit Tool | **Route Timing Tool**

Topics

“Camera Control in RoadRunner Scenario”

Simulation Configuration

“Explore and Simulate a Simple Scenario”

“Validate Scenarios”

“Overview of Simulating RoadRunner Scenarios with MATLAB and Simulink” (Automated Driving Toolbox)

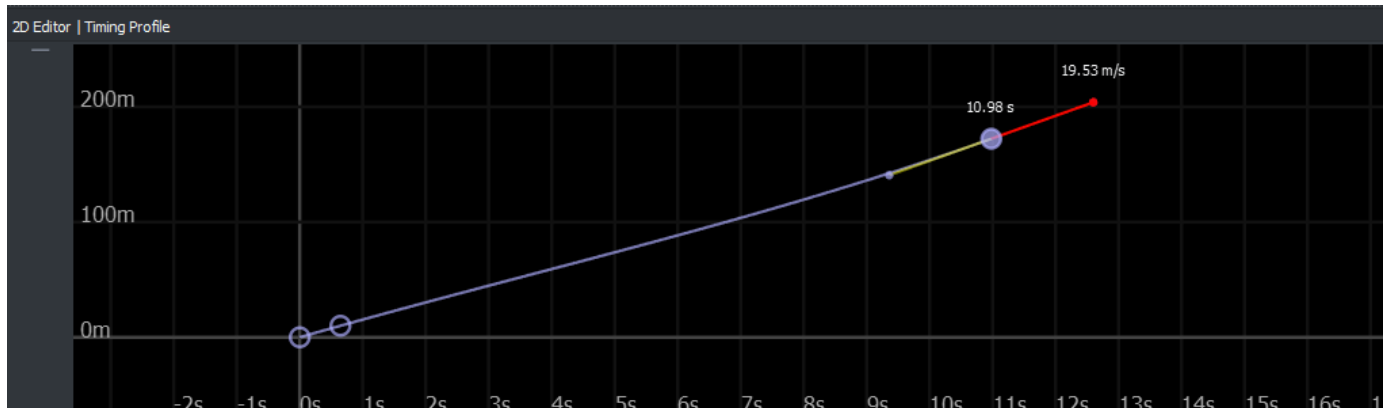
Route Timing Tool

Assign optional timing data to trajectories

Description

The **Route Timing Tool** enables you to assign or modify time and speed values of waypoints on a trajectory. Using the **Route Timing Tool**, you can graphically visualize a selected trajectory in the scenario.

When you click the **Route Timing Tool** button, the **2D Editor** switches to the **Timing Profile** view. You can edit the time and speed data of the trajectory by selecting points in the **Timing Profile**. Alternatively, you can edit the time and speed values through the **Attributes** pane. Assigning optional timing data to trajectories also enables exporting and importing timing data from ASAM OpenSCENARIO®.



Open the Route Timing Tool

On the RoadRunner Scenario toolbar, click the **Route Timing Tool** button:



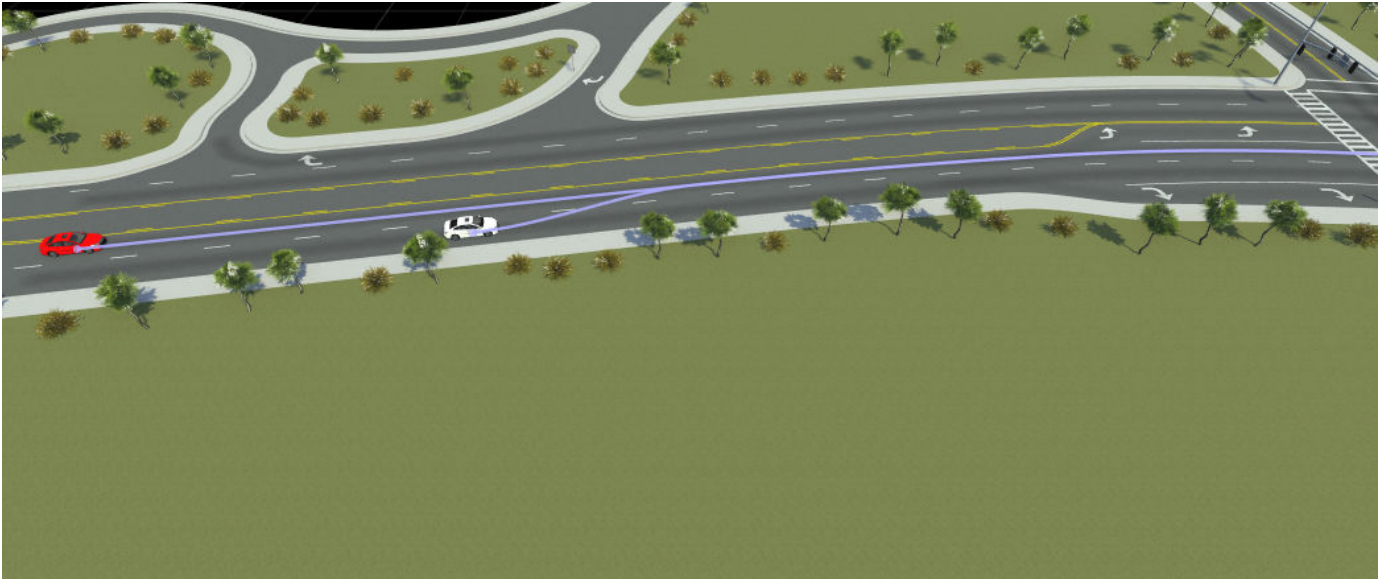
Examples

Modify Trajectory Timing Data Using Route Timing Tool

Open a prebuilt scenario, and then use the **Route Timing Tool** to assign or modify time and speed values of a trajectory.

From the **File** menu, select **Open Scenario**. Then, select the TrajectoryCutIn scenario, which is one of the prebuilt scenarios included by default in the Scenarios folder of RoadRunner Scenario.

In this scenario, one vehicle cuts in front of another vehicle. The vehicles follow predefined paths.

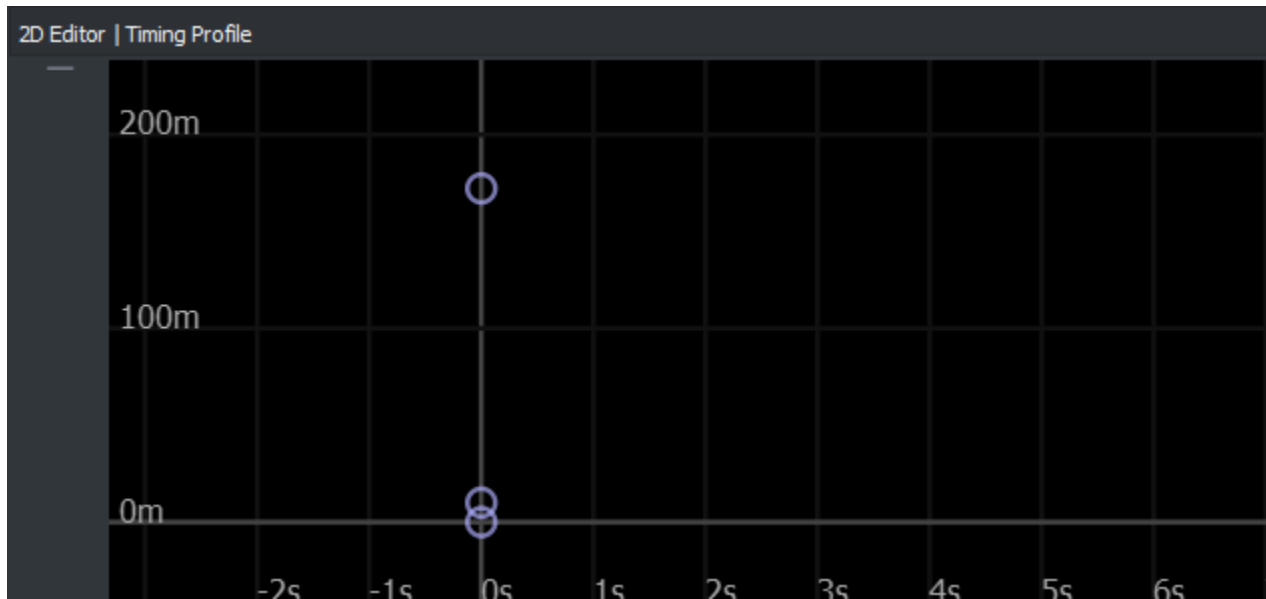


In the scenario editing canvas, click the route of red sedan to select it. The selected route turns red and two waypoints appear on the route.



On the RoadRunner Scenario toolbar, click the **Route Timing Tool**.

The **2D Editor** pane switches to the **Timing Profile** view. The **Timing Profile** displays a grid where the x - axis represents the time in seconds and the y - axis represents the distance of the waypoint in meters. The purple hollow points indicate the waypoints in the selected route. The hollow points indicate that the waypoints do not have time or speed data. You can edit time and speed data by selecting these points.

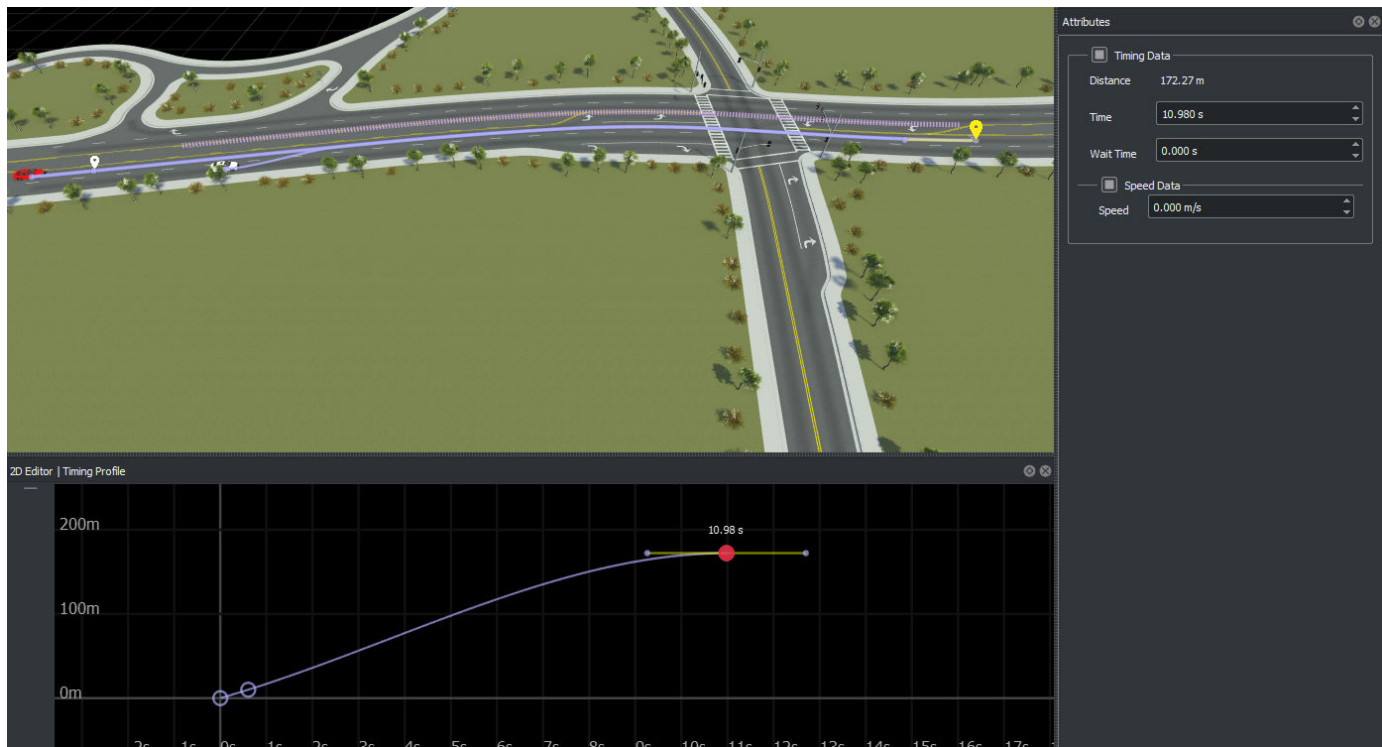


Select the yellow waypoint in the editing canvas by clicking it. This highlights its corresponding point in the **Timing Profile**. The green line on the purple point in the **Timing Profile** pane is the slope and indicates the speed at the particular waypoint. Alternatively, you can view the time and speed data in the **Attributes** pane by selecting **Timing Data** and **Speed Data**.

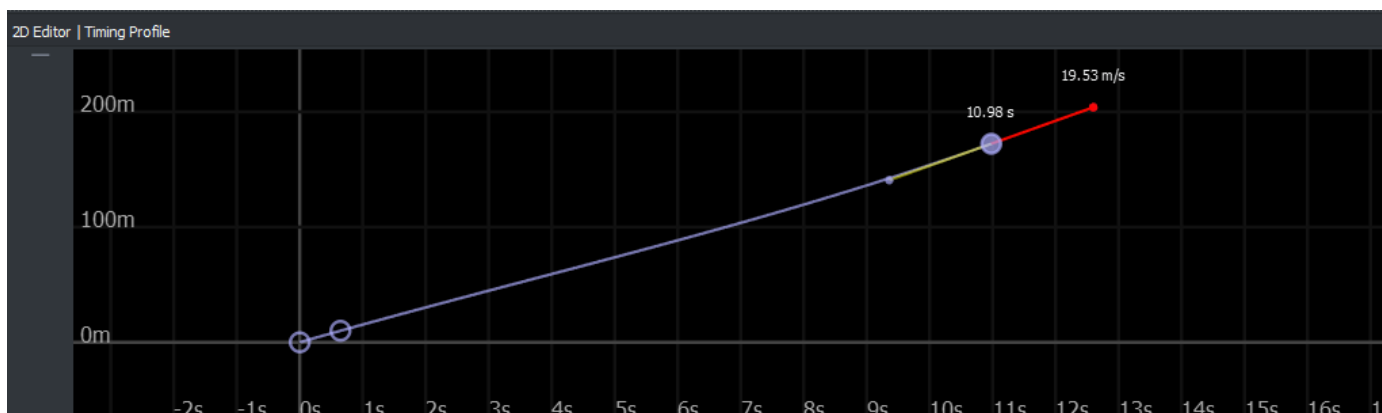
Customize the arrival time of the red sedan at the yellow waypoint by dragging the purple point forward from the y - axis. The label near the red highlighted point in the **Timing Profile** pane indicates that the modified arrival time is 10.98s. The same value appears in the **Time** field under

Timing Data in the **Attributes** pane. The slope at the red point is a straight line, indicating that the speed of the red sedan at the yellow waypoint will be 0m/s.

