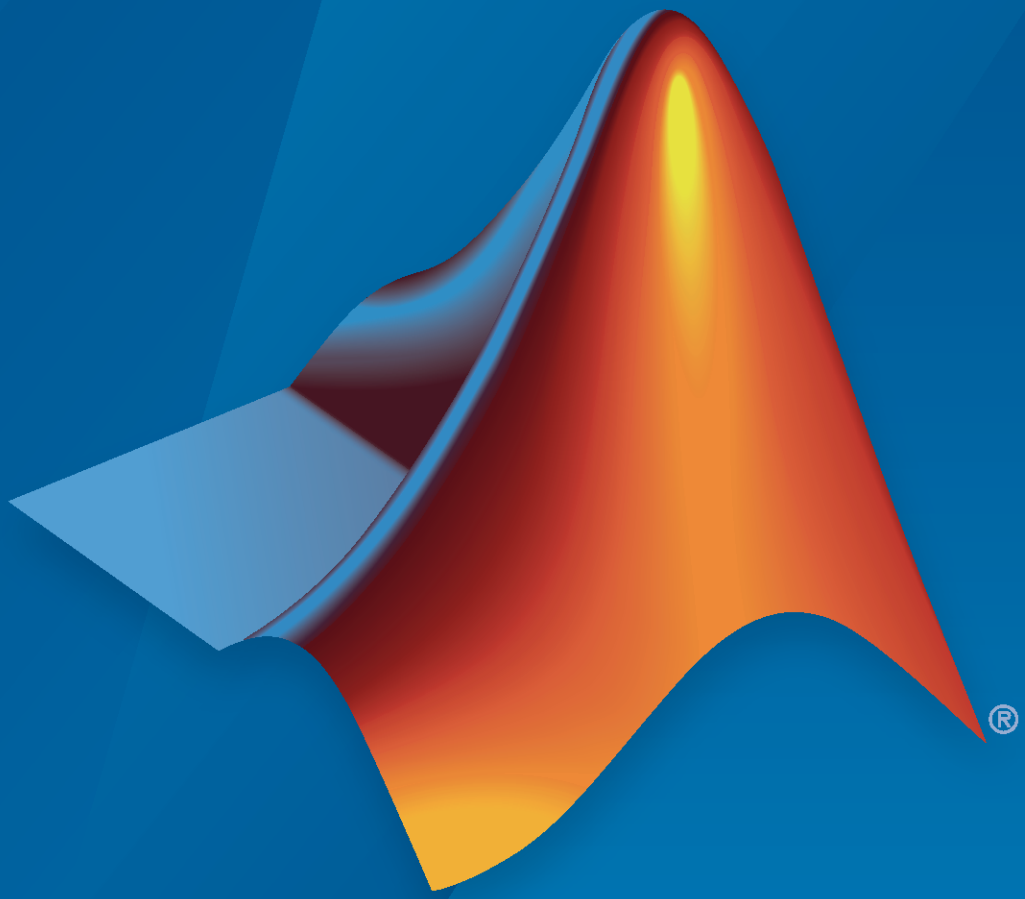
ROS Toolbox Release Notes



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

© COPYRIGHT 2019–2023 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.

Contents

R2023a

ROS Node Generation and Deployment **1-2**

C++ Code Generation Support Enhancement for ROS: Use new features for deploying ROS nodes to target hardware using MATLAB Coder **1-2**

ROS 2 Node Generation and Deployment **1-3**

C++ Code Generation Support Enhancement for ROS 2: Use new features for deploying ROS nodes to target hardware using MATLAB Coder **1-3**

ROS Bag Files and Specialized Messages **1-4**

ROS Bag Viewer App: Play ROS bag files in MATLAB **1-4**
Enhancements for ROS and ROS 2 Specialized Messages: Read point clouds in double precision, read images with Bayer encodings, and write intensity data to point cloud messages **1-4**

