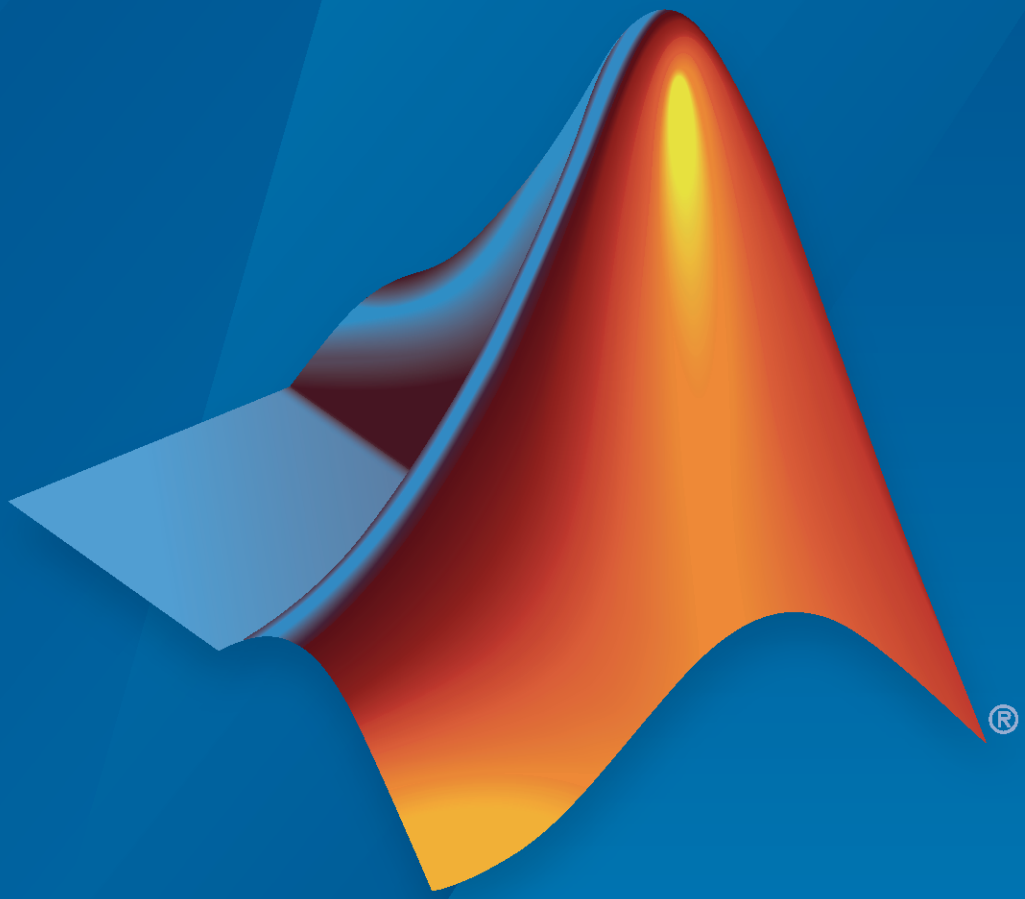
# DDS Blockset

User's Guide



MATLAB® & SIMULINK®

R2023a

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

*DDS Blockset User's Guide*



**Revision History**

| | | |
|---|---|---|
| March 2021 | Online only | New for Version 1.0 (Release 2021a) |
| September 2021 | Online Only | Revised for Version 1.1 (Release 2021b) |
| March 2022 | Online Only | Revised for Version 1.2 (Release 2022a) |
| September 2022 | Online Only | Revised for Version 1.3 (Release 2022b) |
| March 2023 | Online Only | Revised for Version 1.4 (Release 2023a) |

# Contents

## 1 Import DDS



## 2 DDS Definition Management



## 3 Model Architecture

## 4 DDS Interface Configuration



## 5 Deployment



## 6 DDS Examples

# Import DDS

# Import or Create DDS Definitions

To bring DDS concepts such as Domains, Topics, Types, and Quality of Service (QoS) into the Simulink environment so that you can model and configure DDS applications, use the DDS Application Quick Start. The Quick Start enables you to import XML and IDL definitions or create default definitions and configure your application appropriately for its DDS vendor, RTI or eProsima, to provide the framework to model and build your application in Simulink.

To create DDS definitions and configure your application:

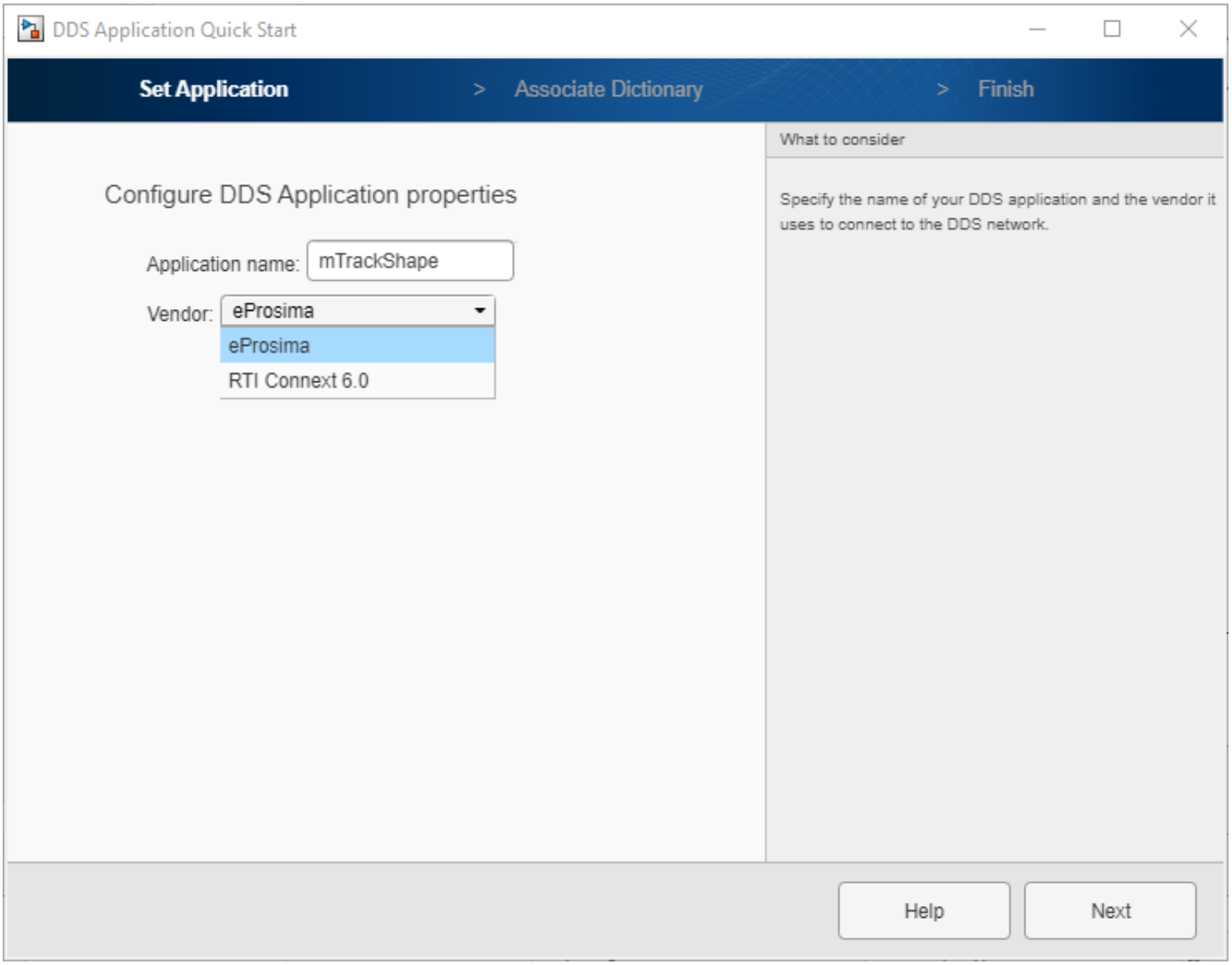
1. “Open the DDS Application Quick Start” on page 1-2.
2. “Specify Application Name and Vendor” on page 1-2.
3. “Specify Source of DDS Definitions” on page 1-3.
4. “Transition from DDS to Simulink Environment” on page 1-4

## Open the DDS Application Quick Start

When you open a model in the DDS Blockset app, you are directed into the DDS Application Quick Start. The Quick Start enables you to name your application, specify your DDS vendor, and import or create DDS definitions required to configure your DDS application. To update these configuration options after completing the Quick Start, you can reopen the DDS Application Quick Start from the **DDS** tab by clicking **Quick Start**.

## Specify Application Name and Vendor

On the first page of DDS Application Quick Start, specify the DDS application name and vendor.



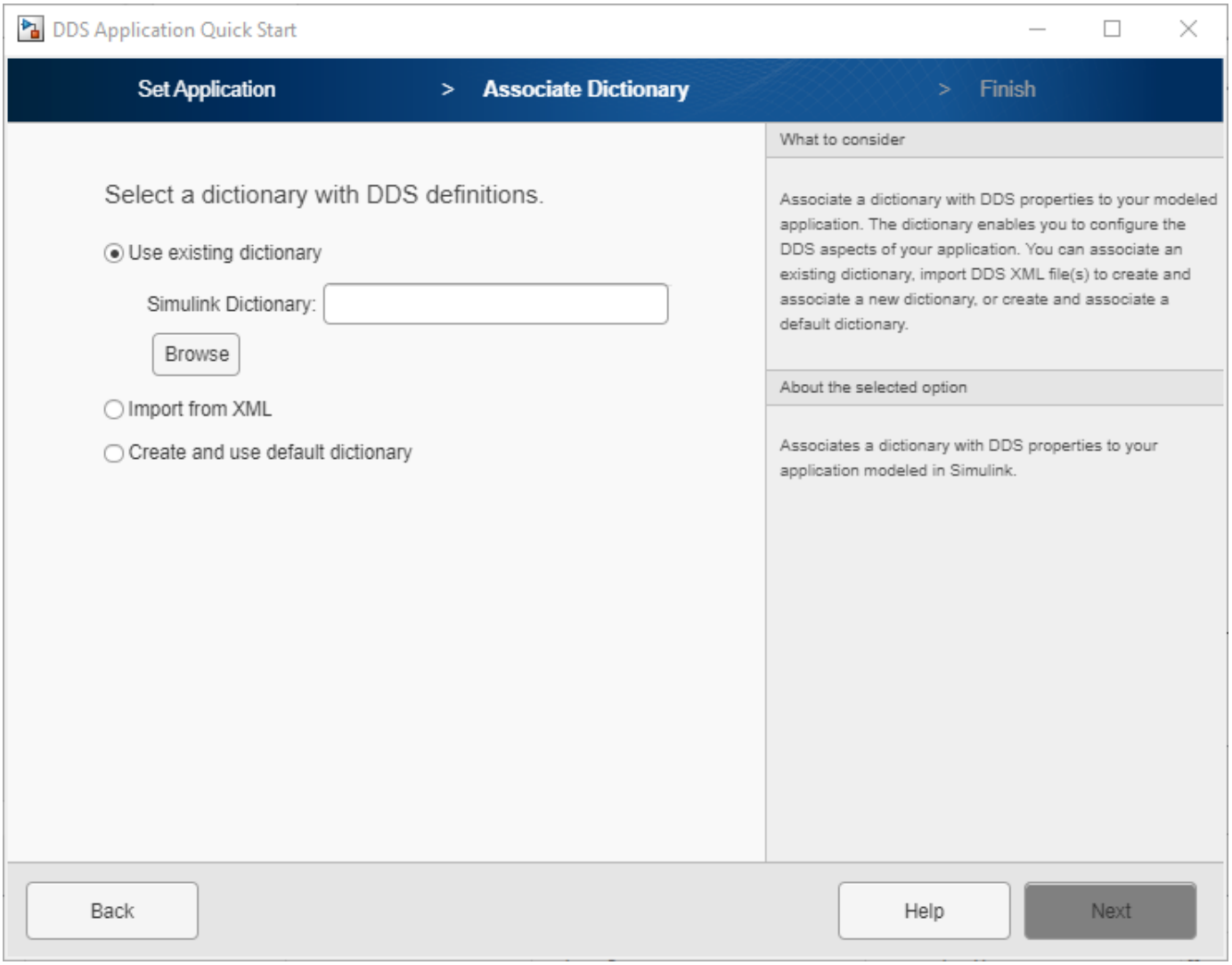

By default, the name of your DDS application is set to the name of its application model. To customize the name, in the **Application name** field, enter a customized name.