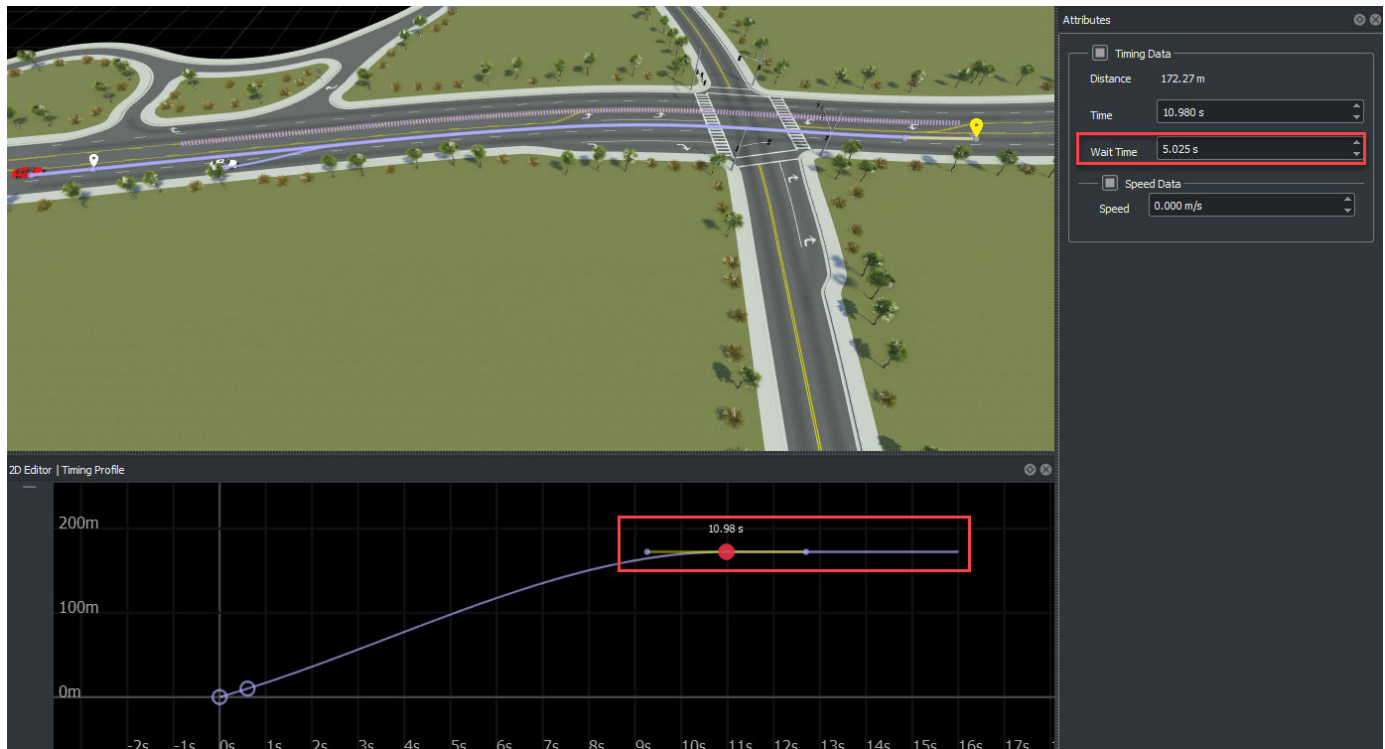
Modifying the arrival time at the yellow waypoint modifies the arrival time at other waypoints automatically by extrapolating those waypoints.



You can modify the speed of arrival of the red sedan at the yellow waypoint by selecting the slope in the **Timing Profile** and adjusting it. This changes the speed value and the label next to the slope indicates that the achieved slope is 19.53m/s. This modified value is also reflected in the **Speed Data** in the **Attributes** pane.



Optionally, you can set a wait time for the red sedan by entering a value in the **Wait Time** field under **Time Data** in the **Attributes** pane. As the wait time increase, the length of the flat section in the **Timing Profile** pane also increases. When the simulation plays, the red sedan waits for 5.025s before it drives to completion.



Version History

Introduced in R2022a

See Also

[Scenario Edit Tool](#) | [Simulation Tool](#)

Topics

- "Design Lane Change Scenario"
- "Explore and Simulate a Simple Scenario"
- "Validate Scenarios"

Assets

Vehicle Assets

Define vehicles to add to driving scenarios

Description



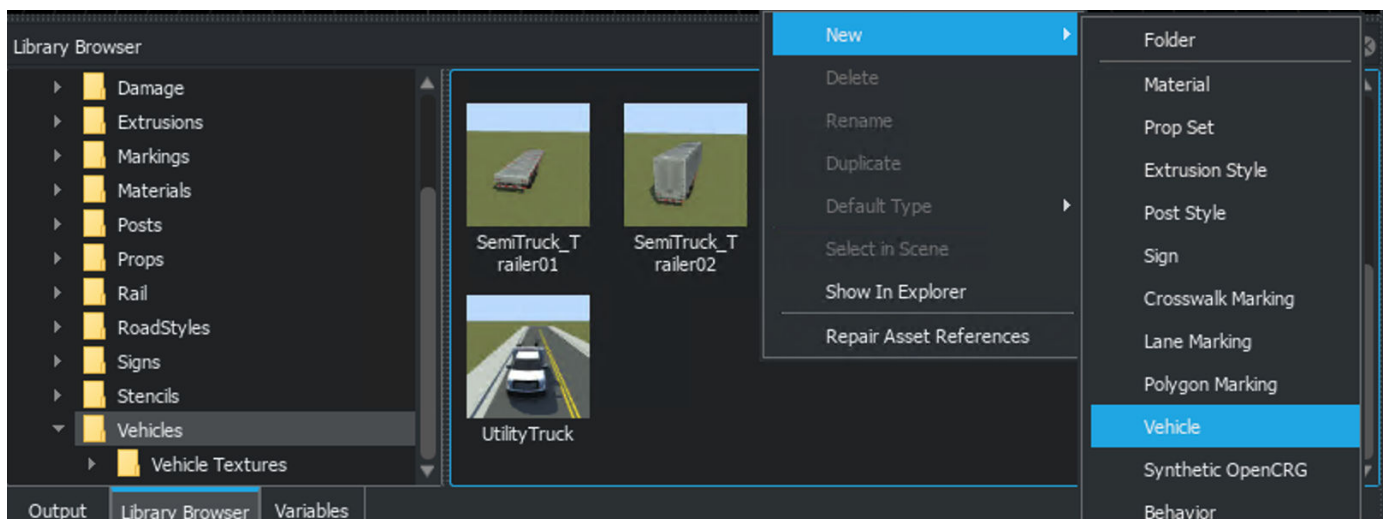
Vehicle assets are the primary actors used to populate driving scenarios. By dragging vehicle assets from the Vehicles folder of the **Library Browser** into the editing canvas, you can create dynamic driving scenarios.

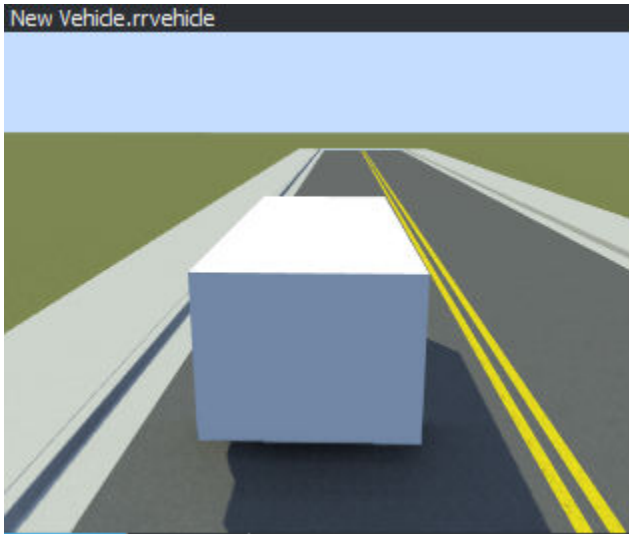
Creation

You can create new vehicle assets either directly within the **Library Browser** or by importing vehicles meshes created outside of RoadRunner into the **Library Browser**

Create Within Library Browser

From the **Library Browser**, select **Vehicle** folder. Right click in the **Library Browser**. Click **New**, then **Vehicle** to create a new vehicle asset with extension `.rrvehicle`. The asset browser displays this new vehicle as a cuboid shape.





You can then customize the attributes of the vehicle from the **Attributes** pane. For details on the attributes you can customize, see the “Placeholder Vehicle Dimensions” on page 2-4 section.

Alternatively, you can create a new asset by duplicating an existing asset. From the **Library Browser**, right click an existing vehicle asset and select **Duplicate**.

Note Asset duplication is not supported for assets created from external source files, such as FBX[®] or texture files.

Create from Imported Vehicle Meshes

If you have vehicle meshes created outside of RoadRunner that you want to import into the **Library Browser**, you can define them in such a way to make them compatible with scenarios. For more details, see “Import Custom Vehicle Meshes”.

Parameters

Attributes by Vehicle Type

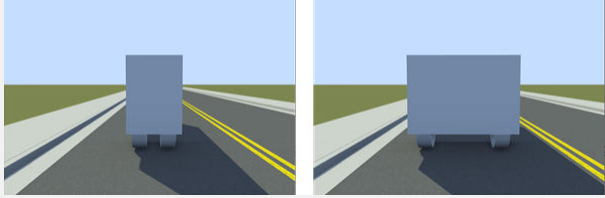
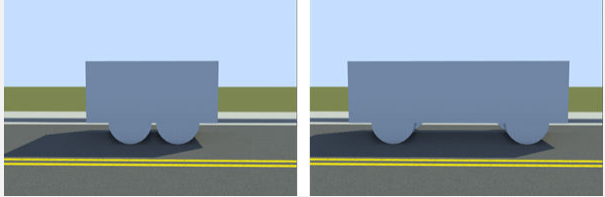
Vehicle assets in the **Library Browser** have attributes that are common across all assets of that type. When you select a vehicle from the **Library Browser**, you can view these attributes in the **Attributes** pane. These attributes are applied during scenario design and simulation but they are included in exported ASAM OpenSCENARIO files. This table describes the attributes. The default value of each attribute depends on the vehicle type.

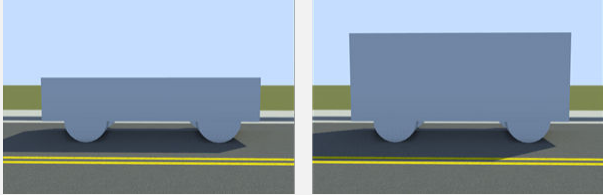
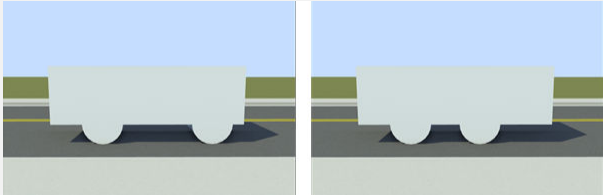
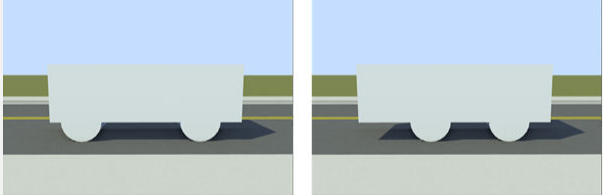
Attribute	Description
Category	Category of vehicle, such as Bus, Car, Van, or Truck.
Mass	Mass of vehicle, in kilograms, specified as a scalar in the range [0, 10,000).

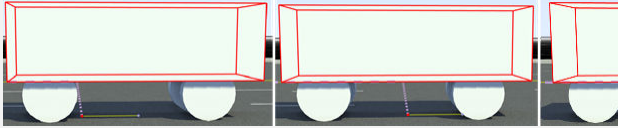
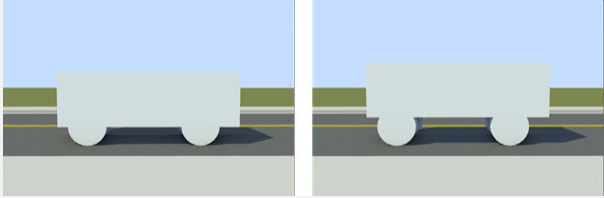
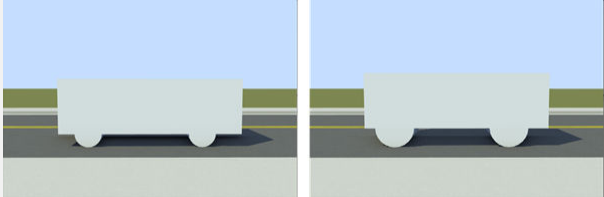
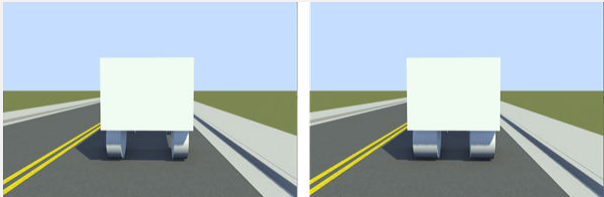
Attribute	Description
Max Speed	Maximum speed of vehicle, in meters per second, specified as a scalar in the range [0, 1,000).
Max Acceleration	Maximum acceleration of vehicle, in meters per second squared, specified as a scalar in the range [0, 1,000).
Max Deceleration	Maximum deceleration of vehicle, in meters per second squared, specified as a scalar in the range [0, 1,000).
Max Steering Angle	Maximum steering angle of vehicle, in degrees, specified as a scalar in the range [0, 100).

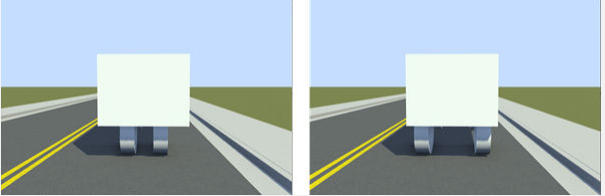
Placeholder Vehicle Dimensions

If you create new vehicles using the Create Within Library Browser procedure, then you can modify additional attributes for the vehicle dimensions. You can modify these dimensions, select the vehicle you created from the **Library Browser** and modify the attributes under **Placeholder Vehicle Dimensions**. RoadRunner Scenario displays the updated dimensions in the asset viewer and in any vehicles of this type added to the scenario. This table describes the placeholder vehicle dimensions.