ROS Network Access **1-5**

64-bit Integer Support for Simulink: Access int64 or uint64 data from ROS messages in Simulink **1-5**
ROS Service Client: Create ROS service clients with persistent connections to service servers **1-5**
ROS Write Point Cloud Data: Write point cloud messages in MATLAB **1-5**
Functionality being removed or changed **1-5**

ROS 2 Network Access **1-8**

ROS 2 Actions: Create ROS 2 action clients in MATLAB **1-8**
Custom Messages: Share and register generated ROS 2 custom messages with MATLAB **1-8**
ROS 2 Write Point Cloud Data: Write point cloud messages in MATLAB . . . **1-8**
ROS 2 Transformation: Apply ROS 2 transformations to logged data **1-8**
Record from Simulink to ROS 2 Bag Log File: Record signals from ROS 2 enabled Simulink models to ROS 2 bag log files **1-8**
ROS 2 Header Assignment Block: Update fields of ROS 2 Header messages **1-9**
Support Switching Between ROS Middleware Configurations **1-9**

Examples **1-10**

New Examples **1-10**
Updated Examples **1-10**

R2022b

ROS Node Generation and Deployment **2-2**

CUDA-Optimized Code Generation Support for ROS: Deploy CUDA-optimized ROS nodes from MATLAB to local host or target hardware using GPU Coder **2-2**
ros_control Controller Generation Enhancement: Tune parameters in the Simulink model with dynamic reconfiguration support **2-2**

ROS 2 Node Generation and Deployment **2-3**

CUDA-Optimized Code Generation Support for ROS 2: Deploy CUDA-optimized ROS 2 nodes from MATLAB to local host or target hardware using GPU Coder **2-3**

ROS Network Access **2-4**

Record from Simulink to Rosbag: Record signals from ROS-enabled Simulink models to rosbag files **2-4**
Custom Messages: Share and register generated ROS custom messages with MATLAB **2-4**
Functionality being removed or changed **2-4**

ROS 2 Network Access **2-6**

ROS 2 Log file: Open, parse, and write to ROS 2 bag file **2-6**
ROS 2 Time: Get system or ROS 2 time information, create ROS 2 duration messages, and manage loop execution rate **2-6**
ROS 2 Parameters: Add parameters to and obtain parameters from ROS 2 nodes **2-6**
Custom Messages: Share and register generated ROS 2 custom messages with MATLAB **2-6**
Support Switching Between ROS Middleware Configurations **2-6**
Functionality being removed or changed **2-7**

ROS Toolbox System Requirements **2-8**

Support for Visual Studio Compiler **2-8**
Support for CMake Platform **2-8**

Examples **2-9**

New Examples **2-9**

R2022a

ROS Node Generation and Deployment **3-2**

C++ Code Generation Support Enhancement for ROS: Generate ROS nodes for deploying to target hardware using MATLAB Coder **3-2**

ros_control Controller Generation Support for ROS: Generate and deploy ros_control controller plugins from Simulink **3-2**

ROS 2 Node Generation and Deployment **3-3**

C++ Code Generation Support Enhancement for ROS 2: Generate ROS 2 nodes for deploying to target hardware using MATLAB Coder **3-3**
CUDA Optimized Code Generation Support for ROS 2: Deploy CUDA-optimized ROS 2 nodes from Simulink to target hardware using GPU Coder **3-3**

ROS Network Access **3-4**

ROS Noetic Support: Use Noetic Ninjemys distribution for ROS **3-4**
ROS Actions: Create ROS Action Server in MATLAB **3-4**
ROS Messages: Write camera parameters to ROS or ROS 2 message **3-4**
Read Scan Block: Read ROS or ROS 2 laser scan messages in Simulink . . **3-4**
Write Image Block: Write ROS or ROS 2 image messages in Simulink **3-4**
Write Point Cloud Block: Write ROS or ROS 2 point cloud messages in Simulink **3-4**
Functionality being removed or changed **3-4**