To specify your DDS vendor (and set your toolchain), from the **Vendor** drop-down list, select RTI or eProsima. If you do not intend to generate code, vendor selection does not affect simulation, so you can accept the default vendor. For more information about the vendor setup for DDS Blockset, see "DDS Blockset System Requirements".

## Specify Source of DDS Definitions

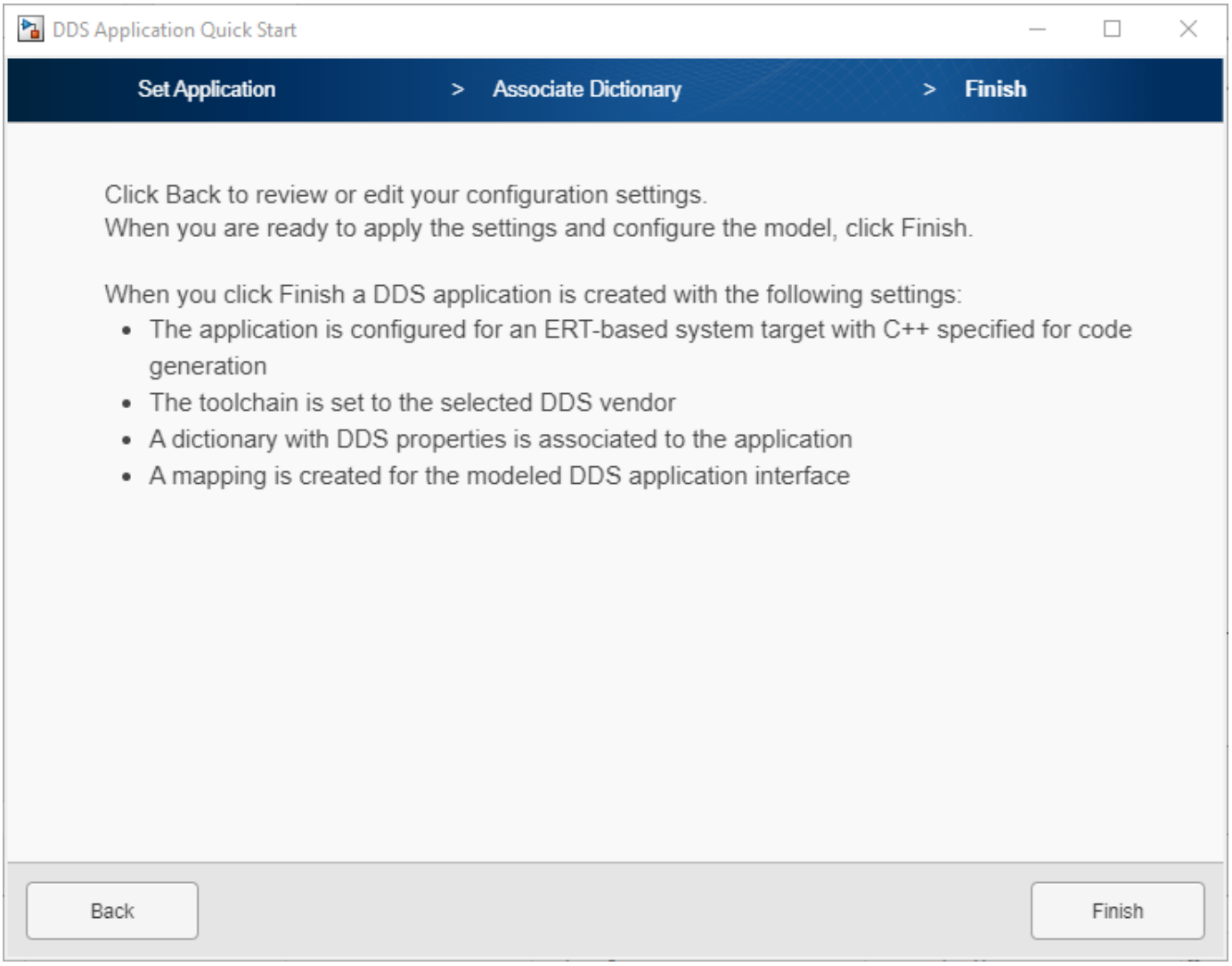
On the second page of the DDS Application Quick Start, specify the source of your DDS definitions for your application. DDS definitions, whether created or imported, are stored in a section of the Simulink data dictionary called the DDS Dictionary. You can create and associate a DDS Dictionary with your application by using one of these options:

- **Use existing dictionary** — If you have an existing DDS Dictionary, you can associate that dictionary with your new DDS application and reuse the definitions. This option is ideal for building applications set to similar requirements or standards.
- **Import from IDL/XML** or **Import from XML**— If you have or would like to specify DDS definitions outside of Simulink and bring them into the Simulink environment you can directly import IDL or XML files when you use RTI or you can directly import XML files when you use eProsima. A new DDS Dictionary is created based on these definitions and then associated with your application. IDL and XML specifications enable maximum flexibility and control over your definitions.
- **Create and use default dictionary** — If you would like to have a new DDS Dictionary created for you based on your application model, you can select to create and use a default dictionary. This option is the quickest and simplest option to get started.




## Transition from DDS to Simulink Environment

To wrap-up, click **Finish**. The Quick Start creates and associates a DDS Dictionary with your application and configures the application so that you can generate an executable specific to RTI or eProsima.




## Review DDS Definitions

### Verify Definitions

To verify your DDS definitions, you can use the DDS Dictionary or Code Mappings editor. To verify Domain, Topic, Type, or QoS definitions, you can use the DDS Dictionary to view and edit these definitions. For more information, see “Manage DDS Definitions” on page 2-2.

To verify imported DataReader and DataWriter definitions, use the Code Mappings editor. To view the readers and writers, open the editor, set the **Configuration Mode** to `Use Reader XML Path` or `Use Writer XML Path` and verify that they appear as drop-down list options for the inports or outports. If you select a reader or writer, you can also verify its Topic and QoS properties loaded in the editor. For more information, see “Interactively Configure DDS Interface” on page 4-2.

#### Troubleshoot

If your imported definitions are not correct, examine and correct any errors in the imported IDL/XML or DDS Dictionary, and then associate the new definitions with your DDS application. To update or change vendor information,on the **DDS** tab, open the **Quick Start** and update the vendor.

## Considerations and Limitations

- Multiple XML Files — The Quick Start allows you to import one XML file. To import additional files, use the DDS Dictionary.
- IDL File Import — IDL import is not directly supported for eProsima. To import IDL specifications, convert IDL to XML and import the XML files.
- Duplicate Data Imported — Imported XML files pull additional XML files referenced by the include mechanism into the DDS Dictionary. If an XML file is referenced multiple times its definitions are imported each time resulting in duplicates.

## See Also

**DDS Dictionary | Code Mappings Editor**

## Related Examples

- “DDS Blockset System Requirements”
- “Manage DDS Definitions” on page 2-2
- “Interactively Configure DDS Interface” on page 4-2

# DDS Definition Management

# Manage DDS Definitions

To manage the DDS aspects of your applications in the Simulink environment, you can use a section of the Simulink data dictionary that contains DDS properties called the DDS Dictionary. The DDS Dictionary is an intuitive graphical interface that enables you to quickly create and edit the DDS Domains, Topics, Data Samples, and Quality of Service (QoS) you need to configure the Publisher, Subscriber, DataReader, and DataWriter aspects of your model. The DDS Dictionary enables you to easily create and configure the DDS aspects of your application without having to delve into the tedious, low-level details of XML or IDL specification.

To create the DDS aspects of your application, open the DDS Dictionary and configure the DDS aspects of your application by working through the graphical interface tabs from left to right:

1. Create DDS data types. To create a DDS application that sends and receives Data Samples, composed of DDS data types, across the DDS network use the **Types** tab of the DDS Dictionary. The DDS Blockset supports the DDS data types Struct, Const, and Enum. If you have numerous types, you can organize these types into libraries and modules for easier access to these types during configuration. For each DDS data type in your dictionary, you can edit the data type and its associated Simulink equivalent data type. These Simulink equivalent data types are used in the application logic in the Simulink environment. For more information, see “Manage Types” on page 2-3.
2. Create Domains and Topics. After you have created DDS data types to send and receive data, you can use the **Domains** tab to configure the Domains and Topics so that your application can publish and subscribe to specific categories of data. Domains specify a portion of the DDS network and a Topic specifies a category of data. You can use the DDS Dictionary to create Domains and configure their names, IDs, and Topics. You can also create Topics and configure their names, Registered Types, and QoS. For more information, see “Manage Domains” on page 2-8.
3. Create Quality of Service (QoS). Finally, you can use the **QoS** tab to import and control QoS policies that specify aspects of the data connection for your DDS application. DDS Blockset provides access to your vendor default QoS profile, a built-in QoS profile library, and the ability to import and edit QoS specified in XML. You can use the DDS Dictionary to apply QoS to Topics or you can use the profiles to specify QoS to the Publisher, Subscriber, DataReader, and DataWriter aspects of your application represented within the model workspace. For more information, see “Manage QoS” on page 2-15.

## See Also

**DDS Dictionary**

## Related Examples

- “What Is DDS?”
- “Manage Types” on page 2-3
- “Manage Domains” on page 2-8
- “Manage QoS” on page 2-15
- “Model DDS Applications” on page 3-2
- “Interactively Configure DDS Interface” on page 4-2

# Manage Types

The DDS middleware platform requires that applications use Data Samples, composed of DDS data types, to send and receive data on the DDS network. When you use DDS Blockset, you configure application model inports and outports as DataReaders and DataWriters that use Data Samples defined in the DDS Dictionary as DDS data types. DDS Blockset supports the DDS data types Const, Struct, and Enum. You can use the DDS Dictionary to import, create, and edit these DDS data types and their Simulink equivalent data types to meet your application requirements.

To import and manage DDS data types for applications:

1. “Import DDS Data Types” on page 2-3 (optional)
2. “View DDS Data Types” on page 2-3
3. “Edit DDS and Simulink Data Types” on page 2-4

To learn how to apply these DDS data types to configure Data Samples for applications, see “Configure Data Samples” on page 2-6 and “Configure DataReaders and DataWriters” on page 2-6.

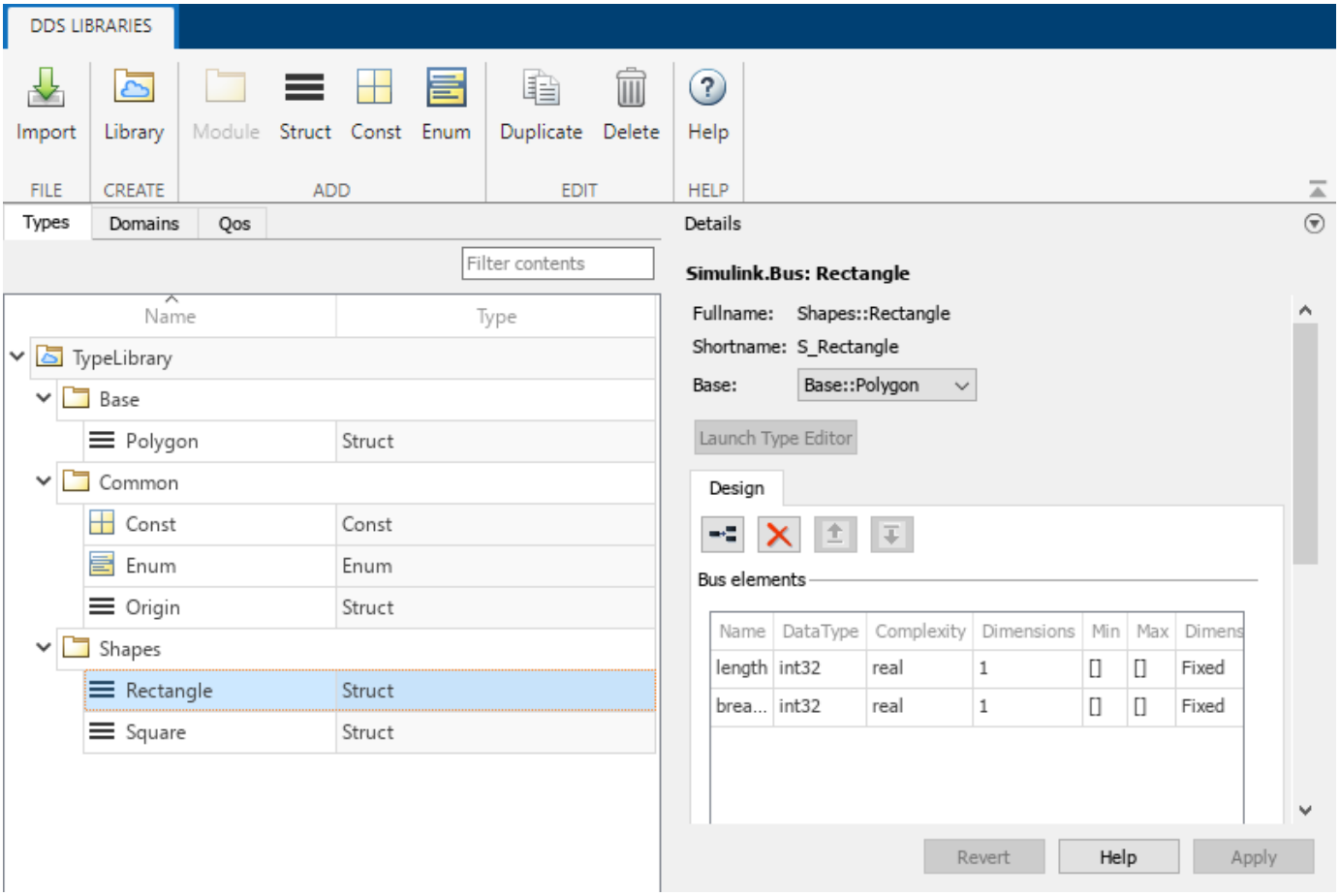
## Import DDS Data Types

If you have previously or would like to specify DDS definitions by using IDL or XML, you can use the DDS Application Quick Start to import those definitions into a DDS Dictionary. If you would like to import more than one XML file or you are working in an existing DDS Dictionary, you can use the DDS Dictionary to import additional XML files. To import from the DDS Dictionary, on the dictionary toolstrip, click **Import** to select and upload additional XML files.

