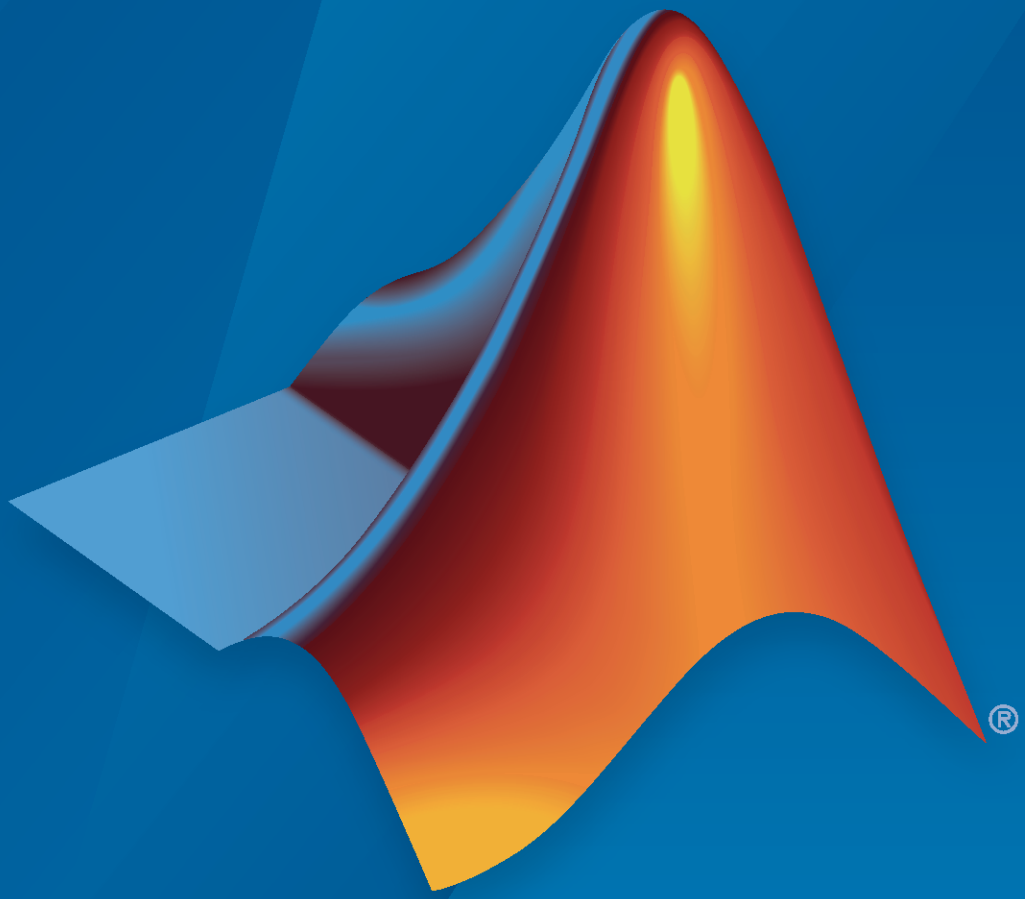
ROS Toolbox Release Notes



MATLAB® & SIMULINK®

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

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

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

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

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

ROS Toolbox Release Notes

Contents

R2022a

R2021b

R2021a

R2020b

R2020a

R2019b

R2022a

ROS Node Generation and Deployment

C++ Code Generation Support Enhancement for ROS: Generate ROS nodes for deploying to target hardware using MATLAB Coder

Key ROS features now support code generation using MATLAB® Coder™, including:

- `rostf`
- `ros.TransformationTree`
- `canTransform`
- `sendTransform`
- `waitForTransform`
- `getTransform`
- `transform`

For a list of functions that support code generation, and examples of their application, see “Code Generation and Deployment”. For details about code generation support and limitations for each function, see the **Extended Capabilities** section on the corresponding function reference page.

For code generation, you must use message structures instead of message objects. For more information, see “MATLAB Programming for Code Generation”.