Attribute	Description
Width	<p>Width of vehicle, in meters, specified as a scalar in the range [0.1, 20].</p>  <p>Default: 1.80 meters</p>
Length	<p>Length of vehicle, in meters, specified as a scalar in the range [0.1, 20].</p>  <p>Default: 4.70 meters</p>

Attribute	Description
Height	<p>Height of vehicle, in meters, specified as a scalar in the range [0.1, 20].</p>  <p>Default: 1.40 meters</p>
Front Overhang	<p>Front overhang of vehicle, in meters, specified as a scalar in the range [0, x], where x is half the Length of the vehicle. Front overhang is the distance from the front axle to the front bumper of the vehicle.</p>  <p>Default: 1.00 meters</p>
Rear Overhang	<p>Rear overhang of vehicle, in meters, specified as a scalar in the range [0, x], where x is half the Length of the vehicle. Rear overhang is the distance from the rear axle to the back bumper of the vehicle.</p>  <p>Default: 0.90 meters</p>

Attribute	Description
<p>Forward Offset</p>	<p>Forward offset, in meters, of the vehicle from its origin, specified as a scalar in the range $[-x, x]$, where x is half the Width of the vehicle. The vehicle origin is on the ground, at the geometric center of the vehicle.</p>  <p>Default: 0.00 meters</p>
<p>Body Height Gap</p>	<p>Gap, in meters, between the vehicle body and the center of its wheel axles, specified as a scalar in the range $[0, 10]$.</p>  <p>Default: 0.00 meters</p>
<p>Wheel Radius</p>	<p>Radius of vehicle wheels, in meters, specified as a scalar in the range $[\]()$.</p>  <p>Default: 0.50 meters</p>
<p>Wheel Width</p>	<p>Width of vehicle wheels, in meters, specified as a scalar in the range $[0.05, 0.9]$.</p>  <p>Default: 0.30 meters</p>

Attribute	Description
Wheel Inset	<p data-bbox="865 300 1466 426">Inset of wheels on vehicle, in meters, specified as a scalar in the range [0, 5]. Wheel inset is the distance between the outer edge of the wheel and the outer edge of the vehicle.</p>  <p data-bbox="865 678 1133 709">Default: 0.05 meters</p>

Vehicle-Specific Attributes

When you add a vehicle to a scenario, you can set attributes that are specific to that vehicle from the **Attributes** pane. These sections describe the attributes you can set.

Vehicle

Attribute	Description
Name	Name of vehicle. In the Logic editor, this name displays in the action phases that apply to that vehicle.
Actor Id	Unique id for the actor. You can modify the actor id by incrementing or decrementing its value in the Attributes pane.
Color	Color of vehicle. To change a vehicle color, select the current color from the Color attribute box and set the color in the Set Color dialog box.
Vehicle Type	Asset type used to display the vehicle in the scenario. To select a new asset type for the selected vehicle, drag a vehicle asset from the Attributes pane onto the Vehicle Type attribute box.
Behavior	<p data-bbox="865 1482 1466 1671">Driving behavior of vehicle. If you do not specify a behavior, then the vehicle either follows its lane or drives along its specified driving path during simulation. To specify a vehicle behavior, drag a behavior asset from the Library Browser onto the Behavior attribute box.</p> <p data-bbox="865 1696 1466 1820">You can specify behaviors defined in RoadRunner, in MATLAB® or Simulink®, or in external simulators such as CARLA. For more details, see “Specify and Assign Actor Behaviors”.</p>


Connection Attributes

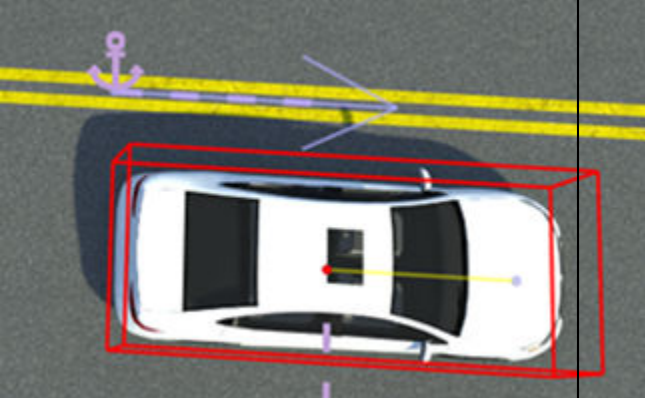
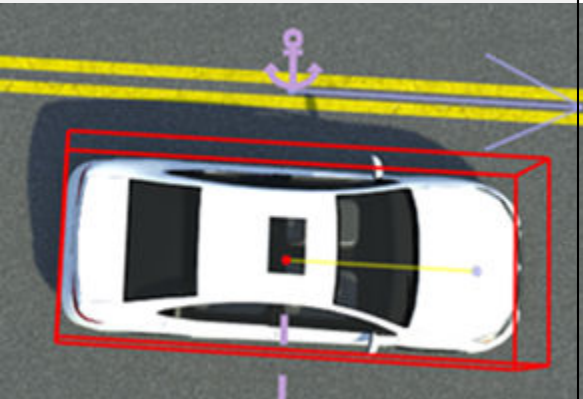
Use these attributes to connect vehicles together, such as a trailer connected to a truck.

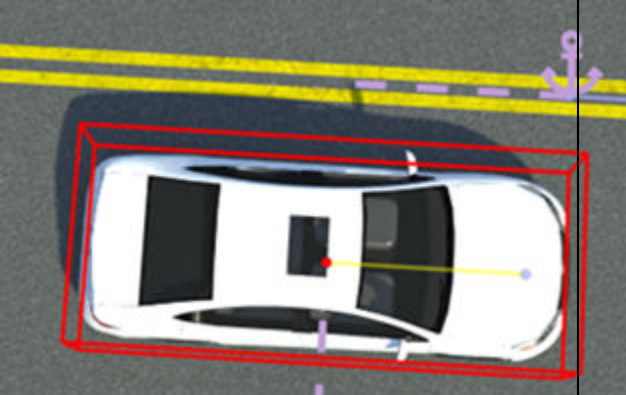
Attribute	Description
Parent Attachment	Parent vehicle for the current vehicle. When you assign a parent vehicle, the Attributes pane changes to show the “Connection Attributes” on page 2-12.

Point Offsets

Using these attributes, you can specify the vehicle relative to a specific point in the scenario, such as a road anchor, path waypoint, or to another vehicle.

Attribute	Description
Enable Anchoring	Enable vehicle to anchor to another point. By default, this attribute is enabled.
Position	Position of vehicle within the scene. Specify the (x,y,z) position of the vehicle by using the enabled X , Y , and Z parameters. Units are in meters. The position is relative to the center of the scene. This parameter is only available if Enable Anchoring attribute is disabled.
Anchor	Anchor point that the vehicle is offset from. An anchor point can be a road anchor, path waypoint, or another vehicle. To select an anchor point, first click the attribute box. Then, select an anchor point from the scenario editing canvas or the Logic editor. RoadRunner outlines these selection areas with blue lines. To frame the camera around the current anchor point in the scenario, click the Frame object in the scene button  .
Lock To Anchor	Lock the vehicle to its current anchor no matter where you drag it within the scenario. If you do not select this attribute, then the vehicle locks to new road anchors as you drag it onto different roads. By default, this attribute is not selected.
Forward Offset	Distance, in meters, that the vehicle is in front of its anchor point.

Attribute	Description
<p>Manual Reference Line</p>	<p>Reference line that the forward offset of the vehicle from its anchor point is measured from, specified as Front (measure from front bumper), Middle (measure from vehicle center) or Back (measure from back bumper). These images show the different reference lines, where the vehicle has a Forward Offset of 0 meters from the road anchor.</p> <ul style="list-style-type: none"> • Front  <ul style="list-style-type: none"> • Middle  <ul style="list-style-type: none"> • Back

Attribute	Description
	

Lane Offset

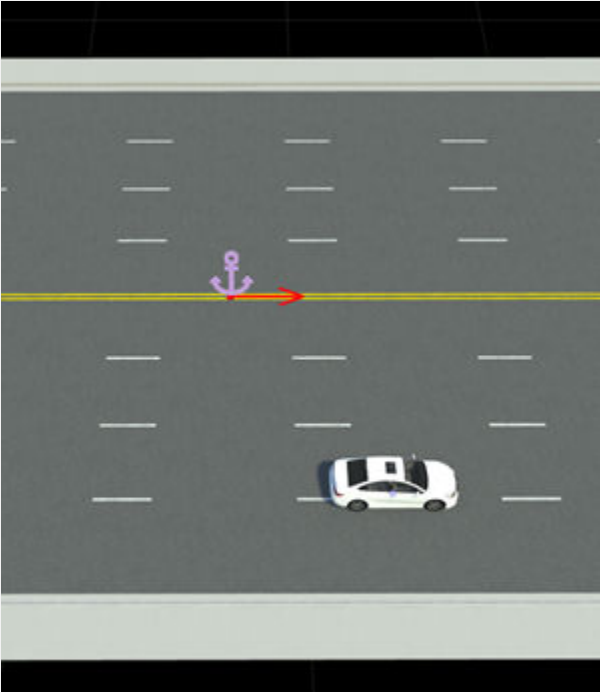
Using these attributes, you can place vehicles relative to the road edge and relative to the lane they are in.

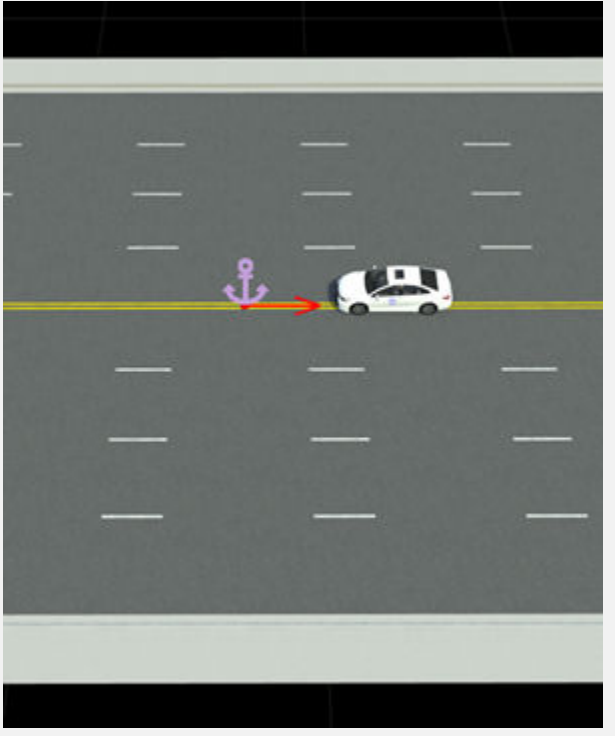
When specifying lane offsets, keep these points in mind:

- Lane offsets can be relative to the road edge only (**Relative To** = Road Edge) attribute. You cannot offset vehicles from the lanes that other vehicles are in.
- Lateral offsets within a lane (**Lateral Offset** attribute) are positive to the right of the vehicle, as shown by this figure.



These sample attributes show the key values that you can set.

Attributes	Description
<p>Relative To – Road Edge</p> <p>Offset From – Left Lane</p> <p>Direction – 2 lane(s)</p> <p>Travel Direction – With Road Anchor</p> <p>Lateral Offset – 1.5 meters</p>	<p>Set vehicle two lanes from the left road edge, traveling in the same direction as the road anchor. Offset the vehicle 1.5 meters to the right within its lane.</p>  <p>The diagram shows a top-down view of a road with two lanes. A yellow double line runs horizontally across the middle of the road, with a purple anchor icon and a red arrow pointing to the right on it. A white car is positioned in the right lane, below the yellow line. The road is flanked by grey shoulders and black areas representing the background.</p>

Attributes	Description
<p>Relative To — Road Edge</p> <p>Offset From — Right Lane</p> <p>Direction — 0 lane(s)</p> <p>Travel Direction — Against Road Anchor</p> <p>Lateral Offset — -1.5 meters</p>	<p>Set vehicle in the lane along the left road edge, traveling in the opposite direction of the road anchor. Offset the vehicle 1.5 meters to the left within its lane.</p> 

Connection Attributes

When you connect a vehicle to another vehicle as a trailer, you can set attributes that are specific to that trailer from the **Attributes** pane. These sections describe the attributes you can set.

Note The connection attributes only become available when the vehicle gets attached to a parent vehicle and becomes a trailer.

Connection Attributes

Use these attributes to connect vehicles together, such as trailer connected to a truck.

Attribute	Description
Parent Attachment	Name of the parent vehicle that the current vehicle attaches to.

Attribute	Description
Type	Specify the behavior of the trailer relative to the parent. Select trailer to allow the trailer to move and use independent dynamics relative to the parent vehicle. Select fixed to lock the trailer rotation to the dynamics of the parent vehicle.

Attachment Point

Use these attributes to edit the attachment point with respect to the selected vehicle, and the Parent Attachment point.


Attribute	Description
Source	Generate an attachment point from the bounding box geometry of the vehicle, select Computed from Geometry . To use an attachment point defined in the FBX file, select Imported from File .
Point Name	<p>When the Source attribute value is Imported from File, this attribute specifies the label of the point in the FBX file. The name of the node in the FBX file must either be hitch or start with attach to appear as an option for the Point Name attribute.</p> <p>When the Source attribute value is Computed from Geometry, this attribute specifies the point in bounding box geometry.</p>

Parent Attachment Point

Attribute	Description
Source	Specify the attachment point on the parent vehicle. To generate an attachment point from the vehicle's bounding box geometry, select Computed from Geometry . To use an attachment point defined in the FBX file, select Imported from File .
Point Name	<p>When the Source attribute value is Imported from File, this attribute specifies the label of the point in the FBX file. The name of the node in the FBX file must either be hitch or start with attach to appear as an option for the Point Name attribute.</p> <p>When the Source attribute value is Computed from Geometry, this attribute specifies the point in bounding box geometry.</p>

Attribute	Description
Offset Right	Horizontal offset, in meters, of the vehicle relative to the parent attachment point in the direction of travel. Default offset right is 0.0 meters.
Offset Forward	Horizontal offset, in meters, of the vehicle relative to the parent attachment point in the direction of travel. Default offset height is 0.0 meters.
Offset Up	Vertical offset, in meters, of the vehicle relative to the parent attachment point in the direction of travel. Default offset height is 0.0 meters.

Use these attributes to specify the trailer direction to the parent attachment point.

Attribute	Description
Heading	<p>Horizontal direction, in degrees, of the trailer relative to the parent attachment point. Default offset heading is 0.0 degrees.</p> <p>When you select a trailer, you can also modify the Heading value the editor using the direction widget, shown in this image, when the trailer is selected.</p> 
Slope	Slope or pitch, in degrees, of the trailer relative to the parent attachment point.