Additionally, if you would like to convert data types in a Simulink data dictionary to DDS data types, you can right-click and choose `Convert to DDS Type` from the context menu to add these data types to a DDS Dictionary.

## View DDS Data Types

To view the DDS data types available for your application, open its associated DDS Dictionary. To open the DDS Dictionary, on the **DDS** tab, click **Code Interface** and select **DDS Dictionary**.




Alternatively, you can open the DDS Dictionary from a Simulink data dictionary. In a Simulink data dictionary, if DDS definitions are available, a **DDS Libraries** node appears in the dictionary. If you open the **DDS Libraries** section of the Simulink data dictionary, in the **Details** pane, you can click **Open DDS Libraries** to open the graphical interface for the DDS Dictionary.

## Edit DDS and Simulink Data Types

You can use the DDS Dictionary to configure these aspects of the DDS data types available for your DDS application:

- “Organize DDS Data Types in the DDS Dictionary” on page 2-4
- “Create New DDS Data Types” on page 2-5
- “Configure DDS Data Type Names” on page 2-5
- “Configure the Values of Simulink equivalent data types” on page 2-5
- “Duplicate DDS Data Types” on page 2-5
- “Delete DDS Data Types” on page 2-5

### Organize DDS Data Types in the DDS Dictionary

To organize large numbers of DDS data types, you can group DDS data types into libraries and then subgroup libraries into modules.

To create a DDS data type library, on the DDS Dictionary toolstrip, click **Library**. To duplicate a library, select a library, and then on the toolstrip, click **Duplicate**. To delete a library, select a library, and then on the toolstrip, click **Delete**.

To add a module within a library, select a library, and then on the DDS Dictionary toolstrip, click **Module**. A module appears nested within the selected library. To duplicate a module, select the module, and then on the toolstrip, click **Duplicate**. To delete a module, select the model, on the toolstrip, click **Delete**.

You can use libraries and nested modules within a DDS Dictionary to organize your DDS data types.

#### Create New DDS Data Types

To create new DDS data types, on the DDS Dictionary toolstrip, click the icon of the data type you would like to create: Const, Struct, or Enum.

#### Configure DDS Data Type Names

To change the name of a DDS data type, in the **Name** column, click and directly edit the spreadsheet.

#### Configure the Values of Simulink equivalent data types

To adjust the properties of the equivalent Simulink data type that represents the DDS data type in the Simulink environment, select the DDS data type to open and view the Simulink data type in the **Details** pane. The equivalent Simulink data types for the supported DDS data types are the following:

| DDS Data Type | Simulink Data Type |
|---|---|
| Const | Numeric MATLAB® variable |
| Enum | Enumeration |
| Struct | `Simulink.Bus` object |

You can configure the Simulink data type properties as available and necessary for your application.

You can view, but not edit, these Simulink equivalent data types in the **Design Data** section of a Simulink data dictionary with the other Simulink data objects that are not a part of the DDS specification. The types shown are kept in sync with their counterparts in the DDS Dictionary, including the renaming and deleting of DDS data types. Simulink uses `Shortname` for data types defined in DDS dictionaries that were created in R2022b. For DDS dictionaries created in R2022a or earlier, Simulink uses `Fullname` for the type.

#### Duplicate DDS Data Types

To duplicate DDS data types, select the data types, and then on the DDS Dictionary toolstrip, click **Duplicate**.

#### Delete DDS Data Types

To delete DDS data types, select the data types, and then on the DDS Dictionary toolstrip, click **Delete**.

## Examples

These examples show how to configure DDS data types from a data management perspective and then how to apply these created DDS data types to configure the inports and outports of DDS

application models so that applications modeled in Simulink can send and receive Data Samples from the DDS network.

### Configure Data Samples

This example shows how to import, create, and configure DDS data types that represent Data Samples.

1. (Optional) Import DDS data types. If you would like to import DDS data types, use the DDS Application Quick Start or DDS Dictionary.
2. Open the DDS Dictionary. On the **DDS** tab, click **Code Interface** and select **DDS Dictionary**.
3. Add a new library. On the DDS Dictionary toolstrip, click **Library**.
4. Create new DDS data types.
   - Create and name a new DDS Struct data type. On the DDS Dictionary toolstrip, click **Struct**. In the **Name** column, rename the data type. Select the new DDS type and in the **Details** pane, view its Simulink equivalent data type, a `Simulink.Bus`object.
   - Create a DDS Const data type. On the DDS Dictionary toolstrip, click **Const**. In the **Name** column, rename the data type. Select the new DDS type and in the **Details** pane, view its Simulink equivalent data type, a numeric MATLAB variable.
   - Create a DDS Enum data type. On the DDS Dictionary toolstrip, click **Enum**. In the **Name** column, rename the data type. Select the new DDS type and in the **Details** pane, view its Simulink equivalent data type, an Enumeration.
5. Delete a DDS data type. Select one of the DDS data types, and on the DDS Dictionary toolstrip, click **Delete**.

### Configure DataReaders and DataWriters

This example shows at a high level how to configure a DDS application model so that the inports and outports that represent DataReaders and DataWriters are configured with DDS data types. The DDS data types enable the application to send and receive Data Samples.

1. Open a model in the DDS Blockset app.
2. Open the DDS Dictionary and create the necessary DDS data types.
3. Construct or adapt the Simulink model so that it acts as a Publisher or Subscriber.
4. Set the inports and outports to DDS data types that match the Registered Type of the Topic that the application subscribes to or publishes.

For more information about how to configure a model and its ports as a Publisher or Subscriber, see “Model DDS Applications” on page 3-2.

## See Also

**DDS Dictionary** | Take DDS Sample | Write DDS Sample

## Related Examples

- “What Is DDS?”
- “Manage DDS Definitions” on page 2-2
- “Manage Domains” on page 2-8

- “Manage QoS” on page 2-15
- “Model DDS Applications” on page 3-2

# Manage Domains

The DDS middleware platform requires that applications specify a Domain, Topic, and Registered Type to publish and subscribe to the DDS network. DDS provides a network, the Global Data Space, that can be partitioned into subnetworks, Domains, and then into categories of data, Topics. To use DDS, your application must specify the Domain it wants to participate in, the Topic it wants to subscribe or publish data to, and the DDS data type for that data, referred to as the Registered Type for the Topic. You can use the DDS Dictionary to import, create, and edit the Domain, Topic, and Registered Types to meet your application requirements.

To create and manage Domains, Topics, and Registered Types for applications:

1. "Import Domains and Topics" on page 2-8 (optional)
2. "View Domains, Topics, and Registered Types" on page 2-8
3. "Edit Domains, Topics, and Registered Types" on page 2-9

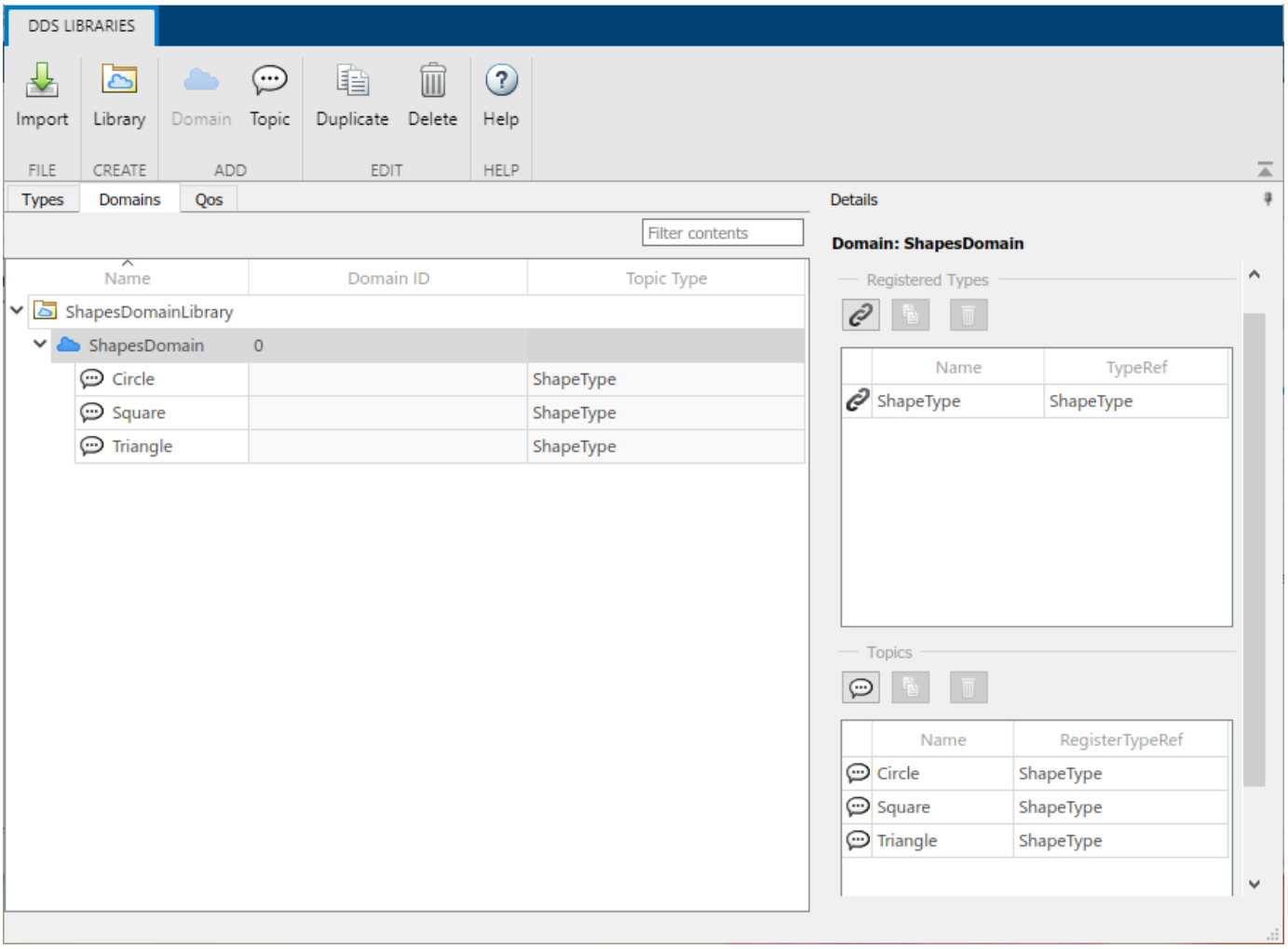
To learn how to create Domains, Topics, and Registered Types, see "Create Domains" on page 2-11, "Create Topics" on page 2-12, and "Register DDS Data Types for Topics" on page 2-12. To learn how to apply these definitions to publish and subscribe to DDS, see "Subscribe to a Topic" on page 2-13, and "Publish a Topic" on page 2-13.

## Import Domains and Topics

If you have previously or would like to specify DDS definitions by using IDL or XML, you can use the DDS Application Quick Start to import those definitions into a DDS Dictionary. If you would like to import more than one XML file or you are working in an existing DDS Dictionary, you can use the DDS Dictionary to import additional XML files. To import from the DDS Dictionary, on the dictionary toolstrip, click **Import** to select and upload additional XML files.