For an example, see “Generate ROS Node for UAV Waypoint Follower”.

ros_control Controller Generation Support for ROS: Generate and deploy ros_control controller plugins from Simulink

You can now generate and deploy `ros_control` controller plugins from Simulink® models. To generate and deploy the plugin for a model:

- From the **ROS** tab, in the **Prepare** section, select **ROS Control Settings**.
- Configure the ROS Control interfaces for the model by specifying the resource names and types for the outports and inports.
- Connect to your ROS device. To generate and deploy the controller plugin, in the **ROS** tab, select **Deploy** then **Build & Load**.
- In the controller configuration YAML file for the robot on your ROS device, add the generated controller plugin information and rebuild the catkin workspace. For an example, see “Generate a ROS Control Plugin from Simulink®”.

ROS 2 Node Generation and Deployment

C++ Code Generation Support Enhancement for ROS 2: Generate ROS 2 nodes for deploying to target hardware using MATLAB Coder

Key ROS 2 features now support code generation using MATLAB Coder, including:

- `ros2svcserver`
- `ros2svcclient`
- `isServerAvailable`
- `waitForServer`
- `call`

For a list of functions that support code generation, and examples of their application, see “Code Generation and Deployment”. For details about code generation support and limitations for each function, see the **Extended Capabilities** section on the corresponding function reference page.

CUDA Optimized Code Generation Support for ROS 2: Deploy CUDA-optimized ROS 2 nodes from Simulink to target hardware using GPU Coder

You can now generate and deploy CUDA-optimized code for ROS 2 Nodes from Simulink models. For an example, see “Generate CUDA ROS 2 Node from Simulink”.

ROS Simulation

ROS Noetic Support: Use Noetic Ninjemys distribution for ROS

As of R2022a, ROS Toolbox supports the ROS Noetic Ninjemys distribution. A ROS distribution is a stable set of ROS packages that defines the supported platforms, message definitions, and new features for the release. In prior releases, ROS Toolbox supported the Melodic Morenia distribution. You must install and set up Python 3.9 for ROS Noetic Ninjemys support.

ROS Actions: Create ROS Action Server in MATLAB

You can now use the `rosactionserver` object to create a ROS action server. Create a function handle for the main action server callback by using the `rosActionServerExecuteGoalFcn` function, which has the necessary callback framework of short functions. You can then create a `rosactionclient` object, connect to the action server, and send goals to the server for it to execute.

ROS Messages: Write camera parameters to ROS or ROS 2 message

You can now use the `rosWriteCameraInfo` function to write camera parameters from a `cameraParameters` or a `stereoParameters` object to a `sensor_msgs/CameraInfo` message structure. For an example, see “Work with Specialized ROS Messages”.

Read Scan Block: Read ROS or ROS 2 laser scan messages in Simulink

You can now use the Read Scan block to read laser scan data from a ROS or ROS 2 laser scan message in Simulink. For an example, see “Read A ROS Scan Message In Simulink®”.

Write Image Block: Write ROS or ROS 2 image messages in Simulink

You can now use the Write Image block to write image pixel data to a ROS or ROS 2 image message structure from Simulink. For an example, see “Write A ROS Image Message In Simulink®”.

Write Point Cloud Block: Write ROS or ROS 2 point cloud messages in Simulink

You can now use the Write Point Cloud block to write point cloud data to a ROS or ROS 2 point cloud message structure from Simulink. For an example, see “Write A ROS Point Cloud Message In Simulink®”.

Functionality being removed or changed

Bus objects for ROS messages in Simulink will now be stored in robotlib data dictionary
Behavior change

Simulink uses bus objects to represent ROS messages. Previously, the bus objects were stored in the base workspace. The Simulink data dictionary `robotlib` will now store these bus objects. You can access this data dictionary by selecting **Model Explorer** from the model toolstrip.

The message object data format will be removed and message structures will be the only supported data format for ROS messages
Warns

The message object data format will be removed in a future release. Instead, use the message structure data format with the following objects and functions. To use ROS message structures, specify the name-value pair argument `Dataformat="struct"`.