ROS 2 Network Access **3-7**

ROS 2 Foxy Support: Use Foxy Fitzroy distribution for ROS 2 . . . . . **3-7**
Switchable ROS 2 DDS: Switch between different ROS Middleware (RMW) implementations for using Data Distribution Service (DDS) **3-7**
ROS Messages: Write camera parameters to ROS or ROS 2 message **3-7**
Read Scan Block: Read ROS or ROS 2 laser scan messages in Simulink . . **3-7**
Write Image Block: Write ROS or ROS 2 image messages in Simulink **3-7**
Write Point Cloud Block: Write ROS or ROS 2 point cloud messages in Simulink **3-8**
Functionality being removed or changed **3-8**

Examples **3-9**

New Examples **3-9**

R2021b

ROS Node Generation and Deployment **4-2**

C++ Code Generation Support for ROS: Generate ROS nodes for deploying to target hardware using MATLAB Coder **4-2**
CUDA Optimized Code Generation Support for ROS: Deploy CUDA-optimized ROS nodes from Simulink to target hardware using GPU Coder **4-2**

ROS 2 Node Generation and Deployment **4-3**

C++ Code Generation Support for ROS 2: Generate ROS 2 nodes for deploying to target hardware using MATLAB Coder **4-3**

ROS Network Access **4-4**

ROS Log files: Open, parse and write to rosbag files **4-4**
Enhancements for rosdevice and ros2device: Run ROS and ROS 2 nodes on local device and nodes generated from MATLAB **4-4**
ROS Support for Custom Actions **4-4**
Functionality being removed or changed **4-4**

ROS 2 Network Access **4-7**

Enhancements for rosdevice and ros2device: Run ROS and ROS 2 nodes on local device and nodes generated from MATLAB **4-7**
ROS 2 Log Files: Read ROS 2 bag files in Simulink using Read Data Block **4-7**
ROS 2 Services: Create ROS 2 service servers and clients **4-7**
ROS 2 Call Service Block: Call service in ROS 2 network **4-7**
ROS 2 Read Image Block: Read ROS 2 image messages in Simulink **4-7**
ROS 2 Read Point Cloud Block: Read ROS 2 point cloud messages in Simulink **4-7**
ROS 2 Support for Custom Services **4-7**

Examples **4-8**

New Examples **4-8**
Updated Examples **4-8**

R2021a

ROS Node Generation and Deployment **5-2**

C++ Code Generation Support for ROS: Generate ROS nodes for deploying to target hardware using MATLAB Coder **5-2**

ROS Network Access **5-3**

ROS Message Structures: Improve performance of ROS messages using structures **5-3**
New ROS Functions for Messages: Use new functions for processing messages as structures **5-4**
ROS Header Assignment Block: Update fields of ROS Header messages **5-5**
ROS Toolstrip Update: Select ROS network types and deploy local or remote nodes with code generation **5-5**

ROS 2 Network Access **5-6**

ROS 2 Log File: Open and parse ROS 2 bag file **5-6**

R2020b

ROS Network Access **6-2**

Velodyne ROS Message Reader: Read Velodyne lidar messages from ROS in MATLAB **6-2**
ROS Toolbox Interface for ROS Custom Messages add-on added to ROS Toolbox **6-2**
ROS Toolbox requires Python 2.7 for starting a ROS Master from MATLAB **6-2**
ROS Melodic: Update ROS 1 version support to Melodic Morenia distribution **6-3**
Sequence Numbers for Header Messages **6-3**

R2020a

ROS Network Access **7-2**

ROS 2 Dashing: Update ROS 2 version support to Dashing Diademata distribution **7-2**

R2019b

ROS Node Generation and Deployment **8-2**

Deployment of ROS Nodes: Deploy ROS and ROS 2 nodes to target hardware using Simulink Coder **8-2**