## View Domains, Topics, and Registered Types

To open a DDS Dictionary to configure Domains, Topics, and Registered Types, on the **DDS** tab, click **Code Interface** and select **DDS Dictionary**. In the DDS Dictionary, click the **Domains** tab.




Alternatively, you can open the DDS Dictionary from a Simulink data dictionary. In a Simulink data dictionary, if DDS definitions are available, a **DDS Libraries** node appears in the dictionary. If you open the **DDS Libraries** section of the Simulink data dictionary, in the **Details** pane, you can click **Open DDS Libraries** to open the graphical interface for the DDS Dictionary.

## Edit Domains, Topics, and Registered Types

You can use the DDS Dictionary to configure these aspects of the Domains, Topics, and Registered Types available for your DDS application:

### Edit Domains


- “Organize Domains in the DDS Dictionary” on page 2-10
- “Create New Domains” on page 2-10
- “Configure Domain Names” on page 2-10
- “Configure Domain IDs” on page 2-10
- “Configure Registered Types Domains” on page 2-10

#### Organize Domains in the DDS Dictionary

To organize large numbers of Domains, you can group the Domains into libraries. To create a Domain library, on the DDS Dictionary toolstrip, click **Library**. To duplicate a library, select a library, on the toolstrip, click **Duplicate**. If you duplicate a library the Domains and Topics contained within that library are also duplicated and reproduced in the duplicate library. To delete a library, on the toolstrip, click **Delete**.

#### Create New Domains

To create a new Domain, select a library, and then on the DDS Dictionary toolstrip, click **Domain**.

#### Configure Domain Names

To change the name of a Domain, in the **Name** column, click and directly edit the spreadsheet.

#### Configure Domain IDs

To change a Domain ID, select the Domain. In the **Domain ID** column, click and directly edit the spreadsheet.

#### Configure Registered Types Domains

To configure the Registered Types for Domains, select a Domain. In the **Details** pane, click the link button to add a Registered Type. To edit the name and DDS type for your Registered Type, use the **Name** and **TypeRef** fields. To manage the Registered Types for the Domain, use the duplicate and delete buttons. To register a type for a Topic, the data type must be registered for its Domain.

#### Duplicate Domains

To duplicate a Domain, select the Domain, and then on the DDS Dictionary toolstrip, click **Duplicate**.

#### Delete Domains

To delete a Domain, select the Domain, and then on the DDS Dictionary toolstrip, click the **Delete**.

### Edit Topics



#### Create new Topics

To create a new Topic, select a Domain, on the DDS Dictionary toolstrip, click **Topic**.

#### Configure Topic Names

To change the name of a Topic, in the **Name** column, click and directly edit the spreadsheet.

#### Configure Registered Types Topics

To configure the Registered Type for a Topic, select the Topic. In the **Details** pane, use the Registered Type drop-down list to select from the DDS data type options. The available options are DDS data types that are in the DDS Dictionary and registered with the Domain. If you do not see a DDS data type that you expected, verify or create the DDS data type in the dictionary, and then register the data type with the Domain for it to appear as an option. After the data type is registered, it appears as the **Topic Type** for the Topic.

#### Configure QoS for Topics

To configure the QoS for a Topic, select the Topic. In the **Details** pane, use the Topic QoS drop-down list to select from the available QoS profiles in the DDS Dictionary. If you do not see the QoS profiles that you expected, import the necessary QoS profiles to the DDS Dictionary.

#### Duplicate Topics

To duplicate a Topic, select a Topic, and then on the DDS Dictionary toolstrip, click **Duplicate**.

#### Delete Topics

To delete a Topic, select the Topic, and then on the DDS Dictionary toolstrip, click **Delete**.

### Edit Registered Types

- “Create Registered Types” on page 2-11
- “Duplicate Registered Types” on page 2-11
- “Delete Registered Types” on page 2-11

#### Create Registered Types

To create Registered Types, select a Domain. In the **Registered Types** section of the **Details** pane, click the link button to add a Registered Type. The DDS data types available are the types in the dictionary. To change the name of a Registered Type, click and directly edit the **Name** field. To change the DDS data type of a Registered Type, click in the **TypeRef** field and select from the drop-down options.

#### Duplicate Registered Types

To duplicate a Registered Type, select a Registered Type and click the **Duplicate** button.

#### Delete Registered Types

To delete a Registered Type, select a Registered Type and click the **Delete** button.

## Examples

These examples show incrementally how to create the Domains, Topics, and Registered Types needed to configure DDS applications. The examples then show how to apply these definitions to enable applications to publish and subscribe to data on the DDS network.

### Create Domains

This example shows how to import, create, and configure Domains.

1. (Optional) Import Domains. If you would like to import Domains, use the DDS Application Quick Start or DDS Dictionary.

2. Open the DDS Dictionary. On the **DDS** tab, click **Code Interface** and select **DDS Dictionary**.
3. Click the **Domains** tab.
4. Create a Domain. On the DDS Dictionary toolstrip, click **Domain**.
5. Configure the Domain. To configure the Domain name and Domain ID, click and edit the spreadsheet.
6. Create Topics within the Domain as necessary.

### Create Topics

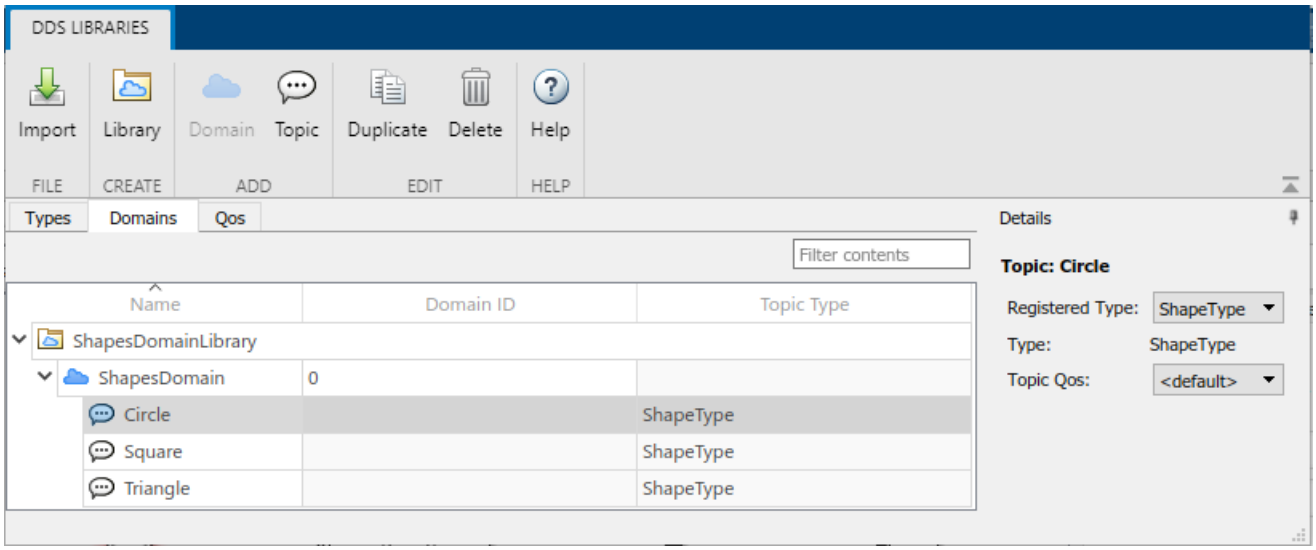
This example shows how to import, create, and configure Topics.

1. (Optional) Import Topics. If you would like to import Topics, use the DDS Application Quick Start or DDS Dictionary.
2. Open the DDS Dictionary. On the **DDS** tab, click **Code Interface** and select **DDS Dictionary**.
3. Topics are contained within Domains. To configure Topics, click the **Domains** tab.
4. Select or create a Domain.
5. Create a Topic. On the DDS Toolstrip, click **Topic**.

### Register DDS Data Types for Topics

This example shows how to create and configure Registered Types for Topics. To configure a Registered Type for a Topic, create the DDS data type in the DDS Dictionary, register the data type with the Domain, and then register the data type with the Topic.

1. Open the DDS Dictionary. On the **DDS** tab, click **Code Interface** and select **DDS Dictionary**.
2. Registered Types are contained within Domains. To configure Registered Types, click the **Domains** tab.
3. Register a DDS data type with the Domain. In the **Details** pane, in the **Registered Types** section, click the link button to create a Registered Type. Adjust the **Name** and **TypeRef** as necessary.
4. Associate the Registered Type for a Topic. In the **Details** pane, in the **Topics** section, the Topics for the selected Domain are displayed. To set the Registered Type for a Topic, click in the **Register TypeRef** field and select from the drop-down options. The options reflect the Registered Types for the Domain.
5. Verify the Registered Type for the Topic. You can view the data type associated with the Registered Type for a Topic in the **Topic Type** column in the spreadsheet. Alternatively, you can select the Topic. In the **Details** pane, the **Registered Type** field shows the Registered Type for the Topic. You can also use the **Registered Type** drop-down to adjust the Registered Type for the Topic.




### Subscribe to a Topic

This example shows at a high level how to use the DDS Dictionary to create a Topic and then configure the application modeled in Simulink to subscribe to that Topic.

1. Create a Topic in the DDS Dictionary.
   a. Open the DDS Dictionary. Topics are contained within Domains. To configure Topics, click the **Domains** tab.
   b. Create a Domain. On the toolstrip, click **Domain**.
   c. Create a Topic. Select the Domain, from the toolstrip, click **Topic**. Click and edit the spreadsheet to configure the Topic.
   d. Registered a DDS data type for the Topic. Select the Domain, in the **Details** pane, register a DDS data type and then apply it to the Topic.
   e. Close the DDS Dictionary.
2. In the application model the inports act as DataReaders and must have the same DDS data type as the Topic they subscribe to. Configure the inports to the Registered Type for the Topic.
3. Configure the DDS interface to subscribe to the Topic.
   a. Open the Code Mappings editor to configure the DDS interface. From the **DDS** tab, click **Code Interface** and select **Individual Element Code Mappings**.
   b. Select the Topic. For the inports in the model, select the Topic specified in the drop-down options by its path (`DomainLibrary/Domain/Topic`).
4. Build and deploy the application to subscribe to the Topic.

For more information about how to configure a model, see “Model DDS Applications” on page 3-2. For more information about how to configure the inports, see “Interactively Configure DDS Interface” on page 4-2.

### Publish a Topic

This example shows at a high level how to use the DDS Dictionary to create a Topic and then configure the application modeled in Simulink to publish for that Topic.

1. Create a Topic in the DDS Dictionary.
   a. Open the DDS Dictionary. Topics are contained within Domains. To configure Topics, click the **Domains** tab.
   b. Create a Domain. On the toolstrip, click **Domain**.
   c. Create a Topic. Select the Domain, on the toolstrip, click **Topic**. Click and edit the spreadsheet to configure the Topic.
   d. Registered a DDS data type for the Topic. Select the Domain, in the **Details** pane, register a DDS data type and then apply it to the Topic.
   e. Close the DDS Dictionary.
2. In the application model, the outports act as DataWriters and must have the same DDS data type as the Topic they publish. Configure the outports to the Registered Type for the Topic.
3. Configure the DDS interface to subscribe to the Topic.
   a. Open the Code Mappings editor to configure the DDS interface. On the **DDS** tab, click **Code Interface** and select **Individual Element Code Mappings**.
   b. Select the Topic. For the inports in the model, select the Topic specified in the drop-down options by its path (`DomainLibrary/Domain/Topic`).
4. Build and deploy the application to publish the Topic.

For more information about how to configure a model, see “Model DDS Applications” on page 3-2. For more information about how to configure the outports, see “Interactively Configure DDS Interface” on page 4-2.

## See Also

**DDS Dictionary | Code Mappings Editor**

## Related Examples

- “What Is DDS?”
- “Manage DDS Definitions” on page 2-2
- “Manage Types” on page 2-3
- “Manage QoS” on page 2-15
- “Model DDS Applications” on page 3-2
- “Interactively Configure DDS Interface” on page 4-2

# Manage QoS

The DDS middleware platform offers several Quality of Service (QoS) policies that can be applied to control the data connection of your application on the DDS network. QoS policies are grouped into sets called profiles, that can be applied to DataWriters and DataReaders, Publishers and Subscribers, or Topics within the DDS network. When you use DDS Blockset, you can apply QoS profiles at each level of specification. DDS Blockset supports the default QoS profile provided by your DDS vendor, a built-in QoS profile included in DDS Blockset, or imported QoS profiles. You can use the DDS Dictionary to import, view, and edit these QoS profiles to meet your application requirements.