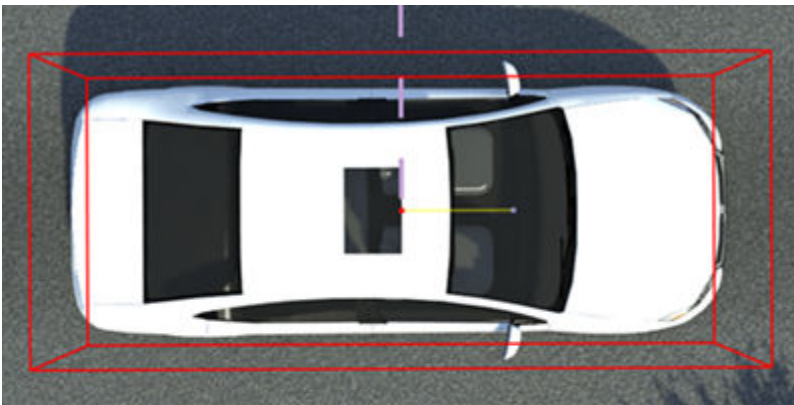
Examples

Add Vehicles to Scenario

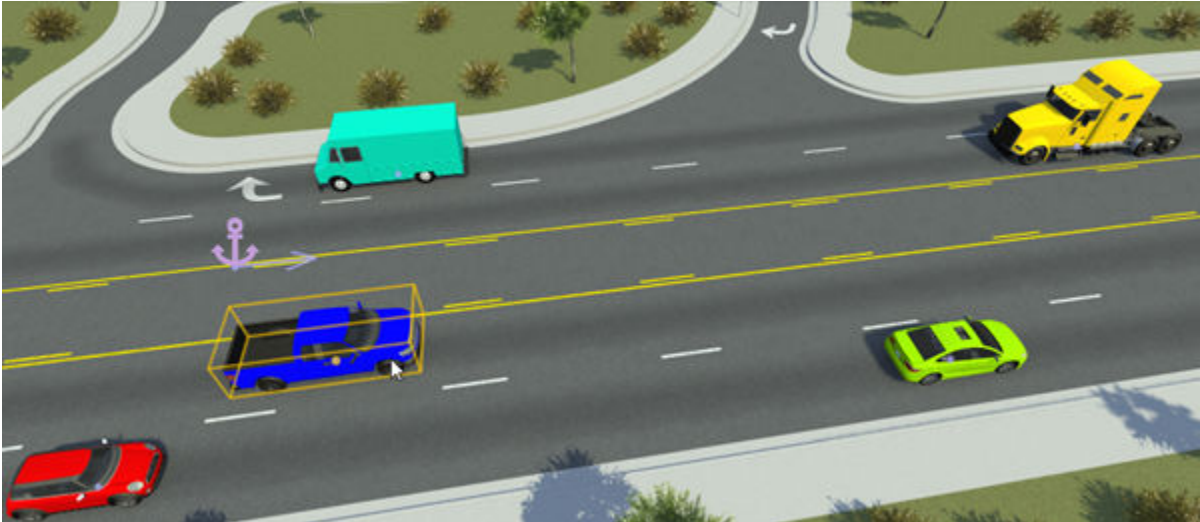
When you switch the scenario editing mode, RoadRunner Scenario opens the **Library Browser** to the **Vehicles** folder. This folder includes a **Sedan** asset. Drag this asset into the scenario.



By default, assets snap to the center of lanes along road networks. The origin of vehicle is located on the ground below its geometric center.



If you have the RoadRunner Asset Library, then the **Vehicles** folder includes a variety of additional vehicle assets that you can add to the scenario.



You can then modify the appearance or behavior of vehicles from the **Attributes** pane. For more details, see the **Parameters** on page 2-3 section.

To control how the vehicle moves in the scenario, you can choose one of the following actions:

- Set a driving path for a vehicle. For more details, see “**Path Editing**”. For an example, see “**Design Path Following Scenario**”.
- Use the built-in behavior of the vehicle. By default, vehicles follow their current lane. For more details, see “**Built-In Behavior for Vehicles**”. For an example, see “**Design Lane Following Scenario**”.
- Specify a custom behavior for the vehicle and add this behavior asset to the **Behavior** attribute of the vehicle. For more details, see “**Specify and Assign Actor Behaviors**”.

Limitations

- In scene editing mode, if you add vehicle assets in the **Vehicles** folder into a scene, the vehicle is treated as part of the scene. In scenario editing mode, you cannot edit or simulate with this vehicle.
- Vehicle assets in the **Vehicles** folder of projects created in R2021b or earlier are treated only as prop assets and you cannot simulate scenarios with them. As a workaround, create a new project and in your scenario, use the vehicle assets in the **Vehicles** folder of the **Library Browser**. Alternatively, copy the **Vehicles** folder from a new project into the **Library Browser** of your existing project.

Tips

You can set vehicle colors, asset types, and other attributes as variables, which you can then modify programmatically. Using this technique, you can generation variations of scenarios with different vehicles. For more details, see “**Generate Scenario Variations Using gRPC API**”.

See Also

Topics

- “Explore and Simulate a Simple Scenario”
- “Design Lane Following Scenario”
- “Design Path Following Scenario”
- “Built-In Behavior for Vehicles”
- “Specify and Assign Actor Behaviors”
- “Define Scenario Logic”
- “Scenario Anchoring System”
- “Path Editing”
- “Design Vehicle with Trailer Scenario”

Character Assets

Define characters or pedestrians to add to driving scenarios

Description

Character assets are secondary actors used to populate driving scenarios. By dragging pedestrian assets from the Characters folder of the **Library Browser** into the editing canvas, you can create dynamic driving scenarios with pedestrians.

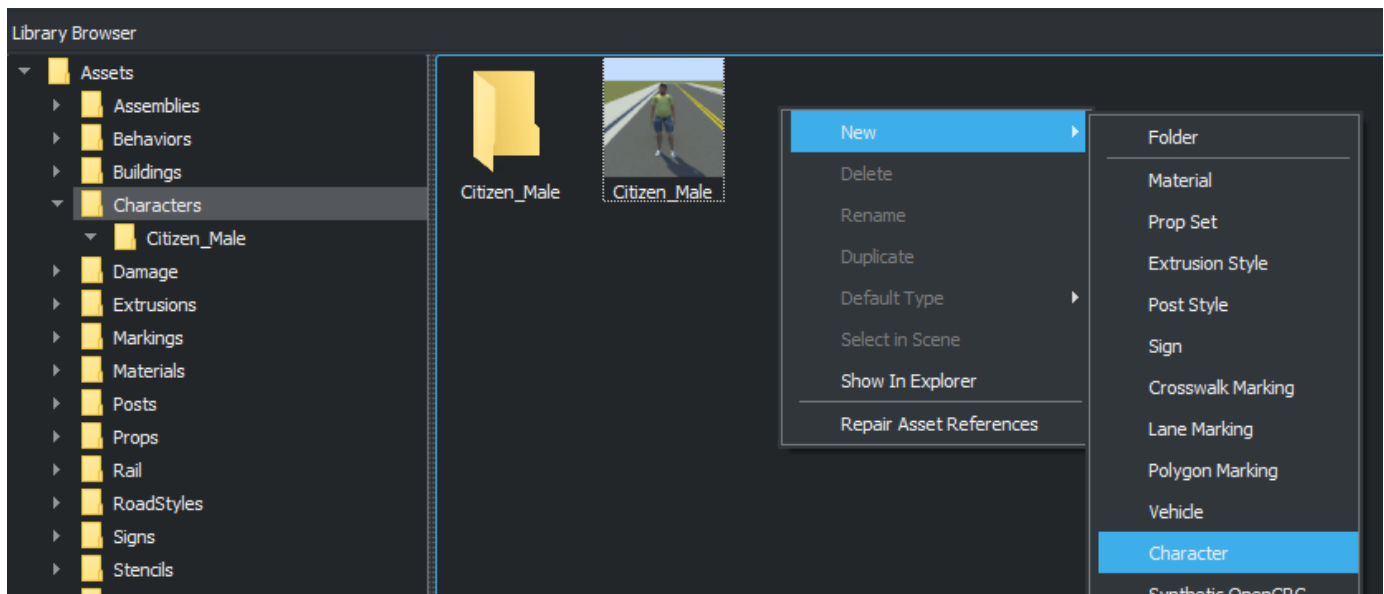
Characters use the same tools to specify their path behavior within a scene as vehicles. For more information on specifying path behaviors, see “Path Editing”.

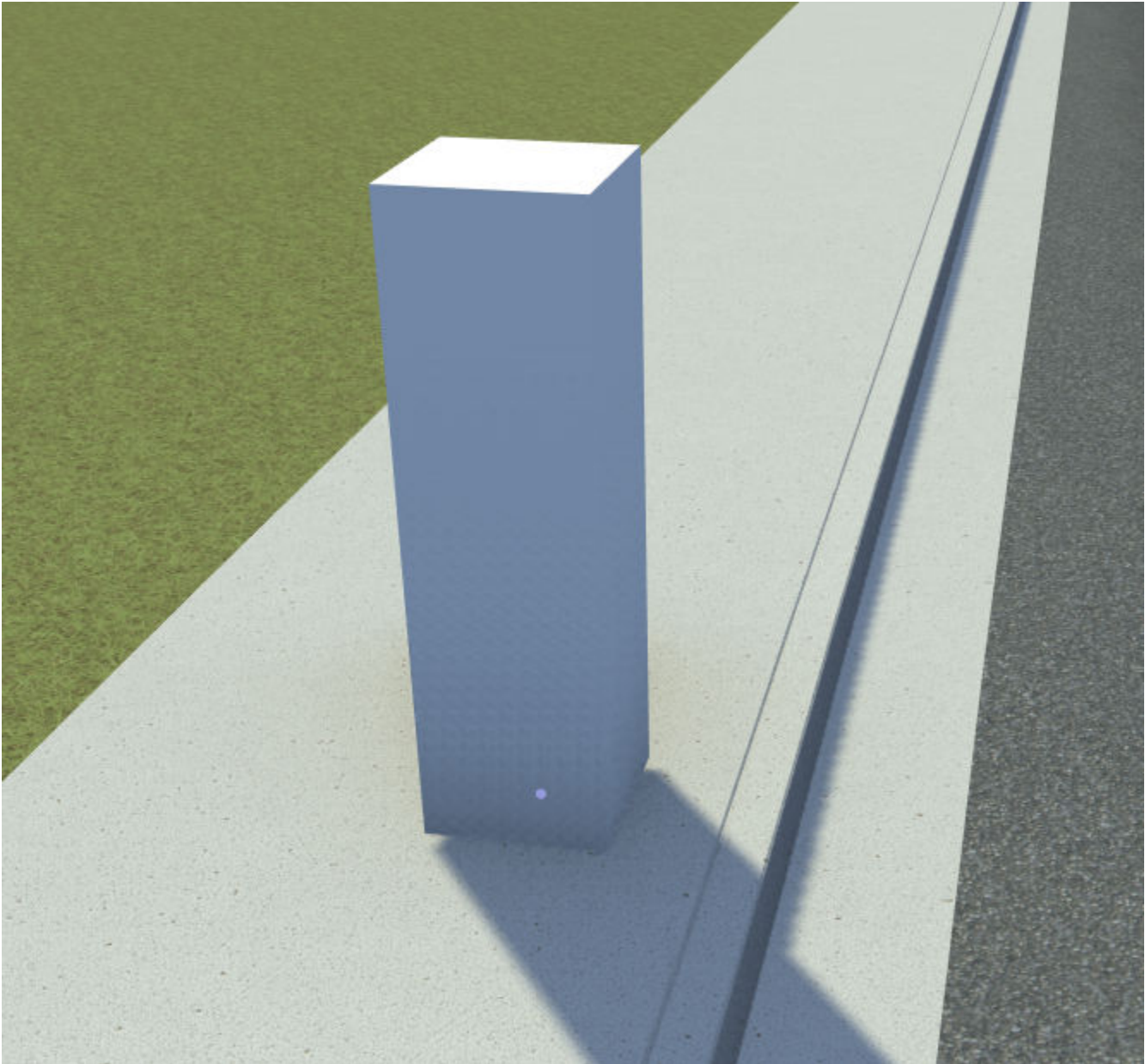
Creation

You can create new character assets either from directly within the **Library Browser** or by importing character meshes created outside of RoadRunner into the **Library Browser**

Create Within Library Browser

From the **Library Browser**, select the Characters folder. Right-click in the **Library Browser** and, from the context menu, select **New**, then **Character** to create a new character asset with the extension `.rrchar.rrmeta`. The asset browser displays this new character as a cuboid shape.





You can then customize the attributes of the character from the **Attributes** pane. For details on the attributes you can customize, see “Placeholder Character Dimensions” on page 2-20.

Create from Imported Character Meshes

To import character meshes created outside of RoadRunner into the **Library Browser**, you must define them in such a way as to make them compatible with scenarios. For more details, see “Import Custom Character Meshes”.

Parameters


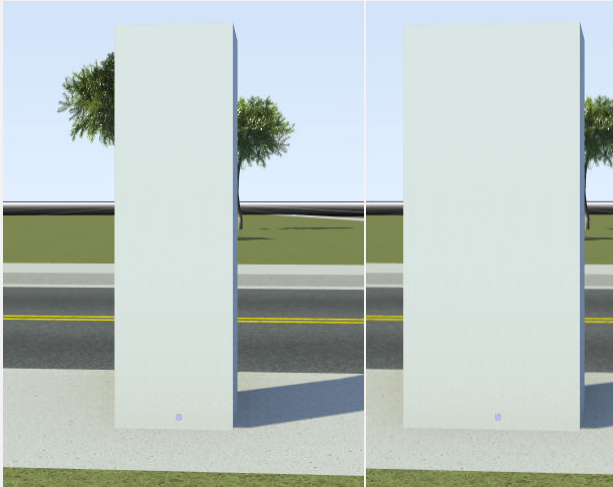
Attributes by Character Type

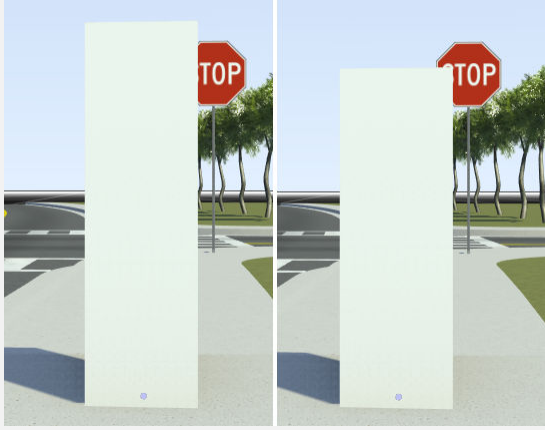
Character assets in the **Library Browser** have attributes that are common across all assets of that type. When you select a character from the **Library Browser**, you can view these attributes in the **Attributes** pane. These attributes are applied during scenario design and simulation, and they are included in exported ASAM OpenSCENARIO files. This table describes the attributes. The default value of each attribute depends on the character type.

Attribute	Description
Category	Category of character, such as Pedestrian, Animal, or Wheelchair.
Skin	FBX file containing the mesh and skeletal rig.
Skeleton	FBX file containing the mesh and skeletal rig.
Idle Animation	FBX file containing an idle animation generated from the skeletal rig in the Skeleton attribute.
Walk Animation	FBX file containing a walking animation generated from the skeletal rig in the Skeleton attribute.
Run Animation	FBX file containing a running animation generated from the skeletal rig in the Skeleton attribute.

Placeholder Character Dimensions

If you create new pedestrians within the **Library Browser**, then you can modify additional attributes for the pedestrian dimensions. To modify these dimensions, select the character you created in the **Library Browser** and modify the attributes under **Placeholder Character Dimensions**. RoadRunner Scenario displays the updated dimensions in the asset viewer and in any vehicles of this type added to the scenario. This table describes the placeholder vehicle dimensions.

Attribute	Description
<p>Width</p>	<p>Width of the character, in meters, specified as a scalar in the range [0.1, 20].</p>  <p>Default: 0.50 meters</p>
<p>Length</p>	<p>Length of the character, in meters, specified as a scalar in the range [0.1, 20].</p>  <p>Default: 0.50 meters</p>

Attribute	Description
<p>Height</p>	<p>Height of the character, in meters, specified as a scalar in the range [0.1, 20].</p>  <p>Default: 1.70 meters</p>

Character-Specific Attributes

When you add a character to a scenario, you can set attributes that are specific to that character in the **Attributes** pane. These tables describe the attributes you can set.