ROS Network Access **8-3**

Network Connection and Exploration: Communicate with ROS and ROS 2 nodes in a network using MATLAB and Simulink **8-3**
Multiplatform Support: Access ROS functionality from Windows, Mac, and Linux **8-3**
Publishers and Subscribers: Send and receive ROS and ROS 2 messages with MATLAB and Simulink via a ROS network **8-3**
Custom Messages: Generate custom messages to use on both ROS and ROS 2 networks based on specified packages **8-3**
Log File Playback: Import ROS log files (rosbags) to filter, visualize, and analyze logged data **8-3**
ROS Toolbox Support Package for TurtleBot-Based Robots: Connect to TurtleBot hardware **8-4**

R2023a

Version: 2.0

New Features

Bug Fixes

Compatibility Considerations

ROS Node Generation and Deployment

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

These ROS features now support code generation using MATLAB® Coder™ software.

- `rosactionserver`
- `rosActionServerExecuteGoalFcn`
- `getFeedbackMessage`
- `isPreemeptRequested`
- `sendFeedback`

ROS 2 Node Generation and Deployment

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

These ROS 2 features now support code generation using MATLAB Coder software.

- `ros2tf`
- `getTransform`
- `canTransform`
- `transform`
- `sendTransform`

ROS Bag Files and Specialized Messages

ROS Bag Viewer App: Play ROS bag files in MATLAB

Use the **ROS Bag Viewer** app to play messages from ROS bag files. You can load the bag file, create viewers for different message types, and specify the topics for those viewers to display. You can then play the bag file and visualize the messages in those viewers. For more information about loading bag files and visualizing messages, see “Get Started with ROS Bag Viewer App”.

You can also specify which topic in the bag file acts as a reference for the playback frequency and also control the speed and direction of playback.

Enhancements for ROS and ROS 2 Specialized Messages: Read point clouds in double precision, read images with Bayer encodings, and write intensity data to point cloud messages

This table lists the enhancements for ROS and ROS 2 specialized message operations in MATLAB and Simulink®.

Features	Enhancements
• `rosReadField` • `rosReadXYZ` • `rosReadRGB`	Reading `sensor_msgs/PointCloud2` messages in double precision now supports code-generation. To read a point cloud message in double precision, specify the `DataType` name-value argument as `double`.
`rosReadImage`	You can now read `sensor_msgs/CompressedImage` messages with `JPEG`, `JPG` and `PNG` message formats.
• Read Image • Read Image (ROS 2)	You can now read image messages with Bayer. Specify the encoding type in the `Image Encoding` parameter.
Write Point Cloud	You can now write intensity data to a point cloud message. To write intensity data to point cloud messages, enable the `Write to Intensity field` parameter.

ROS Network Access

64-bit Integer Support for Simulink: Access int64 or uint64 data from ROS messages in Simulink

You can now access data from ROS message fields of 64-bit `int64` or `uint64` data type directly in Simulink. In prior releases, you had to convert any 64-bit data to `double` before accessing it in Simulink.

ROS Service Client: Create ROS service clients with persistent connections to service servers

Use the `rossvcclient` object to create a ROS service client with persistent connection to a service server by specifying the `IsPersistent` name-value argument as `true`. A persistent service client stays connected to a service until the service server exists on the network.

ROS Write Point Cloud Data: Write point cloud messages in MATLAB

You can now write point cloud data to a ROS point cloud message structure from MATLAB by using these functions.

- `rosWriteXYZ` – Write XYZ 2-D or 1-D matrix data.
- `rosWriteRGB` – Write RGB data.
- `rosWriteIntensity` – Write Intensity data.

Functionality being removed or changed

Specify service type when you create ROS service client
Behavior change in future release

In a future release, the syntax to create a ROS service client without specifying the service type using the `rossvcclient` object, will be removed. You can specify the service type as an input argument when you create the `rossvcclient` object. If the specified service type is not available on the ROS network, `rossvcclient` object will throw an error.