To create and manage the QoS for applications:

1. “Import Quality of Service (QoS)” on page 2-15
2. “View Quality of Service (QoS)” on page 2-15
3. “Edit Quality of Service (QoS)” on page 2-16

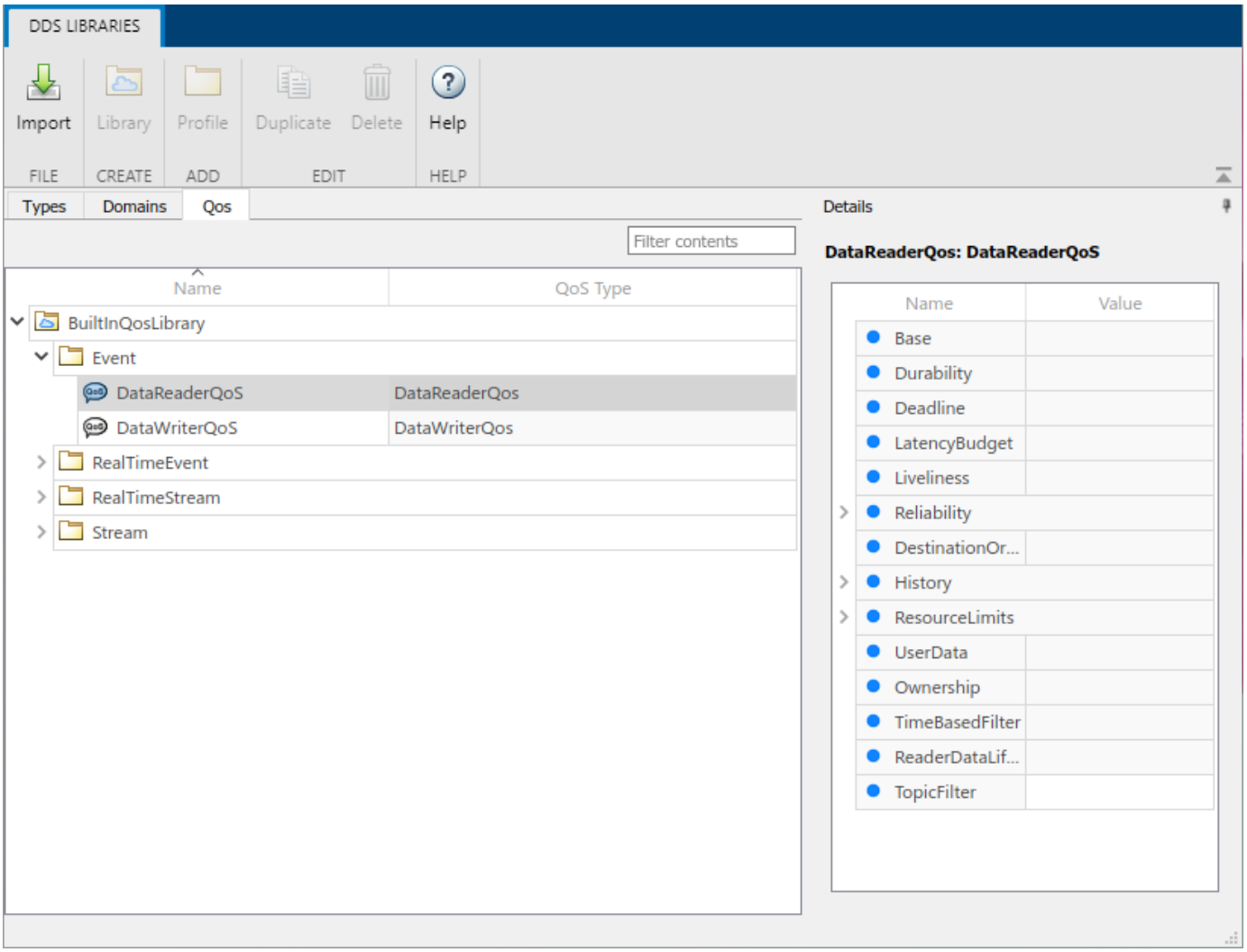
To learn how to apply QoS to your applications, see “Configure QoS for DataReaders and DataWriters” on page 2-18, “Configure QoS for Subscribers and Publishers” on page 2-18, and “Configure QoS for Topics” on page 2-18.

## Import Quality of Service (QoS)

If you have previously or would like to specify DDS definitions by using IDL or XML, you can use the DDS Application Quick Start to import those definitions into a DDS Dictionary. If you would like to import more than one XML file or you are working in an existing DDS Dictionary, you can use the DDS Dictionary to import additional XML files. To import from the DDS Dictionary, on the dictionary toolstrip, click **Import** to select and upload additional XML files.

## View Quality of Service (QoS)

To open a DDS Dictionary to view QoS profiles and policies, on the **DDS** tab, click **Code Interface** and select **DDS Dictionary**. To view QoS, click the **QoS** tab in the DDS Dictionary.




Alternatively, you can open the DDS Dictionary from a Simulink data dictionary. In a Simulink data dictionary, if DDS definitions are available, a **DDS Libraries** node appears in the dictionary. If you open the **DDS Libraries** section of the Simulink data dictionary, in the **Details** pane, you can click **Open DDS Libraries** to open the graphical interface for the DDS Dictionary.

## Edit Quality of Service (QoS)

You can use the DDS Dictionary to view and configure these aspects of the QoS available for your DDS application:

- “View QoS Libraries, Profiles, and Policies” on page 2-17
- “Configure Default QoS” on page 2-17
- “Configure Built-In QoS” on page 2-17
- “Configure QoS Profile Names” on page 2-17
- “Configure the Values of QoS Policies” on page 2-17
- “Duplicate QoS Profiles” on page 2-17

• “Delete QoS Profiles” on page 2-17

#### View QoS Libraries, Profiles, and Policies

QoS policies can be grouped into sets called QoS profiles. These profiles can then be grouped into sets of profiles called libraries. You can view imported libraries, profiles, and policies in the DDS Dictionary **QoS** tab. To adjust the QoS libraries, profiles, and policies available for your application, you must define these definitions in XML and import them into the DDS Dictionary.

#### Configure Default QoS

The default QoS profile is the default provided by your target DDS vendor (RTI or eProsima). To see details of this profile, refer to your vendor documentation.

#### Configure Built-In QoS

DDS Blockset provides a built-in QoS library, composed of four QoS profiles specified by application type:

- Event-based applications
- Real-time event-based applications
- Real-time stream applications
- Stream applications

To access the default QoS profile, if you select the **Create and use default dictionary** option in the DDS Application Quick Start, the built-in library is included in your created default dictionary. If you use a different source of DDS definitions, you can use the DDS Dictionary to import the built-in QoS profile that is included in DDS Blockset, `defaultqos.xml`.

#### Configure QoS Profile Names

To change the name of a QoS profile, in the **Name** column, click and directly edit the spreadsheet.

#### Configure the Values of QoS Policies

To adjust the values of certain QoS policies, select a QoS profile. In the **Details** pane, if a QoS policy is collapsed, you can expand the policy and adjust the value.

#### Duplicate QoS Profiles

To duplicate QoS profiles, select the profile, and then on the DDS Dictionary toolstrip, click **Duplicate**.

#### Delete QoS Profiles

To delete QoS profiles, select the profile, and then on the DDS Dictionary toolstrip, click **Delete**.

## Examples

These examples show how to apply QoS at various levels of your DDS application.

### Configure QoS for DataReaders and DataWriters

In DDS Blockset, application model inports and outports behave as DataReaders and DataWriters. To apply QoS to a reader or writer, you can use the Code Mappings editor and Property Inspector to configure the QoS for individual ports.

1. (Optional) Import QoS. If you would like to import your own QoS or the built-in QoS profile, use the DDS Application Quick Start or DDS Dictionary.
2. Open the DDS Dictionary to view the QoS profiles available for your application. On the **DDS** tab, click **Code Interface** and select **DDS Dictionary**.
3. Configure QoS for DataReaders and DataWriters. If you would like to configure QoS from a previously defined and imported reader or writer, you can import XML to load those definitions into your application. Otherwise, your can use the Code Mappings editor to interactively select a QoS profile for your inport or outport.

For more information about how to configure inports and outports, see “Interactively Configure DDS Interface” on page 4-2.

### Configure QoS for Subscribers and Publishers

In DDS Blockset, the application model behaves as a Publisher or Subscriber and the inports and outports as DataReaders and DataWriters. To apply QoS at the Publisher or Subscriber level, you can simulate this effect by using the Code Mappings editor and Property Inspector to configure all the inports or outports to the same QoS profile. To simulate a Subscriber, configure all the inports to the same QoS profile. To simulate a Publisher, configure all the outports to the same QoS profile.

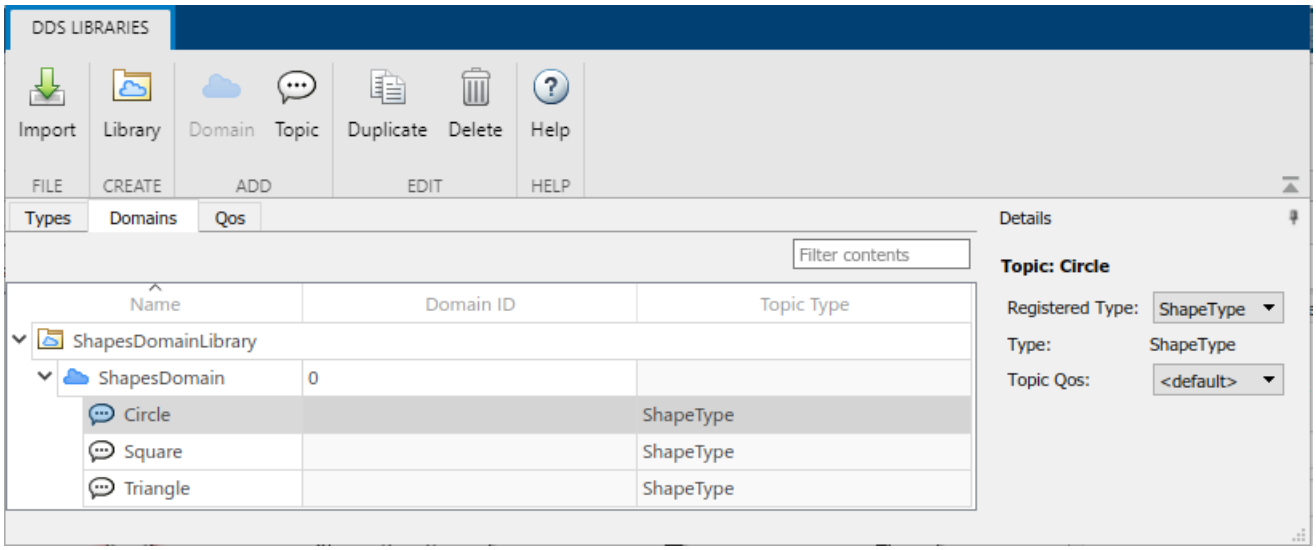
1. (Optional) Import QoS. If you would like to import your own QoS or the built-in QoS profile, use the DDS Application Quick Start or DDS Dictionary.
2. Open the DDS Dictionary to view the QoS profiles available for your application. On the **DDS** tab, click **Code Interface** and select **DDS Dictionary**.
3. Configure QoS for Publishers and Subscribers. If you would like to configure QoS from a previously defined reader or writer, you can import those XML definitions and select the same reader or writer for all inports or outports. Otherwise, you can use the Code Mappings editor to select the same QoS profile for all inports or outports.

For more information about how to configure inports and outports, see “Interactively Configure DDS Interface” on page 4-2.

### Configure QoS for Topics

This example shows how to use the DDS Dictionary to configure the QoS for a Topic.

1. Open the DDS Dictionary.
2. Topics are contained within Domains. To configure Topics, click the **Domains** tab.
3. Select a Topic.




4 Configure the QoS for that Topic. In the **Details** pane, for the **Topic QoS**, select a QoS profile from the drop-down options. By default, Topics are set to the default QoS for the selected DDS vendor. The other options reflect the QoS available in the DDS Dictionary.