- `rospublisher` and `ros.Publisher`
- `rossubscriber` and `ros.Subscriber`
- `rosmessage`
- `rostime`
- `rosduration`
- `rossvcclient` and `ros.ServiceClient`
- `rossvcserver` and `ros.ServiceServer`
- `rosactionclient` and `ros.SimpleActionClient`
- `rostf` and `ros.TransformationTree`

Using structures typically improves performance when you create, update, and use ROS messages. Additionally, using message structures enables code generation. For more information, see Improve Performance of ROS Using Message Structures.

When support for message objects is removed in a future release, all of their corresponding object functions for reading and writing specialized ROS messages will be removed as well. This table lists the corresponding functions for reading and writing specialized ROS messages in the message structure data format.

Message Type	Message Object Function Name	Message Structure Function Name
`Image` `CompressedImage`	`readImage` `writeImage`	`rosReadImage` `rosWriteImage`

`LaserScan`	`readCartesian`
	`readScanAngles`
	`lidarScan`
	`plot`
		`rosReadCartesian`
		`rosReadScanAngles`
		`rosReadLidarScan`
		`rosPlot`

`PointCloud2`	`apply`
	`readXYZ`
	`readRGB`
	`readAllFieldNames`
	`readField`
	`scatter3`
		`rosApplyTransform`
		`rosReadXYZ`
		`rosReadRGB`
		`rosReadAllFieldNames`
		`rosReadField`
		`rosPlot`

Message Type	Message Object Function Name	Message Structure Function Name
`Quaternion`	`readQuaternion`	`rosReadQuaternion`

`OccupancyGrid`	`readBinaryOccupanyGrid`
	`readOccupancyGrid`
	`writeBinaryOccupanyGrid`
	`writeOccupanyGrid`
		`rosReadBinaryOccupancyGrid`
		`rosReadOccupancyGrid`
		`rosWriteBinaryOccupancyGrid`
		`rosWriteOccupancyGrid`

`Octomap`	`readOccupancyMap3D`	`rosReadOccupancyMap3D`

`PointStamped`
`PoseStamped`
`QuaternionStamped`
`Vector3Stamped`
`TransformStamped`
	`apply`	`rosApplyTransform`

All messages	`showdetails`	`rosShowDetails`

ROS 2 Simulation

ROS 2 Foxy Support: Use Foxy Fitzroy distribution for ROS 2

As of R2022a, ROS Toolbox supports the Foxy Fitzroy distribution. A ROS distribution is a stable set of ROS packages that defines the supported platforms, message definitions, and new features for the release. In prior releases, the ROS Toolbox supported the Dashing Diademata distribution. You must install and set up Python 3.9 for ROS 2 Foxy Fitzroy support.

Switchable ROS 2 DDS: Switch between different ROS Middleware (RMW) implementations for using Data Distribution Service (DDS)

ROS 2 uses Data Distribution Services (DDS), such as Real Time Publish Subscribe (RTPS) protocols, as its middleware. As of R2022a, ROS Toolbox enables you to switch between various DDS implementations while working with ROS 2 in MATLAB and Simulink:

- eProsima Fast DDS –– `rmw_fastrtps_cpp` (default)
- eProsima Dynamic Fast DDS –– `rmw_fastrtps_dynamic_cpp`
- Eclipse Cyclone DDS –– `rmw_cyclonedds_cpp`

To switch between different RMW implementations in MATLAB, modify the environment variable `RMW_IMPLEMENTATION` using the `setenv` function.

```
setenv("RMW_IMPLEMNTATION","rmw_cyclonedds_cpp")
```

To switch between different RMW implementations in Simulink, modify the **RMW Implementation** parameter in the **Configure ROS Network Addresses** dialog box. For more information, see “Switching Between ROS Middleware Implementations”.