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`

Message Type	Message Object Function Name	Message Structure Function Name

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

All messages	`showdetails`	`rosShowDetails`

ROS 2 Network Access

ROS 2 Actions: Create ROS 2 action clients in MATLAB

Create a ROS 2 action client by using `ros2actionclient` object. Use `ros2ActionSendGoalOptions` function to execute custom callback functions when it receives goal responses, feedback, and results. You can then send goal request to the action server using the `sendGoal` function with the callback options for execution of the request. Alternatively, to receive the goal execution result within a specified timeout period you can use the `getResult` function.

Custom Messages: Share and register generated ROS 2 custom messages with MATLAB

Use the `ros2genmsg` function to create a custom action client. To send goal messages, create a client for the custom action using `ros2actioncient`.

Use the `ros2RegisterMessages` function to register the generated ROS 2 custom messages with MATLAB.

ROS 2 Write Point Cloud Data: Write point cloud messages in MATLAB

You can now write point cloud data to a ROS 2 point cloud message structure from MATLAB by using these functions.

- `rosWriteXYZ` – Write XYZ 2-D or 1-D matrix data.
- `rosWriteRGB` – Write RGB data.
- `rosWriteIntensity` – Write Intensity data.

ROS 2 Transformation: Apply ROS 2 transformations to logged data

Use the `ros2tf` object to receive, send, verify, and apply ROS 2 transformations to logged data.

- `getTransform` to return the latest known transformation between two coordinate frames.
- `canTransform` to verify the availability of the transformation.
- `transform` to transform Stamped Message into Target co-ordinate frame.
- `sendTransform` to send Dynamic Transformation to a ROS 2 network.

Record from Simulink to ROS 2 Bag Log File: Record signals from ROS 2 enabled Simulink models to ROS 2 bag log files

You can now use the **ROS Logger** app to record ROS 2 messages from Simulink models to ROS 2 bag log files during simulation.

ROS 2 Header Assignment Block: Update fields of ROS 2 Header messages

Update the fields of a ROS Header message in Simulink by using the Header Assignment block. You can specify the frame ID, set the time stamp based on the current ROS time, and use custom Header field names for your ROS application.

Support Switching Between ROS Middleware Configurations

You can now switch to any custom RMW implementations based on static introspection or dynamic introspection in the **ROS Toolbox Preferences** panel.

Examples

New Examples

This release contains the following new examples:

- “Generate Standalone ROS 2 Node”
- “Deploy ROS Node for Sign Following Robot with Time Synchronization Using Simulink”
- “Emergency Braking of Ego Vehicle in CARLA Simulator Using Simulink and CARLA ROS Bridge”
- “Log ROS 2 Messages from Simulink to a ROS 2 Bag File”
- “Use ROS 2 Logger App to Save ROS 2 Messages from Simulink”

Updated Examples

This release contains the following updated examples:

- “Sign-following Robot with ROS in MATLAB”
- “Automated Parking Valet with ROS in Simulink”
- “Automated Parking Valet with ROS 2 in Simulink”

R2022b

Version: 1.6

New Features

Bug Fixes

Compatibility Considerations

ROS Node Generation and Deployment

CUDA-Optimized Code Generation Support for ROS: Deploy CUDA-optimized ROS nodes from MATLAB to local host or target hardware using GPU Coder

You can now generate and deploy CUDA-optimized code for ROS nodes from all MATLAB functions that support GPU Coder™, to local host or target hardware.

ros_control Controller Generation Enhancement: Tune parameters in the Simulink model with dynamic reconfiguration support

You can now specify block parameters in Simulink models as tunable parameters in the generated `ros_control` controller plugins. These blocks support specifying their parameters as tunable:

- Gain (Simulink)
- Slider Gain (Simulink)
- Constant (Simulink)
- Compare To Constant (Simulink)
- Saturation (Simulink)
- String Constant (Simulink)

In the ROS Control interfaces configuration dialog, in the **Parameters** tab you can specify which parameters require dynamic reconfiguration support. This enables you to tune them using ROS services after you deploy the controller plugin.

ROS 2 Node Generation and Deployment

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

You can now generate and deploy CUDA-optimized code for ROS 2 nodes from all MATLAB functions that support GPU Coder, to local host or target hardware.