## See Also

**DDS Dictionary | Code Mappings Editor**

## Related Examples

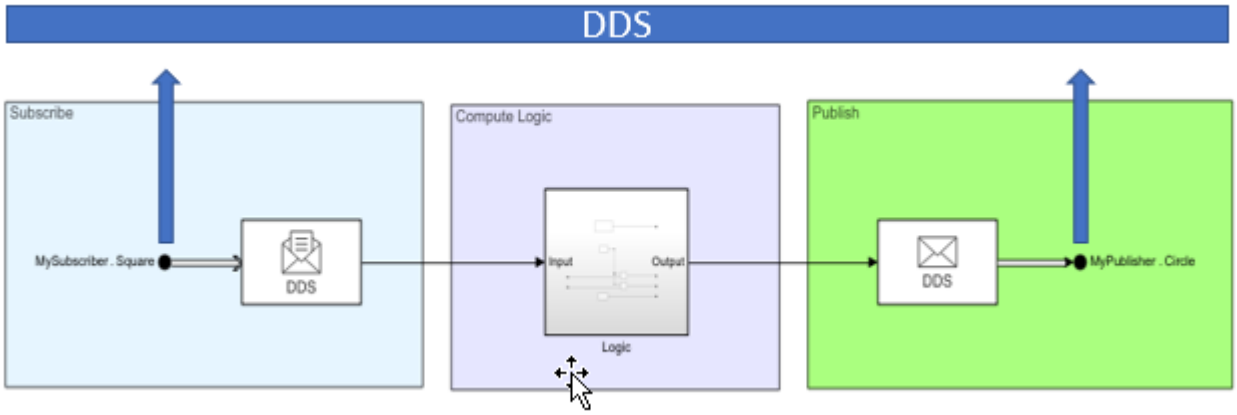
- “What Is DDS?”
- “Manage DDS Definitions” on page 2-2
- “Manage Types” on page 2-3
- “Manage Domains” on page 2-8
- “Interactively Configure DDS Interface” on page 4-2

# Model Architecture

# Model DDS Applications

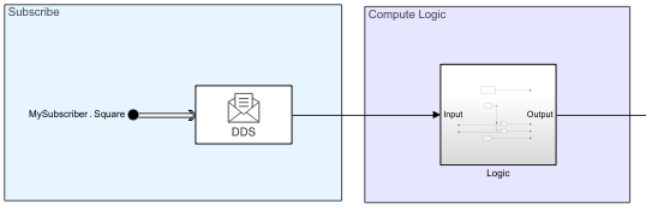
To model a DDS application, construct or adapt a Simulink model so that it can connect to the DDS middleware platform. To prepare a Simulink model so that it can publish and subscribe to the DDS network, the model must be configured as a top model with these modeling aspects:

1. The model must have inports and outports set to DDS data types.
2. The model must have message blocks that send and receive Data Samples from the DDS network. It is recommended that the model uses the DDS Blockset Take DDS Sample and Write DDS Sample blocks. These blocks convert between DDS and Simulink data types to enable the application modeled in Simulink to publish and subscribe to the DDS network.
3. The model itself that contains the DDS application logic. The logic portion of the model is independent of the middleware. The DDS application uses the Simulink equivalent data types to compute its application logic.




You can model DDS applications to act as Subscribers, Publishers, or both.

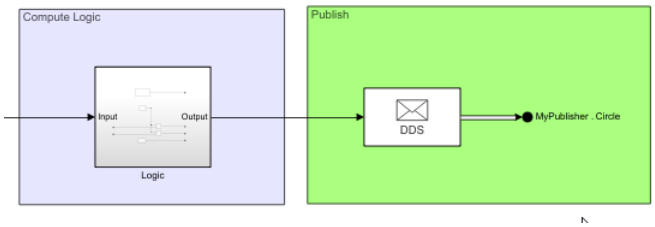
## Model a Subscriber




To configure the Simulink model as a Subscriber:

1. Add an inport and set it to a DDS data type.
2. Insert a Take DDS Sample block to convert the DDS data type to its Simulink equivalent data type.
3. Enclose your application logic in a subsystem. On the boundary of the subsystem, insert In Bus Element ports to accept the Simulink data types from the Take DDS Sample block. For more information, see In Bus Element.

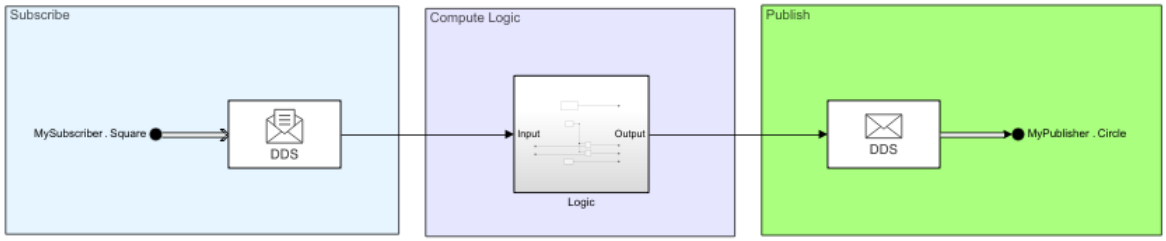
## Model a Publisher




To configure the Simulink model as a Publisher:

1. Enclose your application logic in a subsystem. On the boundary of the subsystem, insert Out Bus Element ports to send Simulink data types to the Write DDS Sample block. For more information, see Out Bus Element.
2. Insert a Write DDS Sample block to convert Simulink data types to DDS data types.
3. Add an outport and set it to a DDS data type.

## Model a Subscriber and Publisher




To configure a Simulink model as a Publisher and Subscriber:

1. Add an inport and set it to a DDS data type.
2. Insert a Take DDS Sample block to convert a DDS data type to its Simulink equivalent data type.
3. Enclose your application logic in a subsystem. On the boundary of the subsystem, insert in bus element ports to receive Simulink data types from the Take DDS Sample block and out bus element ports to send Simulink data types to a Write DDS Sample. For more information, see In Bus Element and Out Bus Element.
4. Insert a Write DDS Sample block to convert Simulink data types to DDS data types.
5. Add an outport and set it to a DDS data type.

## Modeling Pattern Considerations and Limitations

- Modeling – DDS applications must be configured as a top models.
- Ports Configuration – DDS Blockset requires that all ports for must map to DDS. If you have ports that do not map DDS, the model will not simulate or build correctly.

## See Also

Take DDS Sample | Write DDS Sample | **Code Mappings Editor** | **DDS Dictionary**

## Related Examples

- “Manage DDS Definitions” on page 2-2
- “Interactively Configure DDS Interface” on page 4-2

# Model DDS Input Events

DDS Blockset supports event-driven communication where a subscriber application thread waits for an event trigger before executing condition-based algorithms.

In Simulink, you configure DDS events at the input ports of your model and bind the event to a model partition that contains the response algorithm. You can trigger one or more events based on the flow of data into a root-level input port of a model subsystem. Using the Schedule Editor, you can specify a partition to execute in response to each event.

DDS Blockset supports input events for applications using RTI Connext®. In the generated code, the input event is implemented by using WaitSets and Conditions. The WaitSet `wait()` operation blocks execution of the application thread until an input event (or timeout) causes the user-defined condition to be true. For more information about WaitSets and Conditions, see RTI Connext documentation.

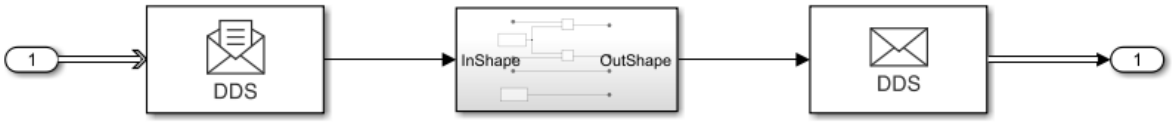
This table lists supported event types and maps DDS callback method names to the corresponding Simulink event name.

| Event Type | DDS Method Name | Simulink Event Trigger Name | Description |
|---|---|---|---|
| onDataAvailable | on_data_available | Input write | Value for input port updates. |
| onSampleLost | on_sample_lost | Input write lost | Input port value update overwrites unprocessed data. |
| onDataDeadlineMissed | on_requested_deadline_missed | Input write timeout | Input port value does not update within a specified amount of time. |

## Model DDS Input Event

This section provides an example of configuring an existing DDS model as the response algorithm for a data available event.

1. Create or open a model configured for DDS. The model should contain Take DDS Sample blocks, application logic, and Inport blocks configured as DDS data types. For more information, see “Model DDS Applications” on page 3-2.




2. Enclose the model in a subsystem. To enclose the model in a subsystem, select all blocks in the DDS model, right-click one of the highlighted elements, and select **Create Subsystem from Selection**. The DDS model is enclosed in a subsystem, and corresponding inport and outport blocks are added at the top level.


3. Configure the model to run tasks concurrently. On the **Modeling** tab, click **Model Settings** to open the configuration parameters:

   a. In the Configuration Parameters dialog box, on the Solver pane, click the arrow next to **Solver details**.

   b. Under **Tasking and sample time** options, select **Allow tasks to execute concurrently on target**.

   c. Click **OK**.

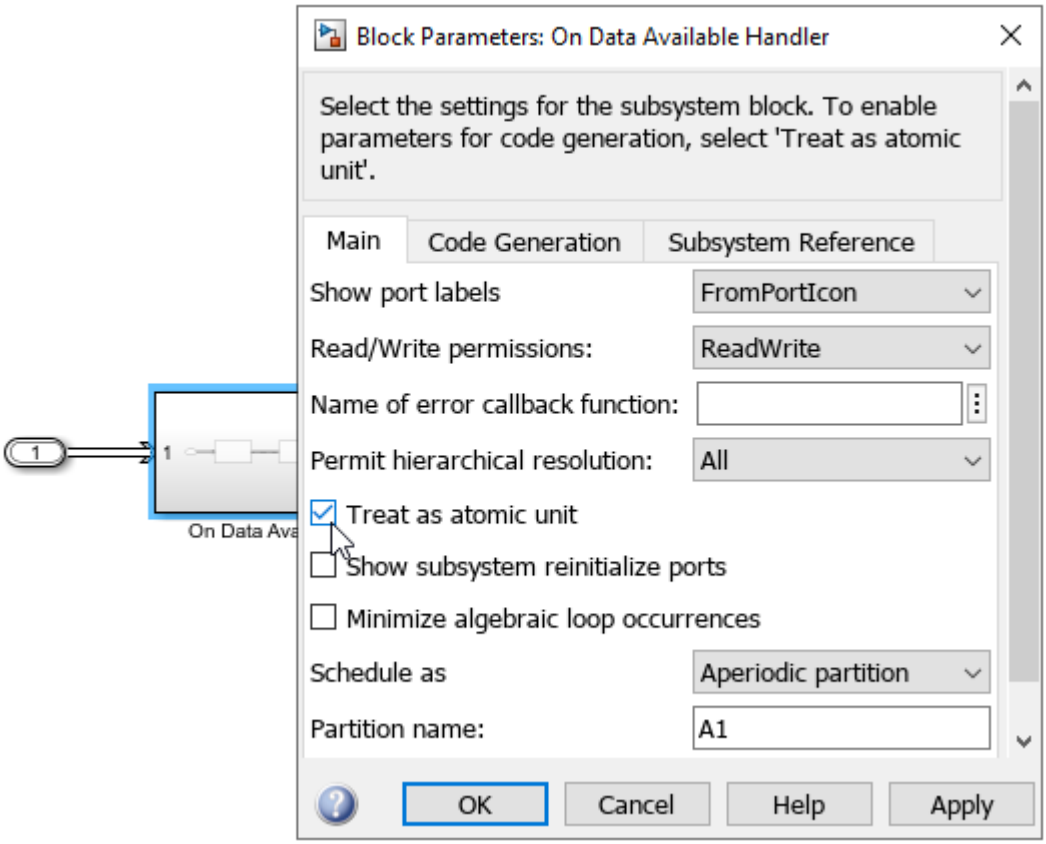
4. Configure the DDS subsystem as an aperiodic partition that can be scheduled to respond to events.

   To configure the subsystem, right-click the subsystem block, select **Block Parameters (Subsystem)**, and then do the following:

   a. In the dialogue box, select **Treat as atomic unit**.

   b. From the **Schedule as** list, select `Aperiodic partition`.

   c. Specify the partition name as `A1`.



d Click **OK**.

After the subsystem is configured, in the top-level model, a badge appears above the subsystem block indicating that the subsystem is partitioned.