ROS Messages: Write camera parameters to ROS or ROS 2 message

You can now use the `rosWriteCameraInfo` function to write camera parameters from a `cameraParameters` or a `stereoParameters` object to a `sensor_msgs/CameraInfo` message structure. For an example, see “Work with Specialized ROS Messages”.

Read Scan Block: Read ROS or ROS 2 laser scan messages in Simulink

You can now use the Read Scan block to read laser scan data from a ROS or ROS 2 laser scan message in Simulink. For an example, see “Read A ROS Scan Message In Simulink®”.

Write Image Block: Write ROS or ROS 2 image messages in Simulink

You can now use the Write Image block to write image pixel data to a ROS or ROS 2 image message structure from Simulink. For an example, see “Write A ROS Image Message In Simulink®”.

Write Point Cloud Block: Write ROS or ROS 2 point cloud messages in Simulink

You can now use the Write Point Cloud block to write point cloud data to a ROS or ROS 2 point cloud message structure from Simulink. For an example, see “Write A ROS Point Cloud Message In Simulink®”.

Functionality being removed or changed

Bus objects for ROS 2 messages in Simulink will now be stored in ros2lib data dictionary

Behavior change

Simulink uses bus objects to represent ROS 2 messages. Previously, the bus objects were stored in the base workspace. The Simulink data dictionary `ros2lib` will now store these bus objects. You can access this data dictionary by selecting **Model Explorer** from the model toolstrip.

Examples

New Examples

This release contains the following new examples:

- Control a Simulated UAV Using ROS 2 and PX4 Bridge
- “Generate a ROS Control Plugin from Simulink®”
- “Generate CUDA ROS 2 Node from Simulink”

R2021b

ROS Node Generation and Deployment

C++ Code Generation Support for ROS: Generate ROS nodes for deploying to target hardware using MATLAB Coder

Key ROS features now support code generation using MATLAB Coder, including:

- `rossvcserver`
- `rossvcclient`
- `rosactionclient`
- `rosparam`
- `ParameterTree`

Many other ROS-related functions now also support code generation. For a list of functions that support code generation, and examples of their application, see Code Generation and Deployment. For details about code generation support and limitations for each function, see the **Extended Capabilities** section on the corresponding function reference page.

For code generation, you must use message structures instead of message objects. For more information, see MATLAB Programming for Code Generation.

For an example, see Generate ROS Node for UAV Waypoint Follower.

CUDA Optimized Code Generation Support for ROS: Deploy CUDA-optimized ROS nodes from Simulink to target hardware using GPU Coder

You can now generate and deploy CUDA-optimized code for ROS Nodes from Simulink models. For an example, see Lane and Vehicle Detection in ROS Using YOLO v2 Deep Learning Algorithm.

ROS 2 Node Generation and Deployment

C++ Code Generation Support for ROS 2: Generate ROS 2 nodes for deploying to target hardware using MATLAB Coder

Key ROS 2 features now support code generation using MATLAB Coder including:

- `ros2subscriber`
- `ros2publisher`
- `ros2node`
- `ros2message`

For a list of functions that support code generation, and examples of their application, see Code Generation and Deployment. For details about code generation support and limitations for each function, see the **Extended Capabilities** section on the corresponding function reference page.

ROS Simulation

ROS Log files: Open, parse and write to rosbag files

Use `rosbagwriter` to create a rosbag file in the specified path, and use the `write` function to write records into that bag file.

Use `rosbagreader` to open, parse and store all messages from a rosbag file. You can select a subset of messages to read by using the `select` function, and then read them using the `readMessages` function.

Enhancements for rosdevice and ros2device: Run ROS and ROS 2 nodes on local device and nodes generated from MATLAB

You can now use the `rosdevice` and `ros2device` objects to run ROS and ROS 2 nodes on local device without an SSH connection. Once you have deployed the nodes, you can run them from MATLAB using the `runNode` function.

ROS Support for Custom Actions

You can now use the `rosgenmsg` function to create custom actions. To send goal messages, create a client for the custom action using `rosactionclient`.