ROS Network Access

Record from Simulink to Rosbag: Record signals from ROS-enabled Simulink models to rosbag files

You can now use the *ROS Logger* app to record ROS messages from Simulink models to rosbag files during simulation. For an example, see Log ROS Messages from Simulink to a Rosbag Logfile.

Custom Messages: Share and register generated ROS custom messages with MATLAB

You can now share the custom messages generated on one machine using the `rosgenmsg` function with another machine running on the same platform and using the same MATLAB version.

Use the `rosRegisterMessages` function to register the generated ROS custom messages with MATLAB.

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 2 Network Access

ROS 2 Log file: Open, parse, and write to ROS 2 bag file

Use the `ros2bagwriter` object to create a ROS 2 bag file at the specified location. Use the `write` object function to write records into that bag file.

ROS 2 Time: Get system or ROS 2 time information, create ROS 2 duration messages, and manage loop execution rate

You can now obtain the current time used by ROS 2 nodes on the network by using the `ros2time` function, and create ROS 2 duration messages by using the `ros2duration` function. You can also control the loop execution rate using the `ros2rate` object.

To obtain the current time in Simulink use the Current Time block.

ROS 2 Parameters: Add parameters to and obtain parameters from ROS 2 nodes

You can now add parameters to a ROS 2 node by specifying the name-value argument `Parameters` while creating a `ros2node` object. Use `getParameter` and `setParameter` functions to obtain the value of, and set the value of a specific parameter associated with a ROS node, respectively.

To obtain the value of a specific parameter from a ROS 2 node in Simulink, use the Get Parameter block.

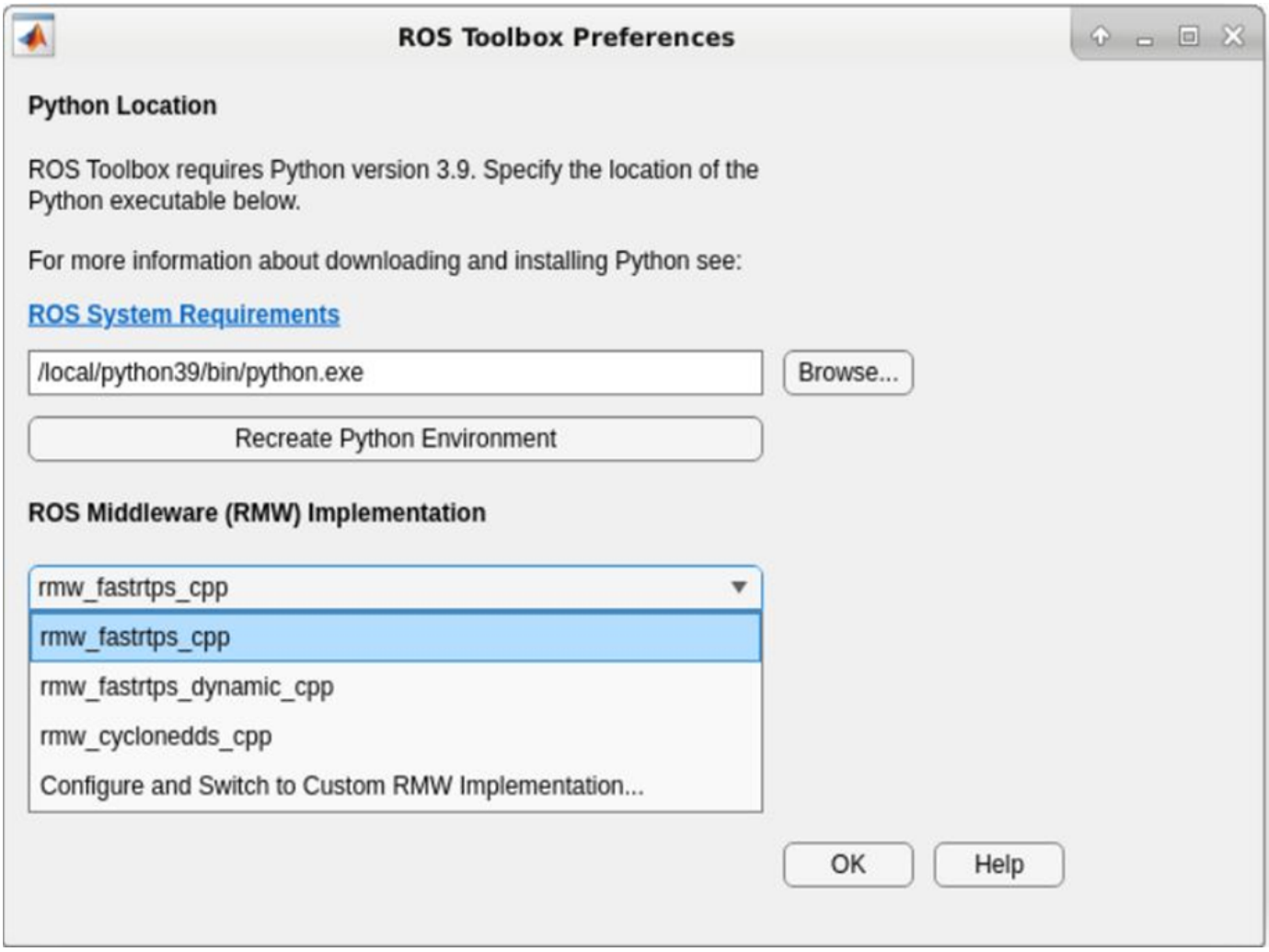
Custom Messages: Share and register generated ROS 2 custom messages with MATLAB