5 Configure events on the input port.

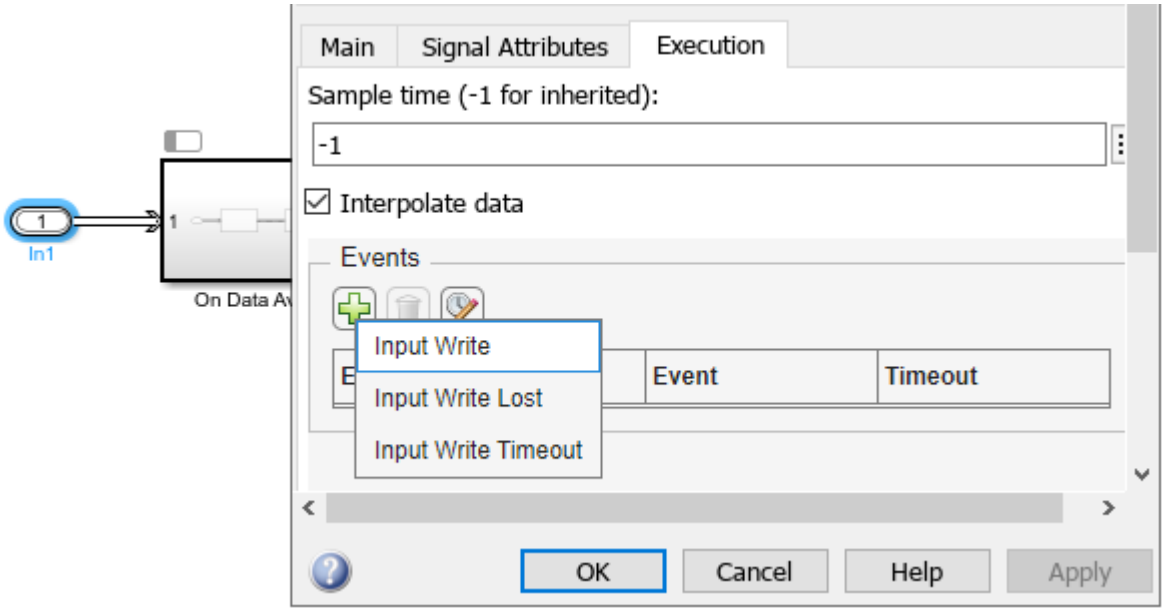
To configure the input port, right-click the Inport block, select **Block Parameters (Inport)**, and then do the following:

a In the dialog box, select the **Execution** tab.

b On the **Execution** tab, under **Events**, click the plus sign to add one or more events:

- `Input Write` (DDS on_data_available) — See `simulink.event.InputWrite` for more information.
- `Input Write Lost` (DDS on_sample_lost) — See `simulink.event.InputWriteLost` for more information.
- `Input Write Timeout` (DDS on_requested_deadline_missed) — See `simulink.event.InputWriteTimeout` for more information.

For this example, select `Input Write`.




Under **Events**, the Events spreadsheet contains event settings for that input port and lists the added events. You can add up to one of each event type. For an `Input Write Timeout` event, you enter the timeout duration in the **Timeout** column. The timeout duration is entered in seconds.

c Click **OK**. After adding the event, an event badge appears above the inport block.

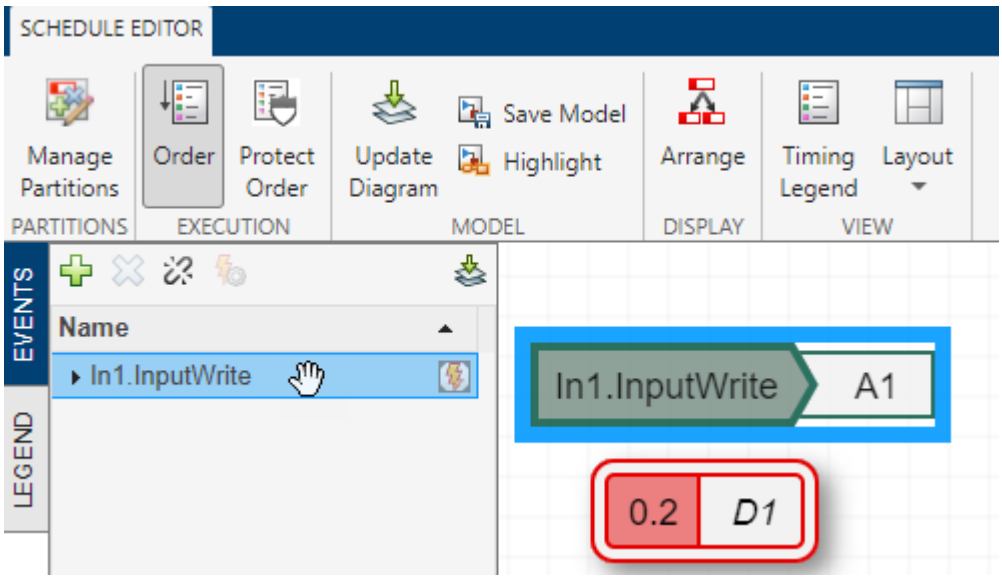
6 Use the Schedule Editor to bind events configured on input ports to partitions that execute on event occurrence. To configure event binding, do the following:

a Click the badge above the subsystem block to open the Schedule Editor. Alternatively, to open the Schedule Editor from the Inport Block, open the Inport Block Parameters dialogue

box, and under the **Execution** tab, in the **Events** section, click  . The Schedule Editor opens.

**b** In the Schedule Editor, click **Update Diagram**, then expand the **Events** tab. The input write event for the input port (`In1.InputWrite`) displays.

**c** Bind the event to the partition by dragging the `In1.InputWrite` event from the Events panel to the `A1` partition shown in the Schedule Editor canvas.




Note that this step might not be required because binding might occur automatically if you configured only one event and one partition.

**d** Schedule additional events or behaviors. See Schedule Editor for more information.

**e** Close the Schedule Editor, then press `Ctrl+D` to update the model. The block diagram updates to reflect the binding of the `In1.InputWrite` event to the `A1` aperiodic partition scheduled as subsystem On Data Available Handler.


## Modeling Pattern Considerations and Limitations

- Modeling – Modeling DDS input events is only supported for DDS Blockset models using RTI Connext 6.0.1+.
- Port Configuration – DDS Blockset requires that all ports must map to DDS and all input ports in models that configure input events must use Inport blocks.

## See Also

Inport | `simulink.event.InputWriteTimeout` | `simulink.event.InputWriteLost` | `simulink.event.InputWrite` | **Schedule Editor**

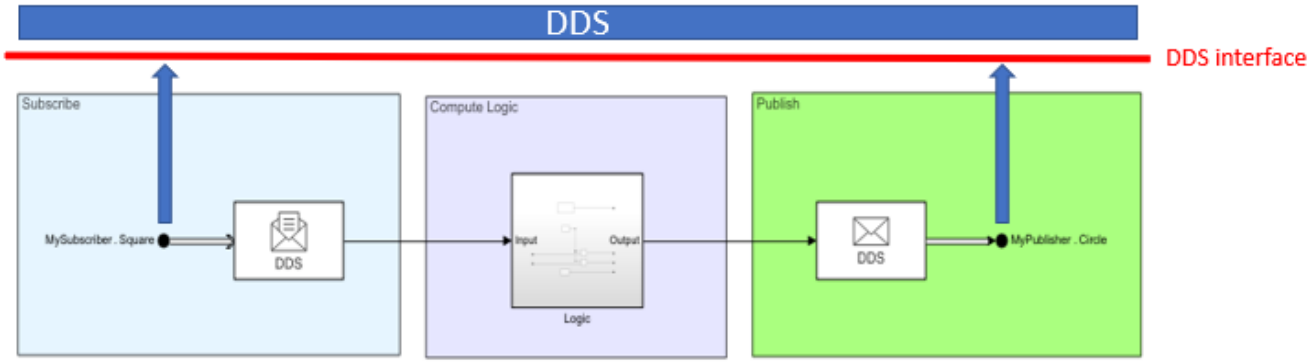
## Related Examples

- “Manage DDS Definitions” on page 2-2
- “Interactively Configure DDS Interface” on page 4-2

# DDS Interface Configuration

# Interactively Configure DDS Interface

To connect a DDS application to publish and subscribe to the DDS network, configure a DDS interface. A DDS interface is the connection point between the application and the DDS network. Specifically, the interface defines the Topic and Quality of Service (QoS) for the DataReaders and DataWriters in an application. In an application model, the inports represent DataReaders and the outports represent DataWriters. To configure an interface, use the Code Mappings editor to configure the ports as readers and writers that meet your application requirements.




## How to Configure a DDS Interface

To configure the interface, you can either configure your model interactively by using default behavior or import XML definitions.

### Configure a Model Interactively by Using Default Behavior

Configuring a model interactively by using default behavior enables a simplified, quick, and intuitive way to configure a DDS interface that does not require prior XML specification of DataReaders and DataWriters. You can use the Code Mappings editor to select Topics for your application and specify to have DDS Blockset generate DataReaders and DataWriters for you to connect to the DDS network. You can also configure additional QoS and Topic filters in the editor.

### Configure a Model by Using Imported XML Definitions

Configuring a model by using imported XML definitions supports using XML specifications for DataReaders and DataWriters to provide greater customization and control over these definitions. Additionally, XML specifications provide the ability to define specific properties of DataReaders and DataWriters. You can use the Code Mappings editor to view the previously defined DataReaders and DataWriters for the model ports and the Topic and QoS properties from those definitions that have been loaded into the application.

### Comparison of Configuration Styles

The differences between these configuration styles are listed in this table.

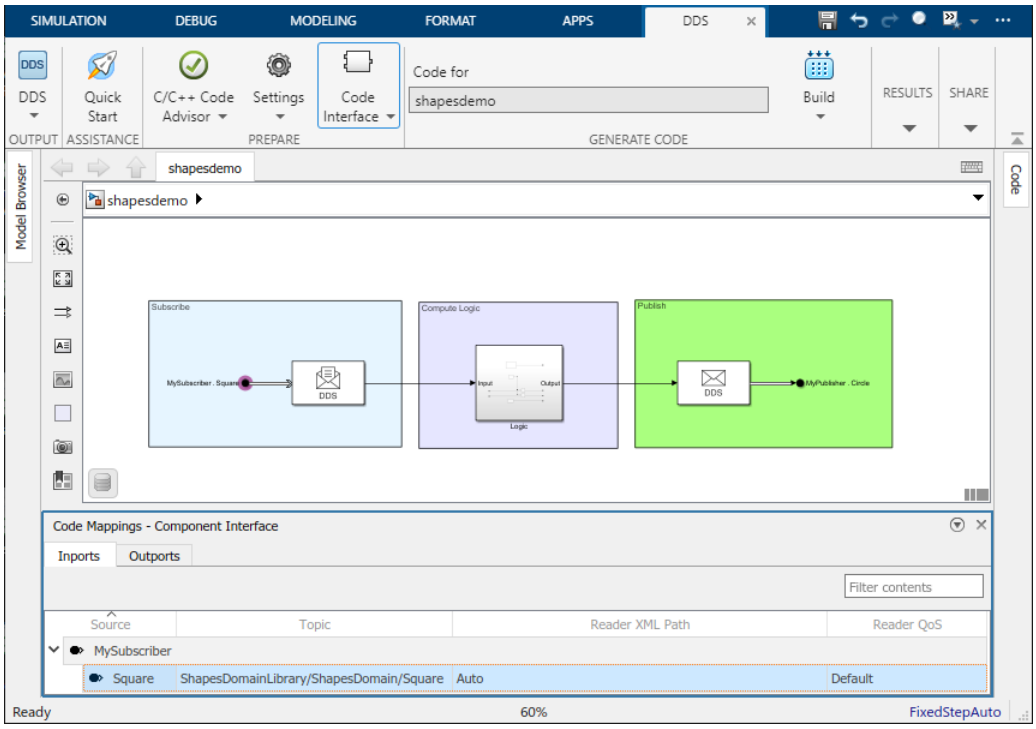
| Configuration Styles | Requirements | Source of Topic and Filter Options | Source of QoS Options | Benefits |
|---|---|---|---|---|
| Configure model interactively by using default behavior | None | • Topics defined in the DDS Dictionary<br>• Configure Topic filters interactively | • Default QoS<br>• Built-in QoS profiles<br>• Imported XML QoS profiles | • Ideal for DDS newcomers<br>• No XML specification necessary<br>• DDS Blockset manages underlying DataReader/ DataWriter specifications so you do not have to manage, create, or troubleshoot these entities |
| Configure model by using imported XML definitions | DataReader or DataWriter must be specified in imported XML | Topics and Topic filters specified within DataReader/ DataWriter definitions | QoS specified within DataReader/ DataWriter definitions | • Easily map DDS applications defined in XML into Simulink<br>• Supports customization and control with explicit DataReader/ DataWriter specifications |

## Configure DDS Interface Interactively by Using Default Behavior

To configure a DDS interface interactively by using default behavior:

**1** Open the environment.

- **a** Open a model in the DDS Blockset app.
- **b** Import or create DDS definitions. Configuring a DDS interface by using default behavior does not require prior XML specifications. You can use the DDS Application Quick Start to create a default dictionary for your application or use an existing DDS Dictionary or imported XML.
- **c** Open the Code Mappings editor to configure the DDS interface. On the **DDS** tab, click **Code Interface** and select **Individual Element Code Mappings**.




2 Select the Topics. For each inport or outport in the application model, select a Topic specified in the drop-down options by its path (`DomainLibrary/Domain/Topic`). The list of drop-down options shows the Topics that are specified with the same DDS data type as the port. If you do not see a Topic that you expected, review the port data type and the Registered Type for the Topic.