Functionality being removed or changed

The message object data format will be removed and message structures will be the only supported data format for ROS messages
Warns

The message object data format will be removed in a future release. Instead, use the message structure data format with the following objects and functions. To use ROS message structures, specify the name-value pair argument `Dataformat="struct"`.

- `rospublisher` and `ros.Publisher`
- `rossubscriber` and `ros.Subscriber`
- `rosmessage`
- `rostime`
- `rosduration`
- `rossvcclient` and `ros.ServiceClient`
- `rossvcserver` and `ros.ServiceServer`
- `rosactionclient` and `ros.SimpleActionClient`
- `rostf` and `ros.TransformationTree`

Using structures typically improves performance when you create, update, and use ROS messages. Additionally, using message structures enables code generation. For more information, see Improve Performance of ROS Using Message Structures.

When support for message objects is removed in a future release, all of their corresponding object functions for reading and writing specialized ROS messages will be removed as well. This table lists

the corresponding functions for reading and writing specialized ROS messages in the message structure data format.

Message Type	Message Object Function Name	Message Structure Function Name
`Image` `CompressedImage`	`readImage` `writeImage`	`rosReadImage` `rosWriteImage`

`LaserScan`	`readCartesian`
	`readScanAngles`
	`lidarScan`
	`plot`
		`rosReadCartesian`
		`rosReadScanAngles`
		`rosReadLidarScan`
		`rosPlot`

`PointCloud2`	`apply`
	`readXYZ`
	`readRGB`
	`readAllFieldNames`
	`readField`
	`scatter3`
		`rosApplyTransform`
		`rosReadXYZ`
		`rosReadRGB`
		`rosReadAllFieldNames`
		`rosReadField`
		`rosPlot`

`Quaternion`	`readQuaternion`	`rosReadQuaternion`

`OccupancyGrid`	`readBinaryOccupanyGrid`
	`readOccupancyGrid`
	`writeBinaryOccupanyGrid`
	`writeOccupanyGrid`
		`rosReadBinaryOccupancyGrid`
		`rosReadOccupancyGrid`
		`rosWriteBinaryOccupancyGrid`
		`rosWriteOccupancyGrid`

`Octomap`	`readOccupancyMap3D`	`rosReadOccupancyMap3D`

`PointStamped`
`PoseStamped`
`QuaternionStamped`
`Vector3Stamped`
`TransformStamped`
	`apply`	`rosApplyTransform`

All messages	`showdetails`	`rosShowDetails`

ROS nodes generated using C++ Code Generation will have the same name as the Simulink model or the entry-point MATLAB function
Behavior change

ROS nodes generated using C++ code generation from MATLAB or Simulink will have the same name as the entry-point function or the model respectively. This is a change from the previous behavior where the name of the nodes used to have a *_node* suffix. Additionally, the header files and source files are now present in the `include` and `src` folders of the generated ROS package.

ROS 2 Simulation

Enhancements for rosdevice and ros2device: Run ROS and ROS 2 nodes on local device and nodes generated from MATLAB

You can now use the `rosdevice` and `ros2device` objects to run ROS and ROS 2 nodes on local device without an SSH connection. Once you have deployed the nodes, you can run them from MATLAB using the `runNode` function.

ROS 2 Log Files: Read ROS 2 bag files in Simulink using Read Data Block

Use Read Data block to read records from ROS 2 Bag files in Simulink.

ROS 2 Services: Create ROS 2 service servers and clients

Use the `ros2svcserver` object to create a ROS 2 service server. Create a `ros2svcclient` object to create a service client object to connect to, send requests, and receive responses from the service server. For more information, see Call and Provide ROS 2 Services.

ROS 2 Call Service Block: Call service in ROS 2 network

You can now use the Call Service block to create a service client in Simulink and call a service in the ROS 2 network. You can specify a custom service and set the Quality of Service (QoS) properties for the block.