You can now share the custom messages generated on one machine using the `ros2genmsg` function with another machine running on the same platform and using the same MATLAB version.

Use the `ros2RegisterMessages` function to register the generated ROS 2 custom messages with MATLAB.

Support Switching Between ROS Middleware Configurations

The ROS Toolbox Preferences GUI enables you to switch between various ROS middleware (RMW) implementations, while working with ROS 2 in MATLAB and Simulink.



As of R2022b, ROS Toolbox enables support for the RMW Implementations:

- `rmw_fastrtps_cpp` (default)
- `rmw_fastrtps_dynamic_cpp`
- `rmw_cyclonedds_cpp`
- `rmw_connextdds` (Supported on Windows® and Linux® only)
- `rmw_iceoryx_cpp` (Supported on Linux only)

Functionality being removed or changed

ros2bag Object Renamed
Behavior change

The `ros2bag` object has been renamed to `ros2bagreader`. Use `ros2bagreader` to open, parse, and store all messages from a ROS 2 bag file. You can select a subset of messages to read by using the `select` function, and then read them using the `readMessages` function.

ROS Toolbox System Requirements

Support for Visual Studio Compiler

As of R2022b, ROS Toolbox supports Visual Studio 2022.

Support for CMake Platform

As of R2022b, ROS Toolbox ships CMake inherently in MATLAB for custom message generation and Code Generation. Manual installation of CMake is not required.

Examples

New Examples

This release contains the following new examples:

- Pick-and-Place Workflow in Unity3D Using ROS
- Use ROS Logger App to Save ROS Messages from Simulink
- Log ROS Messages from Simulink to a Rosbag Logfile

R2022a

Version: 1.5

New Features

Bug Fixes

Compatibility Considerations

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 Network Access

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` (Computer Vision Toolbox) or a `stereoParameters` (Computer Vision Toolbox) 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 Network Access

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` (Computer Vision Toolbox) or a `stereoParameters` (Computer Vision Toolbox) 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

Version: 1.4

New Features

Bug Fixes

Compatibility Considerations

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 Network Access

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 Network Access

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 5-3 release note.

ROS Network Access

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 Network Access

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

Compatibility Considerations

ROS Network Access

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.

Compatibility Considerations

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

Compatibility Considerations

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 Network Access

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 Network Access

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.