Character

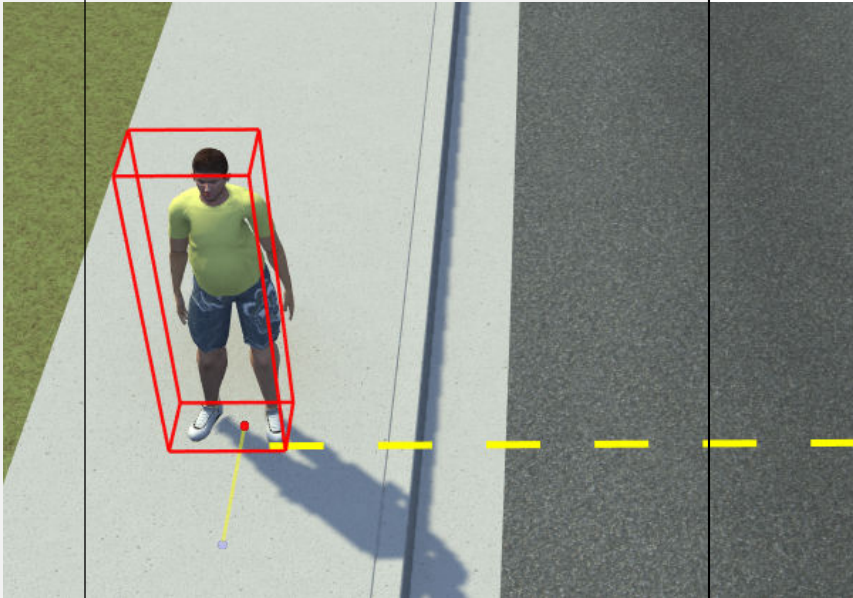
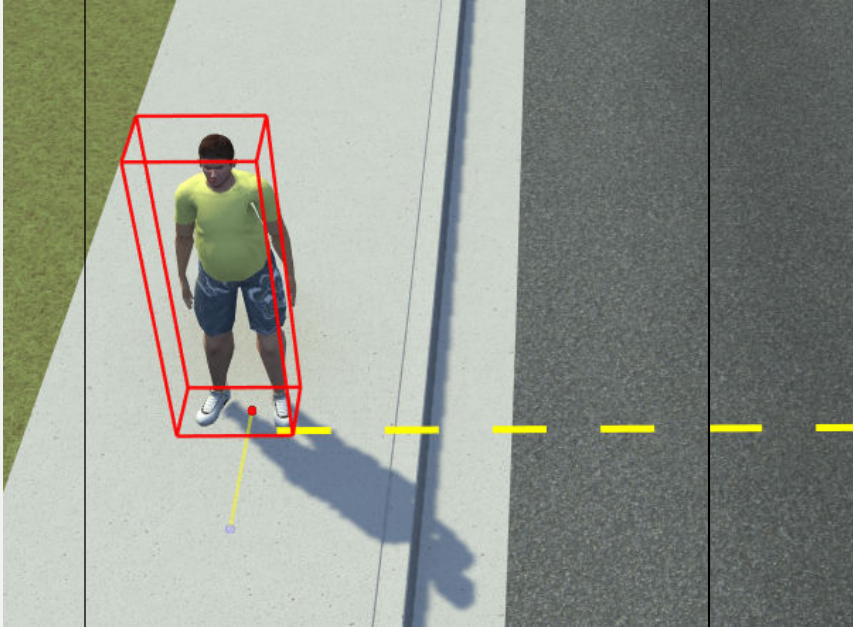
Attribute	Description
<p>Name</p>	<p>Name of the character. In the Logic editor, this name displays in the action phases that apply to that character.</p>
<p>Actor Id</p>	<p>Unique ID for the actor. You can modify the actor ID by changing its value in the Attributes pane.</p>
<p>Color</p>	<p>Color of character. To change a character color, select the current color from the Color attribute box and set the color in the Set Color dialog box. This applies to only the Placeholder Pedestrian character.</p>
<p>Character Type</p>	<p>Asset type used to display the character in the scenario. To select a new asset type for the selected character, drag a character asset from the Attributes pane onto the Character Type attribute box.</p>

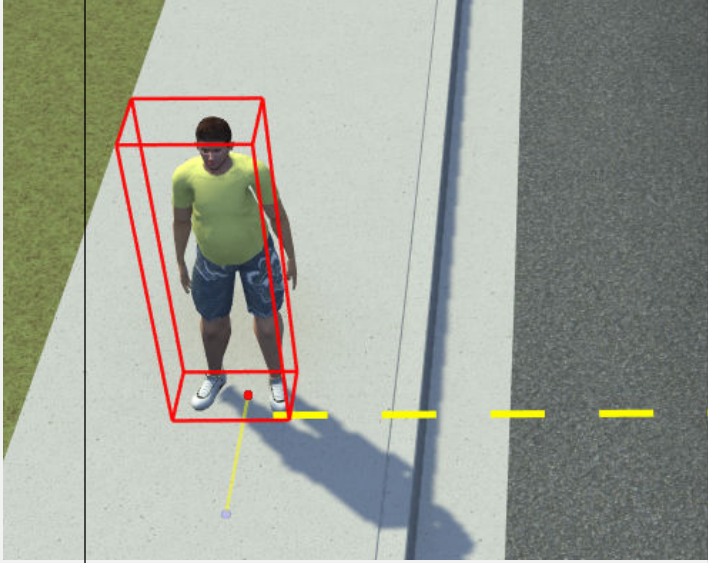
Attribute	Description
Behavior	<p>Behavior of the character. If you do not specify a behavior, then the character either follows its lane or moves along its specified path during simulation. To specify a character behavior, drag a behavior asset from the Library Browser onto the Behavior attribute box.</p> <p>You can specify behaviors defined in RoadRunner, in MATLAB or Simulink, or in external simulators such as CARLA. For more details, see “Specify and Assign Actor Behaviors”.</p>

Point Offsets

Using these attributes, you can specify the character relative to a specific point in the scenario, such as a road anchor, path waypoint, vehicle, or another character.

Attribute	Description
Enable Anchoring	Enable character to anchor to another point. By default, this attribute is enabled.
Position	<p>Position of character within the scene. Specify the xyz-position of the character by using the X, Y, and Z attributes. Units are in meters. The position is relative to the center of the scene.</p> <p>To enable this attribute, you must clear the Enable Anchoring attribute.</p>
Anchor	<p>Anchor point that the character is offset from. An anchor point can be a road anchor, path waypoint, vehicle, or another character.</p> <p>To select an anchor point, first click the attribute box. Then select an anchor point from the scenario editing canvas or the Logic editor. RoadRunner outlines these selection areas with blue lines. To frame the camera around the current anchor point in the scenario, click the</p> <p>Frame object in the scene button .</p>
Lock To Anchor	Lock the character to its current anchor no matter where you drag it within the scenario. If you do not select this attribute, then the character locks to new road anchors as you drag it onto different roads. By default, this attribute is not selected.
Forward Offset	Distance, in meters, that the character is in front of its anchor point.

Attribute	Description
Reference Line	<p>Reference line from which to measure the forward offset of the character from its anchor point, specified as Front, Middle, or Back. These images show the different reference lines, where the vehicle has a Forward Offset of 0 meters from the road anchor.</p> <ul style="list-style-type: none"><li data-bbox="863 510 984 541">• Front  <ul style="list-style-type: none"><li data-bbox="863 1178 1000 1209">• Middle 

Attribute	Description
	<ul style="list-style-type: none"> • Back 

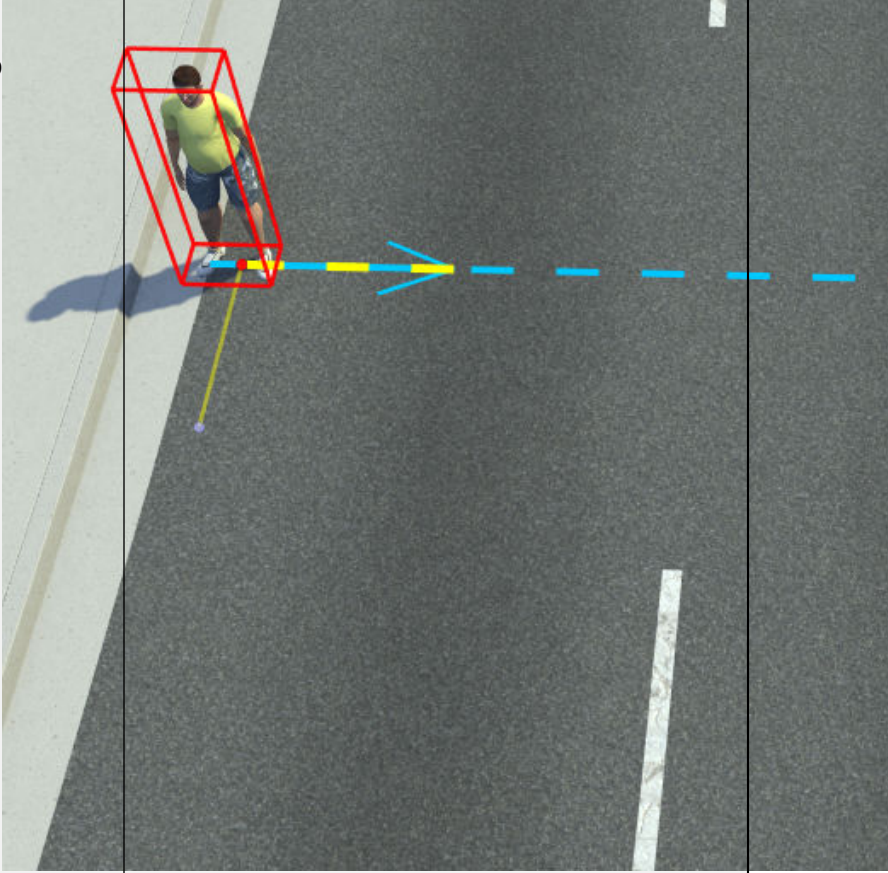
Lane Offset

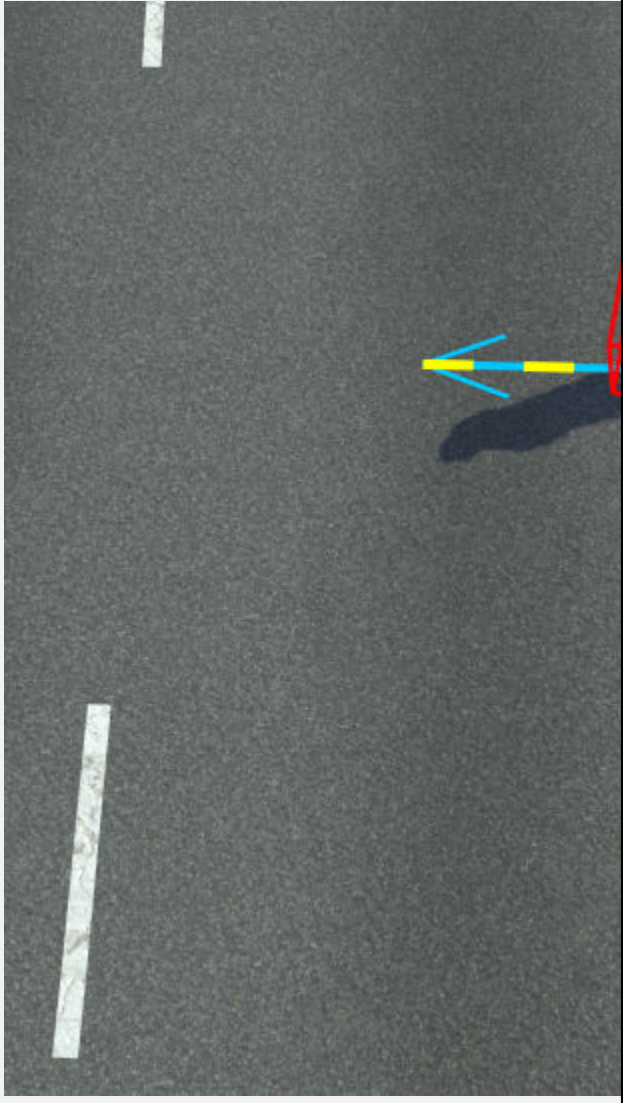
Using these attributes, you can place a character relative to the road edge and relative to the lane they are in.

When specifying lane offsets, keep these points in mind:

- Lane offsets can be relative to the road edge only (**Relative To** attribute value must be Road Edge) attribute. You cannot offset characters from the lanes that other characters or vehicles are in.
- Lateral offsets within a lane (**Lateral Offset** attribute) are positive to the right of the character.

These sample attributes show the key values that you can set.

Attributes	Description
<p>Relative To — Road Edge</p> <p>Offset From — Left Lane</p> <p>Lane Offset — 0 lane(s)</p> <p>Travel Direction — With Road Anchor</p> <p>Lateral Offset — 1.5 meters</p> <p>Direction — To the right</p>	<p>Set character traveling in the same direction as the road anchor. Offset the character 1.5 meters to the right from the lane.</p> 

Attributes	Description
<p>Relative To — Road Edge</p> <p>Offset From — Right Lane</p> <p>Lane Offset — 0 lane(s)</p> <p>Travel Direction — Against Road Anchor</p> <p>Lateral Offset — -1.5 meters</p> <p>Direction — To the right</p>	<p>Set character in the lane along the left road edge, traveling in the opposite direction of the road anchor. Offset the character 1.5 meters to the left within the lane.</p> 

Version History

Introduced in R2022b

See Also

Vehicle Assets

Topics

“Import Custom Character Meshes”

“Path Editing”

Functions

NewScenario

Create new RoadRunner scenario using gRPC

Description

The `NewScenario` method creates a new scenario in the current RoadRunner scene. If RoadRunner has no current scene loaded, then the `NewScenario` method call fails.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC® compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

NewScenarioRequest — New scenario request

empty message

New scenario request, specified as an empty message.

Response

NewScenarioResponse — New scenario response

empty message

New scenario response, returned as an empty message.

Sample Calls

Command Line

Create a new scenario in the project located at `C:\RR\MyProject`.

```
cd "C:\Program Files\RoadRunner R2022b\bin\win64"  
AppRoadRunner --projectPath C:\RR\MyProject  
CmdRoadRunnerApi "NewScenario()"
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner API service. For examples that use this command, see:

- “Generate Scenario Variations Using gRPC API”
- “Reuse Scenarios in Multiple Scenes Using gRPC API”
- “Export Multiple Scenarios Using gRPC API”

Python

Create a new scenario in the current scene.


```
newScenarioRequest = roadrunner_service_messages_pb2.NewScenarioRequest()
api.NewScenario(newScenarioRequest)
```

This sample call is a snippet of a Python® client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Create a new scenario in the current scene.