ROS 2 Read Image Block: Read ROS 2 image messages in Simulink

You can now use the Read Image block to extract image data from an `Image` or `CompressedImage` ROS 2 message in Simulink. For an example, see Read ROS 2 Image Messages in Simulink® and Perform Registration using Feature Matching .

ROS 2 Read Point Cloud Block: Read ROS 2 point cloud messages in Simulink

You can now use the Read Point Cloud block to extract point cloud data from a `PointCloud2` ROS 2 message in Simulink. For an example, see Read ROS 2 Point Cloud Messages In Simulink® and Perform Stitching using Registration.

ROS 2 Support for Custom Services

You can now use the `ros2genmsg` function to create custom services. To call the custom service in MATLAB, create a service server using `ros2svcserver` and a service client using `ros2svcclient`. To call the custom service in Simulink, use the Call Service block.

Examples

New Examples

This release contains the following new examples:

- Generate ROS Node for UAV Waypoint Follower
- Lane and Vehicle Detection in ROS Using YOLO v2 Deep Learning Algorithm
- Read ROS 2 Image Messages in Simulink® and Perform Registration using Feature Matching
- Read ROS 2 Point Cloud Messages In Simulink® and Perform Stitching using Registration

Updated Examples

This release contains the following updated examples:

- “Sign Following Robot with ROS in MATLAB”

R2021a

Version: 1.3

New Features

Bug Fixes

ROS Node Generation and Deployment

C++ Code Generation Support for ROS: Generate ROS nodes for deploying to target hardware using MATLAB Coder

Key ROS features now support code generation using MATLAB Coder, including:

- `rossubscriber`
- `rospublisher`
- `rosmessage`
- `rosrate`
- `rostime`
- `rosduration`

Many other ROS-related functions now also support code generation. For details about code generation support and limitations, check the **Extended Capabilities** section for each reference page.

For code generation, you must use message structures instead of message objects. For more information, see the ROS Message Structures on page 3-3 release note.

ROS Simulation

ROS Message Structures: Improve performance of ROS messages using structures

You can now create messages as structures with fields matching the message object properties. Using structures typically improves performance when you create, update, and use ROS messages, but message fields are no longer validated when set.

To use ROS messages as structures, set the `'DataFormat'` name-value pair argument when creating your publishers, subscribers, or other ROS objects. Any messages generated from these objects use structures.

```
pub = rospublisher("/scan","sensor_msgs/LaserScan","DataFormat","struct")
msg = rosmessage(pub)
```

```
msg =

struct with fields:

MessageType: 'sensor_msgs/LaserScan'
Header: [1×1 struct]
AngleMin: 0
AngleMax: 0
AngleIncrement: 0
TimeIncrement: 0
ScanTime: 0
RangeMin: 0
RangeMax: 0
Ranges: [0×1 single]
Intensities: [0×1 single]
```

You can also create messages as structures directly, but other ROS objects use message objects by default. Make sure to specify the data format as `"struct"` for the publisher, subscriber, or other ROS objects as well.

```
msg = rosmessage("/scan","sensor_msgs/LaserScan","DataFormat","struct")
pub = rospublisher("/scan","sensor_msgs/LaserScan","DataFormat","struct")
```

These functions support structure messages using the `'DataFormat'` name-value argument:

- `rosmessage`
- `rospublisher`
- `rossubscriber`
- `rostime`
- `rostf`
- `rosduration`
- `rossvcclient`
- `rossvcserver`
- `rosactionclient`

For more information, see Improve Performance of ROS Using Message Structures.

New ROS Functions for Messages: Use new functions for processing messages as structures

Functions that operate on specialized ROS messages are now available to support the use of message structures as inputs. These new functions replace the existing object functions of message objects, and support ROS and ROS 2 message structures as inputs instead of message objects.

The object functions will be removed in a future release.