```
NewScenarioRequest request;
ClientContext context;
NewScenarioResponse reply;
Status status = api->NewScenario(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2022a

See Also

NewProject | LoadProject | SaveProject | NewScene | LoadScene | SaveScene | LoadScenario | SaveScenario | SetScenarioVariable | PrepareSimulation | SimulateScenario | Export | Import | Exit | roadrunner_service.proto | roadrunner_service_messages.proto

Topics

“Control RoadRunner Programmatically Using gRPC API”

LoadScenario

Load RoadRunner scenario using gRPC

Description

The `LoadScenario` method loads a specified scenario from the current RoadRunner project. If the scenario that you specify does not belong to the current project, then RoadRunner determines what project the scenario belongs to and loads it from that project instead.

If the scenario was previously saved with the current scene, or if you enable the `keep_current_scene` option, then the scenario loads into the current scene. Otherwise, RoadRunner loads the scene that the scenario was previously saved with and loads the scenario into that scene.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

LoadScenarioRequest — Load scenario request

message

Load scenario request, specified as a message with these fields

Name	Type	Description
<code>file_path</code> (required)	string	<p>Absolute or relative path to the scenario file to load. If you specify a relative path, then the path is relative to the <code>Scenarios</code> folder of the current project.</p> <p>The file name specified for <code>file_path</code> must either end with the <code>.rrscenario</code> extension or have no extension. If the input has no extension, then RoadRunner appends the <code>.rrscenario</code> extension to the specified value before loading the scenario.</p>

Name	Type	Description
keep_current_scene (optional)	bool	<p>Option to load the scenario into the current scene. If you set keep_current_scene to true and there is no current scene, the LoadScenario method call fails.</p> <p>If you set keep_current_scene to false, then RoadRunner loads the scene that the scenario was previously saved with and loads the scenario into that scene.</p> <p>Default: false</p>

Response

LoadScenarioResponse — Load scenario response

empty message

Load scenario response, returned as an empty message.

Sample Calls

Command Line

Load the prebuilt TrajectoryCutIn scenario from the project located at C:\RR\MyProject.

```
cd "C:\Program Files\RoadRunner R2022b\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject
CmdRoadRunnerApi "LoadScenario(file_path='TrajectoryCutIn')"
```

This sample call uses the CmdRoadRunnerApi helper command, which is a precompiled version of the RoadRunner API service. For examples that use this command, see:

- “Generate Scenario Variations Using gRPC API”
- “Reuse Scenarios in Multiple Scenes Using gRPC API”
- “Export Multiple Scenarios Using gRPC API”

Python

Load the prebuilt TrajectoryCutIn scenario from the current project.

```
loadScenarioRequest = roadrunner_service_messages_pb2.LoadScenarioRequest()
loadScenarioRequest.file_path = "TrajectoryCutIn"
api.LoadScenario(loadScenarioRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, api is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Load the prebuilt TrajectoryCutIn scenario from the current project.

```
LoadScenarioRequest request;
std::string filePath = "TrajectoryCutIn";
request.set_file_path(filePath);
ClientContext context;
LoadScenarioResponse reply;
Status status = api->LoadScenario(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2022a

See Also

NewProject | LoadProject | SaveProject | NewScene | LoadScene | SaveScene | NewScenario | SaveScenario | SetScenarioVariable | PrepareSimulation | SimulateScenario | Export | Import | Exit | roadrunner_service.proto | roadrunner_service_messages.proto

Topics

“Control RoadRunner Programmatically Using gRPC API”

SaveScenario

Save RoadRunner scenario using gRPC

Description

The `SaveScenario` method saves a specified RoadRunner scenario. RoadRunner also saves the current scene and project.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

SaveScenarioRequest — Save scenario request

message

Save scenario request, specified as a message with this field.

Name	Type	Description
file_path (optional)	string	<p>Absolute or relative path to the scenario file to save. If you specify a relative path, then the path is relative to the Scenarios folder of the current project.</p> <p>RoadRunner saves the scenario to the path specified by file_path. If you do not specify file_path, then RoadRunner saves the current scenario. If you do not have a current scenario loaded or have not previously saved the current scenario, then RoadRunner returns an error.</p> <p>RoadRunner recursively creates any folders missing from the file path.</p> <p>The file name specified for file_path must either end with the .rrscenario extension or have no extension. If the input has no extension, then RoadRunner appends the .rrscenario extension to the specified value before loading the scenario.</p>

Response

SaveScenarioResponse — Save scenario response

empty message

Save scenario response, returned as an empty message.

Sample Calls

Command Line

Save the current scenario to the Scenarios folder of project C:\RR\MyProject and name the scenario MyScenario.

```
cd "C:\Program Files\RoadRunner R2022b\bin\win64"
AppRoadRunner -projectPath C:\RR\MyProject
CmdRoadRunnerApi "SaveScenario(file_path='MyScenario')"
```

This sample call uses the CmdRoadRunnerApi helper command, which is a precompiled version of the RoadRunner API service. For examples that use this command, see:

- “Generate Scenario Variations Using gRPC API”
- “Reuse Scenarios in Multiple Scenes Using gRPC API”
- “Export Multiple Scenarios Using gRPC API”

Python

Save the current scenario to the `Scenarios` folder of the current project and name the scenario `MyScenario`.

```
saveScenarioRequest = roadrunner_service_messages_pb2.SaveScenarioRequest()
saveScenarioRequest.file_path = "MyScenario"
api.SaveScenario(saveScenarioRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Save the current scenario to the `Scenarios` folder of the current project and name the scenario `MyScenario`.

```
SaveScenarioRequest request;
std::string filePath = "MyScenario";
request.set_file_path(filePath);
ClientContext context;
SaveScenarioResponse reply;
Status status = api->SaveScenario(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2022a

See Also

[NewProject](#) | [LoadProject](#) | [SaveProject](#) | [NewScene](#) | [LoadScene](#) | [SaveScene](#) | [NewScenario](#) | [LoadScenario](#) | [SetScenarioVariable](#) | [PrepareSimulation](#) | [SimulateScenario](#) | [Export](#) | [Import](#) | [Exit](#) | [roadrunner_service.proto](#) | [roadrunner_service_messages.proto](#)

Topics

“Control RoadRunner Programmatically Using gRPC API”

GetScenarioVariable

Get RoadRunner Scenario variable using gRPC

Description

The `GetScenarioVariable` method gets the value of a specified variable in the current scenario. Variables that you can get are in the **Variables** table of the scenario. For more details on defining scenario variables, see “Generate Scenario Variations Using gRPC API”.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

`GetVariableRequest` — Get variable request

message

Get variable request, specified as a message with this field.

Name	Type	Description
name (required)	string	Name of the variable to set.

Response

`GetVariableResponse` — Get variable response

string

Get variable response, returned as a string of the variable value.

Sample Calls

Command Line

Get the `Vehicle1_InitialSpeed` variable value in a scenario named `MyScenario`. The scenario is located in the project `C:\RR\MyProject`.

```
cd "C:\Program Files\RoadRunner R2022b\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject
CmdRoadRunnerApi "LoadScenario(file_path='MyScenario')"
CmdRoadRunnerApi "GetScenarioVariable(name='Vehicle1_InitialSpeed')"
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner API service. For examples that use this command, see:

- “Generate Scenario Variations Using gRPC API”
- “Reuse Scenarios in Multiple Scenes Using gRPC API”
- “Export Multiple Scenarios Using gRPC API”

Python

Get the `Vehicle1_InitialSpeed` variable value in the current scenario.

```
getVariableRequest = roadrunner_service_messages_pb2.GetVariableRequest()
getVariableRequest.name = "Vehicle1_InitialSpeed"
response = api.GetScenarioVariable(getVariableRequest)
print(response.value)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Get the `Vehicle1_InitialSpeed` variable value in the current scenario.

```
GetVariableRequest request;
std::string name = "Vehicle1_InitialSpeed";
request.get_name(name);
ClientContext context;
GetVariableResponse response;
Status status = api->Import(&context, request, &response);
std::cout << response.value() << std::endl;
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2022a

See Also

[NewProject](#) | [LoadProject](#) | [SaveProject](#) | [NewScene](#) | [LoadScene](#) | [SaveScene](#) | [NewScenario](#) | [LoadScenario](#) | [SaveScenario](#) | [PrepareSimulation](#) | [SimulateScenario](#) | [Export](#) | [Import](#) | [Exit](#) | [roadrunner_service.proto](#) | [roadrunner_service_messages.proto](#)

Topics

“Control RoadRunner Programmatically Using gRPC API”

SetScenarioVariable

Set RoadRunner scenario variable using gRPC

Description

The `SetScenarioVariable` method sets a specified variable in the current scenario to the specified value. Variables that you can set are in the **Variables** table of the scenario. For more details on defining scenario variables, see “Generate Scenario Variations Using gRPC API”.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

SetVariableRequest — Set variable request

message

Set variable request, specified as a message with these fields.

Name	Type	Description
name (required)	string	Name of the variable to set.
value (required)	string	New value to assign to the variable. Even if the variable value is numeric, you must specify <code>value</code> as a string.

Response

SetVariableResponse — Set variable response

empty message

Set variable response, returned as an empty message.

Sample Calls

Command Line

Set the `Vehicle1_InitialSpeed` variable in a scenario named `MyScenario` to 10. The scenario is located in the project `C:\RR\MyProject`.

```
cd "C:\Program Files\RoadRunner R2022b\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject
CmdRoadRunnerApi "LoadScenario(file_path='MyScenario')"
CmdRoadRunnerApi "SetScenarioVariable(name='Vehicle1_InitialSpeed' value='10')"
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner API service. For examples that use this command, see:

- “Generate Scenario Variations Using gRPC API”
- “Reuse Scenarios in Multiple Scenes Using gRPC API”
- “Export Multiple Scenarios Using gRPC API”

Python

Set the `Vehicle1_InitialSpeed` variable in the current scenario to 10.

```
setVariableRequest = roadrunner_service_messages_pb2.SetVariableRequest()
setVariableRequest.name = "Vehicle1_InitialSpeed"
setVariableRequest.value = "10"
api.SetScenarioVariable(setVariableRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Set the `Vehicle1_InitialSpeed` variable in the current scenario to 10.

```
SetVariableRequest request;
std::string name = "Vehicle1_InitialSpeed";
request.set_name(name);
string value = "10";
request.set_value(value);
ClientContext context;
SetVariableResponse reply;
Status status = api->Import(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2022a

See Also

`NewProject` | `LoadProject` | `SaveProject` | `NewScene` | `LoadScene` | `SaveScene` | `NewScenario` | `LoadScenario` | `SaveScenario` | `PrepareSimulation` | `SimulateScenario` | `Export` | `Import` | `Exit` | `roadrunner_service.proto` | `roadrunner_service_messages.proto`

Topics

“Control RoadRunner Programmatically Using gRPC API”

PrepareSimulation

Prepare RoadRunner simulation for scenario simulation engine using gRPC

Description

The `PrepareSimulation` method submits simulation data, such as the scenario and map definitions, to the scenario simulation engine (SSE). Call this method to make the simulation data available to be queried from MATLAB or Simulink, or from external simulators such as CARLA, without having to simulate the scenario.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

PrepareSimulationRequest — Prepare simulation request

empty message

Prepare simulation request, specified as an empty message.

Response

PrepareSimulationResponse — Prepare simulation response

empty message

Prepare simulation response, returned as an empty message.

Sample Calls

Command Line

Prepare simulation data in the current scenario for use with the SSE. The scenario is located in the project `C:\RR\MyProject`.

```
cd "C:\Program Files\RoadRunner R2022b\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject
CmdRoadRunnerApi "PrepareSimulation()"
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner API service. For examples that use this command, see:

- “Generate Scenario Variations Using gRPC API”
- “Reuse Scenarios in Multiple Scenes Using gRPC API”

- “Export Multiple Scenarios Using gRPC API”

Python

Prepare simulation data in the current scenario for use with the SSE.

```
prepareSimulationRequest = roadrunner_service_messages_pb2.PrepareSimulationRequest()
api.PrepareSimulation(prepareSimulationRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Prepare simulation data in the current scenario for use with the SSE.

```
PrepareSimulationRequest request;
ClientContext context;
PrepareSimulationResponse reply;
Status status = api->PrepareSimulation(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2022a

See Also

[NewProject](#) | [LoadProject](#) | [SaveProject](#) | [NewScene](#) | [LoadScene](#) | [SaveScene](#) | [NewScenario](#) | [LoadScenario](#) | [SaveScenario](#) | [SetScenarioVariable](#) | [SimulateScenario](#) | [Export](#) | [Import](#) | [Exit](#) | [roadrunner_service.proto](#) | [roadrunner_service_messages.proto](#)

Topics

“Control RoadRunner Programmatically Using gRPC API”

SimulateScenario

Simulate RoadRunner scenario using gRPC

Description

The `SimulateScenario` method simulates the current scenario. By default, all other calls to the RoadRunner application are blocked until the simulation is over.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

SimulateScenarioRequest – Simulate scenario request message

Simulate scenario request, specified as a message with these fields.

Name	Type	Description
<p> pacing (optional)</p>	<p> google.protobuf.DoubleValue</p>	<p> Simulation pacing to control how fast the simulation runs, specified as a nonnegative number.</p> <ul style="list-style-type: none"> • A value between 0 and 1 is slower than real time. • A value of 1 equates to real time. • A value greater than 1 is faster than real time. <p> If you omit this value, then the simulation runs as fast as possible, given the performance of the CPU and the complexity of the scenario.</p>

Name	Type	Description
simulation_end_time (optional)	google.protobuf.DoubleValue	Maximum amount of time, in seconds, that the simulation runs. The simulation ends when it reaches simulation_end_time, if it does not stop earlier. If you omit this value, then the simulation runs until it stops for another reason, such as if an end condition is met, a collision occurs, or you manually stop the simulation.
blocking (optional)	google.protobuf.BoolValue	Option to block calls to the RoadRunner application during simulation. If true, then calls to RoadRunner are blocked until the simulation ends. If false, you can make calls to the RoadRunner application immediately after starting the simulation. Default: true

Response

SimulateScenarioResponse — Simulate scenario response

empty message

Simulate scenario response, returned as an empty message.

Sample Calls

Command Line

Simulate the current scenario and slow down the simulation to 50% of real-time speed. The scenario is located in the project C:\RR\MyProject.

```
cd "C:\Program Files\RoadRunner R2022b\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject
CmdRoadRunnerApi "SimulateScenario(pacing.value='0.5')"
```

This sample call uses the CmdRoadRunnerApi helper command, which is a precompiled version of the RoadRunner API service. For examples that use this command, see:

- “Generate Scenario Variations Using gRPC API”
- “Reuse Scenarios in Multiple Scenes Using gRPC API”
- “Export Multiple Scenarios Using gRPC API”

Python

Simulate the current scenario and slow down the simulation to 50% of real-time speed.

```
simulateScenarioRequest = roadrunner_service_messages_pb2.SimulateScenarioRequest()
simulateScenarioRequest.pacing.value = 0.5
api.SimulateScenario(simulateScenarioRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Simulate the current scenario and slow down the simulation to 50% of real-time speed.

```
SimulateScenarioRequest request;
request.mutable_pacing()->set_value(0.5);
ClientContext context;
SimulateScenarioResponse reply;
Status status = api->SimulateScenario(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2022a

See Also

NewProject | LoadProject | SaveProject | NewScene | LoadScene | SaveScene | NewScenario | LoadScenario | SaveScenario | SetScenarioVariable | PrepareSimulation | Export | Import | Exit | roadrunner_service.proto | roadrunner_service_messages.proto

Topics

“Control RoadRunner Programmatically Using gRPC API”

Objects

createSimulation

Create RoadRunner Scenario simulation using MATLAB

Syntax

```
rrSim = createSimulation(rrApp)
```

Description

`rrSim = createSimulation(rrApp)` creates and returns a scenario simulation object for the current scenario.

Examples

Create Scenario Simulation

Create a scenario simulation object in RoadRunner Scenario using MATLAB.

Call the `roadrunner` function and pass in the location where you want to create the project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located at "C:\RR\MyProject". This call returns an object `rrApp` that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder);
```

Open an existing scenario in RoadRunner Scenario by calling the `openScenario` function and passing it the `rrApp` object and the specific scenario filename that you want to open. This call opens the desired scenario in the RoadRunner Scenario application through MATLAB.

```
filename = "MyScenario.rrscenario";  
openScenario(rrApp, filename);
```

Create a scenario simulation object by calling the `createSimulation` function and passing it the `rrApp` object. This call returns a simulation object `rrSim` for the current scenario, through which you can control the simulation programatically.

```
rrSim = createSimulation(rrApp);
```

```
Connection status: 1
```

```
Connected to RoadRunner Scenario server on localhost:61720, with client id {fb457314-9501-4395-a
```

Input Arguments

rrApp — RoadRunner application

`roadrunner` object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

Output Arguments

rrSim — RoadRunner Scenario simulation object

`Simulink.ScenarioSimulation` object

RoadRunner Scenario simulation object, specified as a `Simulink.ScenarioSimulation` object. This object enables you to control a simulation, access and modify the runtime parameters of a simulation, and report custom diagnostic messages during a simulation.

Version History

Introduced in R2022a

See Also

`roadrunner` | `Simulink.ScenarioSimulation` | `getScenarioVariable` | `setScenarioVariable` | `close`

Topics

“RoadRunner Scenario Fundamentals”

“Simulate a RoadRunner Scenario Using MATLAB Functions”

exportScenario

Export scenario from RoadRunner Scenario using MATLAB

Syntax

```
exportScenario(rrApp, filename, formatname)
exportScenario(rrApp, filename, formatname, exportoptions)
```

Description

`exportScenario(rrApp, filename, formatname)` exports a scenario file to one of the file formats that RoadRunner supports.

`exportScenario(rrApp, filename, formatname, exportoptions)` sets options for export using `exportoptions`. The export options configuration is specified as one of the export options objects compatible with the format name specified in the `formatname` argument. You can export only ASAM OpenSCENARIO files.

Examples

Export Scenario

Export a scenario from RoadRunner Scenario using MATLAB.

Call the `roadrunner` function and pass in the location where you want to create the project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located at "C:\RR\MyProject". This call returns an object `rrApp` that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";
rrApp = roadrunner(projectFolder);
```

Open a scenario in the project by calling the `openScenario` function. You must pass the `rrApp` object and the RoadRunner scenario you wish to open as input arguments while calling the `openScenario` function. This example uses the 'FourWaySignal.rrscenario' scenario, which is one of the scenario included by default in RoadRunner projects and is located in the `Scenarios` folder of the project.

```
scenarioname = "MyScenario.rrscenario";
openScenario(rrApp, scenarioname);
```

Before exporting the file, set export options by creating an `openScenarioExportOptions` object to enable export of signals and objects from the file.

```
options = openScenarioExportOptions(OpenDriveOptions=openDriveExportOptions(OpenDriveVersion = 1
```

Once the scenario opens successfully, call the `exportScenario` function to export the scenario to ASAM OpenSCENARIO®. Pass `rrApp`, the scenario filename of the scenario, the export format, and the export options as input arguments to the function.

```
filename = "FourWaySignal.xosc";
formatname = "OpenSCENARIO";
exportScenario(rrApp, filename, formatname, options);
```

Input Arguments

rrApp — RoadRunner application

roadrunner object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

filename — File path to the exported file

character vector | string scalar

File path to the exported file, specified as a character vector or string scalar. `filename` is absolute or relative to the exported file. If you specify a relative path, then the exported file is saved relative to the `Exports` folder of the current project. If any folders in the path are missing, RoadRunner tries to create them. `filename` can include the extension for the exported file or have no extension. If it has no extension, then RoadRunner appends the extension of the format specified by the `formatname` to the file name before exporting the scenario.

Example: While calling

```
exportScenario(rrApp, "FourWaySignal.xosc", "OpenSCENARIO", options),
```

"FourWaySignal.xosc" represents the file name of the exported file, which is relative to the `Exports` folder of the current project.

Data Types: char | string

formatname — Export format name

character vector | string scalar

Export format name, specified as a character vector or string scalar. This argument specifies the export format name corresponding to a valid RoadRunner export format. Format name options are case-insensitive. RoadRunner only supports ASAM OpenSCENARIO format.

Example: While calling

```
exportScenario(rrApp, "FourWaySignal.xosc", "OpenSCENARIO", options),
```

`OpenSCENARIO` specifies that the file will be exported to ASAM OpenSCENARIO format.

Data Types: char | string

exportoptions — Export options configuration

exportoptions object

Export options configuration, specified as one of the export options objects compatible with the format name specified in the `formatname` argument. This argument specifies the options that can be used with `export`. Only `openScenarioExportOptions` object is supported.

Export Format Options Object	Description	Properties	
<p>openScenarioExportOptions</p>	<p>Specifies options for exporting RoadRunner scene and scenario to ASAM OpenSCENARIO.</p> <p>openScenarioExportOptions (Name=Value) creates an export options configuration object for the ASAM OpenSCENARIO format with properties specified as one or more name-value arguments. If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	<p>SceneGraphFormatName</p>	<p>Format of scene graph file to export, specified as a string or character vector. Only OpenSceneGraph format is supported. Specify this option to export a new scene graph file for the scenario. To reuse an existing scene graph file, specify the file in the SceneGraphFileName property.</p> <p>Default: "auto"</p>
		<p>SceneGraphFileName</p>	<p>Name of previously exported scene graph file, specified as a string or character vector. Specify this option to reuse an existing scene, which can speed up the export process. If SceneGraphFormatName is specified, then this property is ignored and a new scene</p>

Export Format Options Object	Description	Properties	
			graph file is generated. Default: "auto"
		OpenSceneGraphOptions	Export options for the OpenSceneGraph file, specified as an openScenarioExportOptions object. Default: "auto"
		OpenDriveOptions	Export options for the ASAM OpenDRIVE® file, specified as an openDriveExportOptions object. Default: "auto".
		Example: options = openScenarioExportOptions(OpenDriveOptions = openDriveExportOptions(ExportObjects=true));	

Data Types: char | string

Version History

Introduced in R2022a

See Also

roadrunner | importScenario | openScenario | close

Topics

"RoadRunner Scenario Fundamentals"

"Simulate a RoadRunner Scenario Using MATLAB Functions"

getScenarioVariable

Get the value of RoadRunner scenario variable using MATLAB

Syntax

```
value = getScenarioVariable(rrApp,name)
```

Description

`value = getScenarioVariable(rrApp,name)` returns the value of a variable in the current scenario with the specified name.

Examples

Get Scenario Variable

Get a scenario variable in RoadRunner Scenario using MATLAB.

Call the `roadrunner` function and pass in the location where you want to create the project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located at "C:\RR\MyProject". This call returns an object `rrApp` that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder);
```

Open an existing scenario in RoadRunner Scenario by calling the `openScenario` function and passing it the `rrApp` object and the specific scenario filename that you want to open. This call opens the desired scenario in the RoadRunner Scenario application through MATLAB.

```
filename = "MyScenario.rrscenario";  
openScenario(rrApp,filename);
```

Get the value of a scenario variable. Call the `getScenarioVariable` function and pass it the `rrApp` object and the variable whose value you want to retrieve. For example, this call retrieves a value of 17.88 as the initial speed of the Ambulance in the scenario, `MyScenario`. This example has an existing variable, `Ambulance_InitialSpeed`.

```
name = "Ambulance_InitialSpeed";  
value = getScenarioVariable(rrApp,name);
```

Input Arguments

rrApp — RoadRunner application

roadrunner object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

name — Name of variable to retrieve value

character vector | string scalar

Name of variable to retrieve value, specified as a character vector or string scalar. If the specified variable does not exist in the scenario, it results in an error. For more details on creating variables in a scenario, see “Generate Scenario Variations”.

Example: `getScenarioVariable(rrApp, "Ambulance_InitialSpeed");` retrieves the initial speed of the Ambulance vehicle.

Output Arguments

value — Value assigned to variable

character vector | string scalar

Value assigned to variable, returned as a character vector or string scalar.

Example: `getScenarioVariable(rrApp, "Ambulance_InitialSpeed");` retrieves the initial speed as 17.88 m/s assigned of the Ambulance vehicle.

Version History

Introduced in R2022a

See Also

`roadrunner` | `setScenarioVariable` | `close`

Topics

“RoadRunner Scenario Fundamentals”

“Simulate a RoadRunner Scenario Using MATLAB Functions”

importScenario

Import file into RoadRunner Scenario using MATLAB

Syntax

```
importScenario(rrApp, filename, formatname)
importScenario(rrApp, filename, formatname, importoptions)
```

Description

`importScenario(rrApp, filename, formatname)` imports a file that is in a format that RoadRunner supports into the currently opened scenario.

`importScenario(rrApp, filename, formatname, importoptions)` sets options for import using `importoptions` argument. You can import only ASAM OpenSCENARIO files.

Examples

Import Scenario

Import a scenario in RoadRunner Scenario using MATLAB.

Call the `roadrunner` function and pass in the location where you want to create the project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located at "C:\RR\MyProject". This call returns an object `rrApp` that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";
rrApp = roadrunner(projectFolder);
```

Open a new scenario in the current project by calling the `newScenario` function and passing it the `rrApp` object. This call opens a blank scenario in the currently opened project.

```
newScenario(rrApp);
```

Before importing the ASAM OpenSCENARIO® file, set import options by creating an `openScenarioImportOptions` object to enable import of signals from the file.

```
options = openScenarioImportOptions(OpenDriveOptions = openDriveImportOptions(ImportSignals=true
```

Call the `importScenario` function and pass the `rrApp` object, the filename, and the `options` object as input arguments. This function call imports data from the specified filename into the currently opened scenario.

```
filename = "C:\RR\MyProject\Assets\FourWaySignal.xosc";
formatname = "OpenSCENARIO";
importScenario(rrApp, filename, formatname, options);
```

Input Arguments

rrApp — RoadRunner application

roadrunner object

RoadRunner application associated with a project, specified as a roadrunner object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. rrApp provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

filename — Path of file which is to be imported

character vector | string scalar

Path of file which is to be imported, specified as a character vector or string scalar. filename is absolute or relative path to the file to be imported. If you specify a relative path, then you must specify a path to a file in the **Assets** folder of the current project.

Example: While calling `importScenario(rrApp, "C:\RR\MyProject\Assets\FourWaySignal.xosc", "OpenSCENARIO", importoptions)`, "C:\RR\MyProject\Assets\FourWaySignal.xosc" represents the file path of the file to be imported, which is relative to the Assets folder of the current project.

Data Types: char | string

formatname — Import format name

character vector | string scalar

Import format name, specified as a character vector or string scalar. This argument specifies the format name corresponding to a valid import format that RoadRunner supports. Format name options are case-insensitive. You can import only ASAM OpenSCENARIO files.

Example: While calling `importScenario(rrApp, "C:\RR\MyProject\Assets\FourWaySignal.xosc", "OpenSCENARIO", importoptions)`, OpenSCENARIO specifies that the file will be imported to ASAM OpenSCENARIO format.

Data Types: char | string

importoptions — Import options configuration

openScenarioImportOptions object

Import options configuration, specified as openScenarioImportOptions object compatible with the file specified in the filename argument. Only openScenarioImportOptions object for ASAM OpenSCENARIO file is supported.

Import Format Options Object	Description	Properties	
<p>openScenarioImportOptions</p>	<p>Specifies options for importing ASAM OpenSCENARIO file into RoadRunner scenario.</p> <p>openScenarioImportOptions (Name=Value) creates an import options configuration object for the ASAM OpenSCENARIO format with properties specified as one or more name-value arguments. If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	<p>OpenDriveOptions</p>	<p>Options to import an ASAM OpenDRIVE file, specified as an openDriveImportOptions object.</p> <p>Default: "auto".</p>
		<p>Example: options = openScenarioImportOptions(OpenDriveOptions = openDriveImportOptions(ImportSignals=true));</p>	

Data Types: char | string

Version History

Introduced in R2022a

See Also

roadrunner | exportScenario | close

Topics

“RoadRunner Scenario Fundamentals”

“Simulate a RoadRunner Scenario Using MATLAB Functions”

newScenario

Create new scenario in RoadRunner Scenario using MATLAB

Syntax

```
newScenario(rrApp)
```

Description

`newScenario(rrApp)` creates a new scenario in the current RoadRunner scene. You must create or load a new scene before creating a new scenario.

Examples

Create New RoadRunner Scenario

Create a new scenario in RoadRunner Scenario using MATLAB.

Start RoadRunner and open the project called "C:\RR\MyProject". Return the RoadRunner application object in `rrApp`. This example assumes that RoadRunner is installed in its default location in Windows. This call returns an object `rrApp` that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder);
```

Open an existing scene in RoadRunner. For example, open the `FourWaySignal` scene, which is one of the scenes included by default in RoadRunner projects.

```
filename = "FourWaySignal.rrscene";  
openScene(rrApp, filename);
```

Create a new, empty scenario in the scene using the `newScenario` function.

```
newScenario(rrApp);
```

Input Arguments

rrApp — RoadRunner application

`roadrunner` object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

Version History

Introduced in R2022a

See Also

roadrunner | openScenario | saveScenario | openScene | close

Topics

“RoadRunner Scenario Fundamentals”

“Simulate a RoadRunner Scenario Using MATLAB Functions”

openScenario

Open scenario in RoadRunner Scenario using MATLAB

Syntax

```
openScenario(rrApp, filename)
openScenario(rrApp, filename, keepCurrentScene)
```

Description

`openScenario(rrApp, filename)` opens the specified scenario in the RoadRunner scene in which it was previously saved. If no current scene is specified in the project, MATLAB returns an error.

`openScenario(rrApp, filename, keepCurrentScene)` specifies whether to open the specified scenario in the scene it was previously saved with or in the current scene, regardless of which scene it was previously saved with.

Examples

Open Scenario

Open a scenario in RoadRunner Scenario using MATLAB.

Start RoadRunner and open the project called "C:\RR\MyProject". Return the RoadRunner application object in `rrApp`. This example assumes that RoadRunner is installed in its default location in Windows. This call returns an object `rrApp` that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";
rrApp = roadrunner(projectFolder);
```

Open an existing scenario in RoadRunner Scenario by calling the `openScenario` function and passing it the `rrApp` object and the specific scenario `filename` that you want to open. This call opens the desired scenario in the RoadRunner Scenario application through MATLAB.