Message Types	Object Function Name	New Function Name
`Image` `CompressedImage`	`readImage` `writeImage`	`rosReadImage` `rosWriteImage`

`LaserScan`	`readCartesian`
	`readScanAngles`
	`lidarScan`
	`plot`
		`rosReadCartesian`
		`rosReadScanAngles`
		`rosReadLidarScan`
		`rosPlot`

`PointCloud2`	`apply`
	`readXYZ`
	`readRGB`
	`readAllFieldNames`
	`readField`
	`scatter3`
		`rosApplyTransform`
		`rosReadXYZ`
		`rosReadRGB`
		`rosReadAllFieldNames`
		`rosReadField`
		`rosPlot`

`Quaternion`	`readQuaternion`	`rosReadQuaternion`

`OccupancyGrid`	`readBinaryOccupanyGrid`
	`readOccupancyGrid`
	`writeBinaryOccupanyGrid`
	`writeOccupanyGrid`
		`rosReadBinaryOccupancyGrid`
		`rosReadOccupancyGrid`
		`rosWriteBinaryOccupancyGrid`
		`rosWriteOccupancyGrid`

`Octomap`	`readOccupancyMap3D`	`rosReadOccupancyMap3D`

`PointStamped`
`PoseStamped`
`QuaternionStamped`
`Vector3Stamped`
`TransformStamped`
	`apply`	`rosApplyTransform`

Message Types	Object Function Name	New Function Name
All messages	`showdetails`	`rosShowDetails`

ROS Header Assignment Block: Update fields of ROS Header messages

The Header Assignment block enables you to update the fields of a ROS Header message in Simulink. You can specify the frame ID manually, set the time stamp based on the current ROS time, and use custom Header field names for your specific ROS application.

ROS Toolstrip Update: Select ROS network types and deploy local or remote nodes with code generation

The **ROS** tab on the Simulink Toolstrip now enables you to switch between ROS and ROS 2 board implementations and specify a ROS device for deploying ROS nodes from Simulink.

To add the **ROS** tab to the Simulink Toolstrip, open the **APPS** tab and click the down arrow. Under **Control Systems**, select **Robot Operating System (ROS)** to add the tab to your model.

The **ROS Network** list allows you to choose between ROS, Raspberry Pi™ ROS, or ROS 2. This sets the **Hardware board** and other configuration parameters appropriately.

The **Deploy to** list enables you to select `Localhost` or `Remote Device`. You can also set your remote device parameters by selecting **Manage Remote Device**.

ROS 2 Simulation

ROS 2 Log File: Open and parse ROS 2 bag file

Use the `ros2bag` object to open, parse, and store all messages from the ROS 2 bag file. You can retrieve the messages from the `ros2bag` object using the `readMessages` function.

R2020b

Version: 1.2

New Features

Bug Fixes

Version History

ROS Simulation

Velodyne ROS Message Reader: Read Velodyne lidar messages from ROS in MATLAB

The `velodyneROSMessageReader` object reads ROS messages from a rosbag or ROS network and loads the point cloud data into MATLAB as `pointCloud` objects. The `velodyneROSMessageReader` object supports the `'VLP16'`, `'VLP32C'`, `'HDL32E'`, and `'HDL64E'` device models.

ROS Toolbox Interface for ROS Custom Messages add-on added to ROS Toolbox

The ROS Toolbox Interface for ROS Custom Messages add-on is now part of the ROS Toolbox installation. You no longer need to use the **Add-On Explorer** to install the `rosgenmsg` and `ros2genmsg` functions.

Also, you no longer need to restart MATLAB after adding custom message definitions.

To use custom messages, follow these steps:

- Use the `rosgenmsg` or `ros2genmsg` function with the path to the parent folder of your custom message packages as the input argument.
- Add the custom message folder to the MATLAB:

  ```
  addpath('<folderpath>/matlab_msg_gen_ros1/<arch>/install/m')

  savepath
  ```
- Refresh all message class definitions, which requires clearing the workspace:

  ```
  clear classes

  rehash toolboxcache
  ```

No MATLAB restart is required to see the updated ROS messages.

ROS Toolbox requires Python 2.7 for starting a ROS Master from MATLAB

When connecting to a ROS network, you must have Python version 2.7. Download Python from the homepage and set the version using the `pyenv` function.

Version History

To connect to a ROS network using the `rosinit` function, set your Python version using the `pyenv` function. The selected version of Python will be used for all subsequent MATLAB operations, and is persistent across sessions.

ROS Melodic: Update ROS 1 version support to Melodic Morenia distribution

Starting this release, the ROS Toolbox supports the ROS Melodic Morenia distribution. A ROS distribution is a stable set of ROS packages that define the supported platforms, message definitions, and new features for the release. In previous releases, the ROS Toolbox supported the Indigo Igloo distribution.

Sequence Numbers for Header Messages

Version History

Previously, when manually setting the `Seq` property of a ROS `Header` message (`std_msgs/Header`), a subscriber that receives this message would keep the given sequence number (from the published message) and not increment the value.

In R2020b, the subscriber always increments the sequence value whenever receiving a new message by adding one to the `Seq` property.

R2020a

Version: 1.1

Bug Fixes

ROS Simulation

ROS 2 Dashing: Update ROS 2 version support to Dashing Diademata distribution

Starting this release, the ROS Toolbox supports the Dashing Diademata distribution. A ROS distribution is a stable set of ROS packages that define the supported platforms, message definitions, and new features for the release. In previous releases, the ROS Toolbox supported the Bouncy Bolson distribution.

R2019b

Version: 1.0

New Features

ROS Node Generation and Deployment

Deployment of ROS Nodes: Deploy ROS and ROS 2 nodes to target hardware using Simulink Coder

For examples that generate code for standalone ROS nodes, see:

- Generate a Standalone ROS Node from Simulink®
- Generate a Standalone ROS 2 Node from Simulink®

ROS Simulation

Network Connection and Exploration: Communicate with ROS and ROS 2 nodes in a network using MATLAB and Simulink

Connect to ROS and ROS 2 to prototype robotics applications and access robotics hardware or simulators over a ROS network. You can create your own ROS network using MATLAB or connect to an existing ROS network. To set up a ROS network, start by calling `rosinit`. For ROS 2 networks, see `ros2node`.

For more information, see Network Connection and Exploration.

Multiplatform Support: Access ROS functionality from Windows, Mac, and Linux

The ROS Toolbox enables you to connect to and run ROS and ROS 2 networks on Windows®, Mac, and Linux platforms.

Publishers and Subscribers: Send and receive ROS and ROS 2 messages with MATLAB and Simulink via a ROS network

ROS shares information using messages. Messages are a simple data structure for sharing data. To receive, or subscribe to, a message, use `rossubscriber` or `ros2subscriber`. To send, or publish, a message, use `rospublisher` or `ros2publisher`. For an example of sending and receiving messages, see Exchange Data with ROS Publishers and Subscribers or Exchange Data with ROS 2 Publishers and Subscribers.

For more information, see Publishers and Subscribers.

Custom Messages: Generate custom messages to use on both ROS and ROS 2 networks based on specified packages

You can create your own ROS custom messages and use them in MATLAB and Simulink with ROS networks to transmit information. For ROS custom messages, use `rosAddons` to install the necessary addon, and then use the `rosgenmsg` function. To learn the requirements for generating custom messages, see ROS Custom Message Support. For ROS 2, use `ros2genmsg` with your custom message packages and see the ROS 2 Custom Message Support example.

Log File Playback: Import ROS log files (rosbags) to filter, visualize, and analyze logged data

ROS topics are stored in log files called rosbags. You can access and filter information from rosbags in MATLAB. For an example of working with rosbags, see Work with `rosbag` Logfiles.

You can access transformations between coordinate systems as ROS topics and use them to transform data in MATLAB. For more information, see Access the tf Transformation Tree in ROS.

For more information, see ROS Log Files and Transformations

ROS Toolbox Support Package for TurtleBot-Based Robots: Connect to TurtleBot hardware

For more information, see ROS Toolbox Support Package for TurtleBot -Based Robots.