```
filename = "MyScenario.rscenario";
openScenario(rrApp, filename);
```

Input Arguments

rrApp — RoadRunner application

roadrunner object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

filename — RoadRunner scenario file name

character vector | string scalar

RoadRunner scenario file name, specified as a character vector or string scalar. If you specify this argument as a relative path, then the path is relative to the `Scenarios` folder of the current project. `filename` must end with either the `.rrscenario` extension or have no extension. If it has no extension, then RoadRunner appends the `.rrscenario` extension to `filename` before opening the scenario.

Example: `openScenario(rrApp, "MyScenario.rrscenario")` opens `MyScenario.rrscenario` from the `Scenarios` folder of the current project.

Data Types: `char` | `string`**keepCurrentScene — Open scenario in current scene**`false` or `0` (default) | `true` or `1`

Open scenario in current scene, specified as a numeric or logical `0` (`false`) or `1` (`true`). If you set `keepCurrentScene` to `true` and there is no current scene, then MATLAB returns an error. If you set `keepCurrentScene` to `false`, then RoadRunner opens the scene that the scenario was previously saved with and opens the scenario into that scene.

Data Types: `logical`

Version History

Introduced in R2022a

See Also

`roadrunner` | `newScenario` | `saveScenario` | `close`

Topics

"RoadRunner Scenario Fundamentals"

"Simulate a RoadRunner Scenario Using MATLAB Functions"

remapAnchor

Remap road anchor in RoadRunner Scenario in MATLAB

Syntax

```
remapAnchor(rrApp, source, target)
remapAnchor(rrApp, source, target, NewAnchorName=newName)
```

Description

`remapAnchor(rrApp, source, target)` remaps dependencies from a source anchor in the scenario to a different target road anchor. You can specify the name of the source anchor and the name or position of the target anchor.

`remapAnchor(rrApp, source, target, NewAnchorName=newName)` also sets the name of the target anchor when the target is specified as a position.

Examples

Remap Anchors in a Scenario

Remap road anchor to a target anchor

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";
rrApp = roadrunner(projectFolder, InstallationFolder='C:\Program Files\RoadRunner R2022b\bin\win
```

Open an existing scenario in RoadRunner Scenario by calling the `openScenario` function and passing it the `rrApp` object and the specific scenario filename that you want to open. This call opens the desired scenario in the RoadRunner Scenario application through MATLAB.

```
filename = "TrajectoryCutIn.rrscenario";
openScenario(rrApp, filename);
```

Remap the source anchor to the target anchor by calling the `remapAnchor` function and passing it the `rrApp` object and the source anchor name, the target anchor name and the new anchor name.

```
remapAnchor(rrApp, "ScenarioStart", [-20 0 0], NewAnchorName="anchor3");
```

Input Arguments

rrApp — RoadRunner application

roadrunner object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

source — Source anchor in scenario

character vector | string scalar

Source anchor in scenario, specified as a character vector or string scalar.

target — Target anchor in scenario

character vector | string scalar | three-element numeric vector

Target anchor in scenario, specified as one of these values.

- A character vector or string scalar indicating the name of the target anchor. A name remap searches for a road anchor with matching name in the scenario and scene.
- A three-element numeric vector indicating the position of the target anchor relative to the center of the scene. A position remap creates a new road anchor on the road closest to the provided position. The distance between this position and the closest road in the scene must be greater than 25 meters.

newName — New anchor name for position remap

"" (default) | character vector | string scalar

New anchor name for a position remap, specified as a character vector or string scalar. If the value is not specified, the `remapAnchor` function uses the default value.

Version History

Introduced in R2022b

See Also

`roadrunner` | `getScenarioVariable`

Topics

“RoadRunner Scenario Fundamentals”

“Simulate a RoadRunner Scenario Using MATLAB Functions”

saveScenario

Save scenario in RoadRunner Scenario using MATLAB

Syntax

```
saveScenario(rrApp)
saveScenario(rrApp, filename)
```

Description

`saveScenario(rrApp)` saves the current scenario.

`saveScenario(rrApp, filename)` saves the current scenario to the specified file `filename`. RoadRunner also saves the current scene and project.

Examples

Save Scenario

Save a scenario in RoadRunner Scenario using MATLAB.

Start RoadRunner and open the project called "C:\RR\MyProject". Return the RoadRunner application object in `rrApp`. This example assumes that RoadRunner is installed in its default location in Windows. This call returns an object `rrApp` that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";
rrApp = roadrunner(projectFolder);
```

Open an existing scenario in RoadRunner Scenario by calling the `openScenario` function and passing it the `rrApp` object and the specific scenario `filename` that you want to open. This call opens the desired scenario in the RoadRunner Scenario application through MATLAB.

```
filename = "MyScenario.rrscenario";
openScenario(rrApp, filename);
```

Save the scenario to another file. Call the `saveScenario` function and pass it the `rrApp` object and the new `filename` with which you want to save the scenario. This call saves the scenario, the current scene and the project.

```
newFilename = "MyScenario1.rrscenario";
saveScenario(rrApp, newFilename);
```

Input Arguments

rrApp — RoadRunner application

roadrunner object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

filename — RoadRunner scenario file name

character vector | string scalar

RoadRunner scenario file name, specified as a character vector or string scalar. If you specify this argument as a relative path, then the path is relative to the `Scenarios` folder of the current project. If you do not specify `filename`, then RoadRunner saves the current scenario to its existing file. If you do not have a current scenario open, then RoadRunner returns an error. RoadRunner recursively creates any folders on the specified file path that do not already exist.

`filename` must end with `.rrscenario` extension or have no extension. If it has no extension, then RoadRunner appends the `.rrscenario` extension to the file name before saving the scenario. If the file being saved already exists, then RoadRunner overwrites it.

Example: `saveScenario(rrApp, "MyScenario1.rrscenario")` saves the current open scenario to the `MyScenario1.rrscenario` file in to the `Scenarios` folder of the current project.

Data Types: `char` | `string`

Version History

Introduced in R2022a

See Also

`roadrunner` | `newScenario` | `openScenario` | `close`

Topics

“RoadRunner Scenario Fundamentals”

“Simulate a RoadRunner Scenario Using MATLAB Functions”

setScenarioVariable

Set RoadRunner scenario variable using MATLAB

Syntax

```
setScenarioVariable(rrApp,name,value)
```

Description

`setScenarioVariable(rrApp,name,value)` sets a specified variable in the current scenario to the specified value.

Examples

Set Scenario Variable

Set a scenario variable in RoadRunner Scenario using MATLAB.

Call the `roadrunner` function and pass in the location where you want to create the project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located at "C:\RR\MyProject". This call returns an object `rrApp` that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder);
```

Open an existing scenario in RoadRunner Scenario by calling the `openScenario` function and passing it the `rrApp` object and the specific scenario filename that you want to open. This call opens the desired scenario in the RoadRunner Scenario application through MATLAB.

```
filename = "MyScenario.rrscenario";  
openScenario(rrApp,filename);
```

Set the value of a scenario variable. Call the `setScenarioVariable` function and pass it the `rrApp` object, the variable whose value you want to set, and the value that you want to assign to the variable. For example, this call assigns a value of 17.88 as the initial speed of the Ambulance in the scenario, `MyScenario`.

```
name = "Ambulance_InitialSpeed";  
value = "17.88";  
setScenarioVariable(rrApp,name,value);
```

Input Arguments

rrApp — RoadRunner application

`roadrunner` object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

name — Name of variable to set

character vector | string scalar

Name of variable to set, specified as a character vector or string scalar. If the specified variable does not exist in the scenario, it results in an error. For more details on creating variables in a scenario, see “Generate Scenario Variations”.

Example: `setScenarioVariable(rrApp,"Ambulance_InitialSpeed","17.88");` sets the initial speed of the Ambulance vehicle.

value — Value assigned to variable

character vector | string scalar

Value assigned to variable, specified as a character vector or string scalar. This argument specifies the new value to assign to the variable. Even if the variable value is numeric, you must specify `value` as a character vector or string scalar.

Example: `setScenarioVariable(rrApp,"Ambulance_InitialSpeed","17.88");` sets the initial speed of the Ambulance to 17.88 m/s.

Version History

Introduced in R2022a

See Also

`roadrunner` | `getScenarioVariable` | `close`

Topics

“RoadRunner Scenario Fundamentals”

“Simulate a RoadRunner Scenario Using MATLAB Functions”

Configurations

Simulation Configuration

Specify simulation constants, timeout values, and cosimulation platform parameters

RoadRunner Scenario specifies simulation settings in an XML configuration file, `SimulationConfiguration.xml`.

With the configuration file, you can modify the properties of the simulation and the MATLAB and RoadRunner Scenario cosimulation bridge. These are the default paths for this configuration file on Windows® and Linux:

- Windows — `C:\Users\username\AppData\Roaming\MathWorks\RoadRunner\R20NNx\Scenario\Config\`
- Linux — `~/.local/share/MathWorks/RoadRunner/R20NNx/Scenario/Config/`

username is your Windows user ID. *R20NNx* is the version of MATLAB you currently have installed.

Note Modifying the `SimulationConfiguration.xml` requires that you restart RoadRunner Scenario to apply the changes.

Parameters

Timeout Values

These are the default timeout values used by the gRPC server for each event type, specified in milliseconds.

Event Type	Timeout Value
<code>SimulationStartEvent</code>	30000
<code>SimulationStepEvent</code>	6000
<code>SimulationPostStepEvent</code>	6000
<code>SimulationStopEvent</code>	6000
<code>CreateActorEvent</code>	60000
<code>DestroyActorEvent</code>	10000

If the client proposes a timeout value in its `profile` attribute, then the greater of the proposed value and value in this table is used. The minimum timeout value on the RoadRunner Scenario server side is 2000 ms; any lower value is equated to 2000 ms by the server.

If you set a timeout value using the MATLAB `settings` function, then for each event type, the greater of the proposed value and value in this table is used. For example, if you set the timeout as 10 seconds through the `settings` function, then the timeout for `SimulationStartEvent` remains 30 seconds. However, the timeout values for `SimulationStepEvent`, `SimulationPostStepEvent`, and `SimulationStopEvent` increase from six seconds to 10 seconds.

Network Ports

Port — Scenario Simulation gRPC server network port

35706 (default) | positive integer in range [1024, 65535]

Network port the Scenario Simulation gRPC server listens to.

Example: `<Port name="ScenarioServer" value="35706"/>`

Data Types: double

Co-Simulation Platforms

Platform — Platform name

MATLAB | CARLA

Name of the platform executable.

ExecutablePath — Full path to platform executable

string scalar

Full path to the platform executable.

Example: `<ExecutablePath>C:\Program Files\MATLAB\R2022a\matlab\bin\matlab.exe</ExecutablePath>`

StartTimeOut — Wait time for platform launch

nonnegative integer scalar

The duration RoadRunner will wait in millisecond to launch the platform

Example: `<StartTimeOut>60000</StartTimeOut>`

NoDesktop — Launch MATLAB in without desktop

true (default) | false

Open MATLAB in no desktop mode.

Example: `<NoDesktop>>true</NoDesktop>`

Simulation Constants

AlignmentAngleThreshold — Threshold angle to determine a vehicle's alignment on a lane

0.8028514559 (default) | positive scalar

Threshold angle to determine a the alignment of a vehicle in a lane, by comparing the angle formed from the vehicle heading and the lane tangent evaluated at the vehicle map location using this rule:

- Forward — angle $<$ AlignmentAngleThreshold
- Backward — angle $>$ π - AlignmentAngleThreshold
- Not Aligned — AlignmentAngleThreshold \leq angle \leq π - AlignmentAngleThreshold

Example: `<Parameter name="AlignmentAngleThreshold" value="0.8028514559"/>`

Data Types: double

LaneChangeActionActorAngleThreshold — Threshold angle to determine completion of a lane change action

0.0349 (default) | positive scalar

Threshold angle to determine the completion of a lane change action, by enforcing the vehicle heading to strictly align against the lane tangent evaluated at the vehicle map location.

Example: `<Parameter name="LaneChangeActionActorAngleThreshold" value="0.0349"/>`

Data Types: double

MaxSearchDistance — Maximum longitudinal distance for vehicle lane adjacency search

500 (default) | positive scalar

Maximum longitudinal distance used to search for the lane adjacency between two vehicles.

Example: `<Parameter name="MaxSearchDistance" value="500"/>`

Data Types: double

LateralOffsetComparisonTolerance — Tolerance to compare vehicle lateral offset

0.001 (default) | positive scalar

Tolerance, specified in meters, used when comparing the lateral offset of a vehicle to its desired value.

Example: `<Parameter name="LateralOffsetComparisonTolerance" value="0.001"/>`

Data Types: double

LaneMappingHeightTolerance — Tolerance to compare vehicle lateral offset

0.5 (default) | positive scalar

Tolerance, specified in meters, used when comparing the projected distance of a vehicle onto an on-lane candidate against the vehicle chassis height. This tolerance value is used for lane mapping, and is intended to narrow down the on-lane map locations for a multilayer, overlapping road network.

Example: `<Parameter name="LaneMappingHeightTolerance" value="0.5"/>`

Data Types: double

SpeedComparisonTolerance — Tolerance to compare vehicle speed against goal

0.1 (default) | positive scalar

Tolerance, specified in meters per second, used when comparing the speed of a vehicle to its goal speed or to the speed of another vehicle.

Example: `<Parameter name="SpeedComparisonTolerance" value="0.1"/>`

Data Types: double

MaxSimulationTime — Maximum time allowed per simulation

1000 (default) | positive integer

Maximum time, in seconds, to allow a simulation to run.

Example: `<Parameter name="SpeedComparisonTolerance" value="0.1"/>`

Data Types: uint64

SimulationStepSize — Simulation time update step size

0.02 (default) | positive scalar

The step size, in seconds, used to update the simulation.

Example: `<Parameter name="SimulationStepSize" value="0.02"/>`

Data Types: double

SimulationPacing — Simulation time pacing relative to real time

1 (default) | positive scalar

By default, the time step of the simulation pipeline tries to keep pace with the real world time. For example 0.01 seconds of simulation time occurs in 0.01 seconds in the real world. Simulation pacing enables the simulation time steps to occur either faster or slower than real world time. The rendering pipeline always updates at 60 frames per second (FPS). The impact of this parameter on the relative pacing between the simulation and real world time differs depending on its value.

- Less than one — The simulation runs slower than real-time, a value of 0.5 would be 0.5x real-time.

Note When the `SimulationPacing` is less than one, frames can be rendered twice, due to the rendering pipeline running more frequently than the simulation pipeline. This appears as jitter in the visualization.

- Equal to one — The simulation runs in real-time, at 60 FPS.
- Greater than one — The simulation runs faster than real-time. For example, a value of 2 would be 2x real-time.

Example: `<Parameter name="SimulationPacing" value="1">`

Data Types: double

Version History

Introduced in R2022a

See Also