3 Select the DataReaders and DataWriters. To use default behavior, set the **Reader XML Path** or **Writer XML Path** to `Auto` to have DDS Blockset automatically generate a reader or writer for the port.

4 (Optional) Configure Topic filters. You can create a filter for a Topic so that your application only receives data relevant to your application without having to implement additional logic (in DDS this is referred to as a Key). To create a filter, in the row of the Topic that you want to filter, select the pencil icon and configure the properties **Filter Kind**, **Filter Expression**, and **Filter Parameter List**. The **Filter Kind** property refers to whether you want to specify your filter by using a string or by using a SQL expression. This choice also determines how the filter is expressed in the exported XML from your application. You can then use the **Filter Expression** and **Filter Parameter List** fields to create your filter.

   The following are simple examples of how to configure filters by using a String Match or a SQL expression:

   - Create a String Match filter — For example, if you want to specify a filter so that you only receive data from a qualitative Topic that publishes squares of different colors, but you want to only subscribe to the colors `Blue` and `Yellow`, you can configure:

     - Set **Filter Kind** to `String Match`
     - Set **Filter Expression** to `Color MATCH %0`
     - Set **Filter Parameter List** to `Blue` and in the next line `Yellow`

     The generated XML code would be the following:

```
<filter name="MyFilter" kind="builtin.stringMatch">
<expression>Color MATCH %0</expression>
<parameter_list>
<param>Blue</param>
<param>Yellow</param>
</parameter_list>
</filter>
```

- Create a SQL filter — For example, if you want to specify a filter so that you only receive data from a numeric Topic, `shapesize`, for values greater than 5 you can configure:
  - Set **Filter Kind** to `SQL`
  - Set **Filter Expression** to `shapesize > 5`

The generated XML code would be the following:

```
<data_reader name="XMLAppCreationTestReader" topic_ref="XMLAppCreationUserTestTopic">
 <datareader_qos base_name="qosLibrary::DefaultProfile"/>
 <filter name="XMLAppCreationUserTestCft" kind="builtin.sql">
 <expression> shapesize &gt; 5 </expression>
 </filter>
 </data_reader>
```

5 (Optional) Configure the QoS. To specify QoS, in the editor, set the **Reader QoS** or **Writer QoS** to a drop-down option. You have several options to configure QoS:

- You can leave the QoS policy as the default. The default QoS profile is the default provided by your target DDS vendor (RTI or eProsima). To see details of this profile, refer to your vendor documentation.
- You can select from built-in QoS profiles. The built-in QoS profiles are profiles provided by the DDS Blockset. These profiles provide a set of QoS policies applicable to event-based, real-time event-based, real-time stream, and stream applications.
- You can select from QoS profiles imported from XML. Imported QoS profiles also appear as options in this drop-down list.

For more information, see “Manage QoS” on page 2-15.

6 Build and deploy application to DDS network.

## Configure DDS Interface by Using XML Definitions

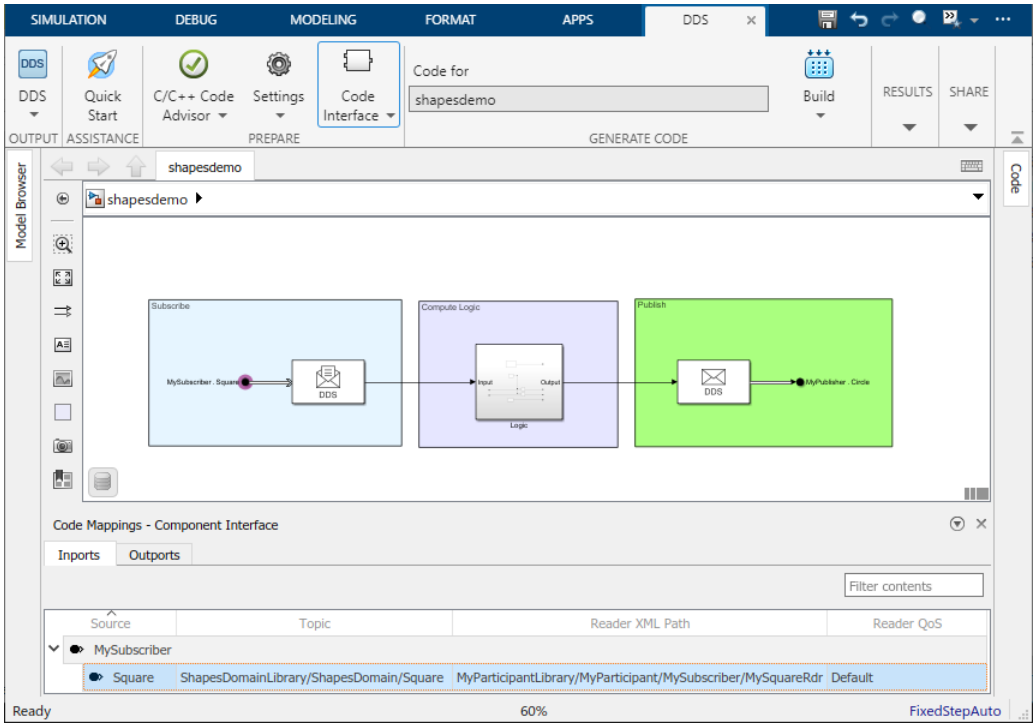
To configure a DDS interface by using imported XML definitions:

1 Open the environment.

a Open a model in the DDS Blockset app.

b Import DDS definitions. Import XML definitions by using the DDS Application Quick Start or DDS Dictionary.

c Configure the DDS interface. Open the Code Mappings editor. On the **DDS** tab, click **Code Interface** and select **Individual Element Code Mappings**.




2 Select the Topics. For each inport or outport in the application model, the Topics appear in the editor as specified in your imported XML and shown by its path (`DomainLibrary/Domain/Topic`). The additional set of drop-down options shows the Topics that are specified with the same DDS data type as the port. If you do not see a Topic that you expected, review the port data type and the Registered Type for the Topic.

3 Select the DataReaders and DataWriters. The DataReaders and DataWriters appear in the editor as specified in your imported XML

4 (Optional) Configure Topic filters. If you are using imported XML definitions, the keys fields are pulled from the XML definitions of the DataReaders and are viewable in a dialog box as read-only when you select the pencil icon. If you would like to edit these properties, you can edit and reimport the source XML file that contains the definitions for your DataReaders. Alternatively, you can select a different DataReader and manually configure the key fields.

5 (Optional) Configure the QoS. To specify QoS, in the editor, the **Reader QoS** or **Writer QoS** should already be filled with the QoS defined in your XML definitions for your selected reader or writer. You have several options to configure QoS:

- You can leave the QoS policy as the default. The default QoS profile is the default provided by your target DDS vendor (RTI or eProsima). To see details of this profile, refer to your vendor documentation.
- You can select from built-in QoS profiles. The built-in QoS profiles are profiles provided by the DDS Blockset. These profiles provide a set of QoS policies applicable to event-based, real-time event-based, real-time stream, and stream applications.
- You can select from QoS profiles imported from XML. Imported QoS profiles also appear as options in this drop-down list.

For more information, see “Manage QoS” on page 2-15.

6 Build and deploy application to DDS network. Build the model and use the executable to deploy the application.

## Considerations and Limitations

- Topic filters — Topic filters are not available for RTI Connext Miro or eProsima.
- DDS Definitions — The DDS Topics and QoS for your application are retrieved from the DDS Dictionary associated with your application model. Ensure that this dictionary is on your MATLAB path.
- Data Types — Inports and outports must have the same DDS data type as the Topic that they subscribe to or publish.
- Unique Reader/Writer Mapping — Inports and outports must map to unique DataReaders and DataWriters. For example, two different inports cannot map to the same DataReader, and two different outports cannot map to the same DataWriter.
- Accurate Data Management — Inports and outports must map to Topics and Quality of Service (QoS) definitions that can be found in the associated DDS dictionary or XML. For example, if you map the ports for a DDS application, and then remove Topics or QoS profiles from the dictionary, you might create invalid mappings where previously mapped ports now are configured to deleted definitions.
- Mixed Mapping Configuration Modes — When you configure a DDS interface, you can use different configuration modes to map different inports and outports. For example, you can use `Use Topic and QoS` to configure one inport and `Use Reader XML Path` to configure a different inport within the same model.

## See Also

**Code Mappings Editor | DDS Dictionary**

## Related Examples

- “What Is DDS?”
- “Import or Create DDS Definitions” on page 1-2
- “Manage DDS Definitions” on page 2-2
- “Model DDS Applications” on page 3-2
- “Deploy DDS Applications” on page 5-2

# Deployment

# Deploy DDS Applications

DDS Blockset connects applications modeled in Simulink to DDS by providing out-of-the-box support for the DDS vendors RTI and eProsima. To use out-of-the-box DDS, create and model a DDS application in Simulink, set up the environment, and use Embedded Coder to build the application model. The build creates exported XML, generated C++ code, and an application executable that you can use to directly connect to the DDS network.

## Build and Deploy DDS Applications

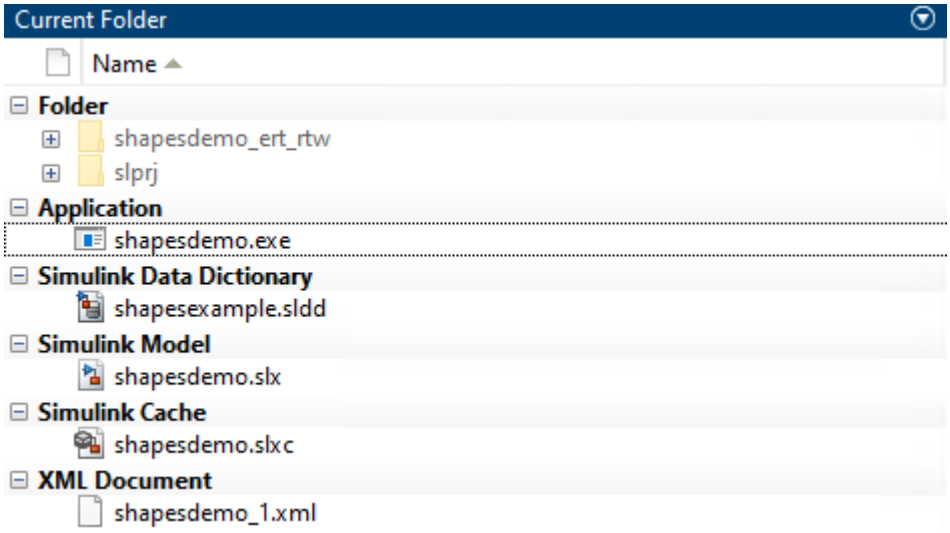
To deploy your application on the DDS network:

1. Ensure that your model is configured correctly. Verify that the model ports are configured and mapped appropriately for DDS. For more information, see “Interactively Configure DDS Interface” on page 4-2.
2. Set up the environment. DDS Blockset generates an executable specific to the DDS vendor that you select, RTI or eProsima. To verify or change your vendor selection, you can use the Configuration Parameters dialog box to review the toolchain setting for your application. To build an executable of your application, set up your environment in a supported platform with a supported C++ compiler. If your target vendor is eProsima, additional setup is not required. If your target vendor is RTI, you must also install RTI Connext. For more information, see “DDS Blockset System Requirements”.
3. Build the application model. On the **DDS** tab, click **Build**.
4. Run the executable and connect your application to the DDS network.

## Overview of Generated Files

When you build your DDS application model, the following folders and files are generated in your current working folder:

- Application executable — The executable that you can deploy to connect the application to the DDS network.
- Embedded Coder build folder — The generated C++ code files.
- Simulink project folder (slprj) — The model simulation files.
- Simulink Data Dictionary file — The associated DDS Dictionary (`.sldd`) file.
- DDS Application model — The Simulink model for the application.
- Exported XML/IDL file — The XML/IDL specifications of your DDS application.




You can use these generated files to analyze, deploy, and port your DDS application. Additionally, you can use the packNGo functionality to relocate and rebuild your application.

## Portability of DDS Applications

To relocate, unpack, and rebuild your DDS application in another development environment, you can use packNGo. The packNGo function enables you to relocate files so that you can recompile for a specific target environment or rebuild in a development environment where MATLAB is not installed. By default, the function packages the files as a flat folder structure in a ZIP file within the code generation folder. After you relocate the ZIP file, use a standard ZIP utility to unpack the compressed file.

To configure your model to build with packNGo:

1. Open the Configuration Parameters dialog box.
2. Select **Pack code and artifacts**.
3. On the toolstrip, click **Build**.

For more information, see `packNGo` (Embedded Coder).

## Implementation Details and Generated C++ Code

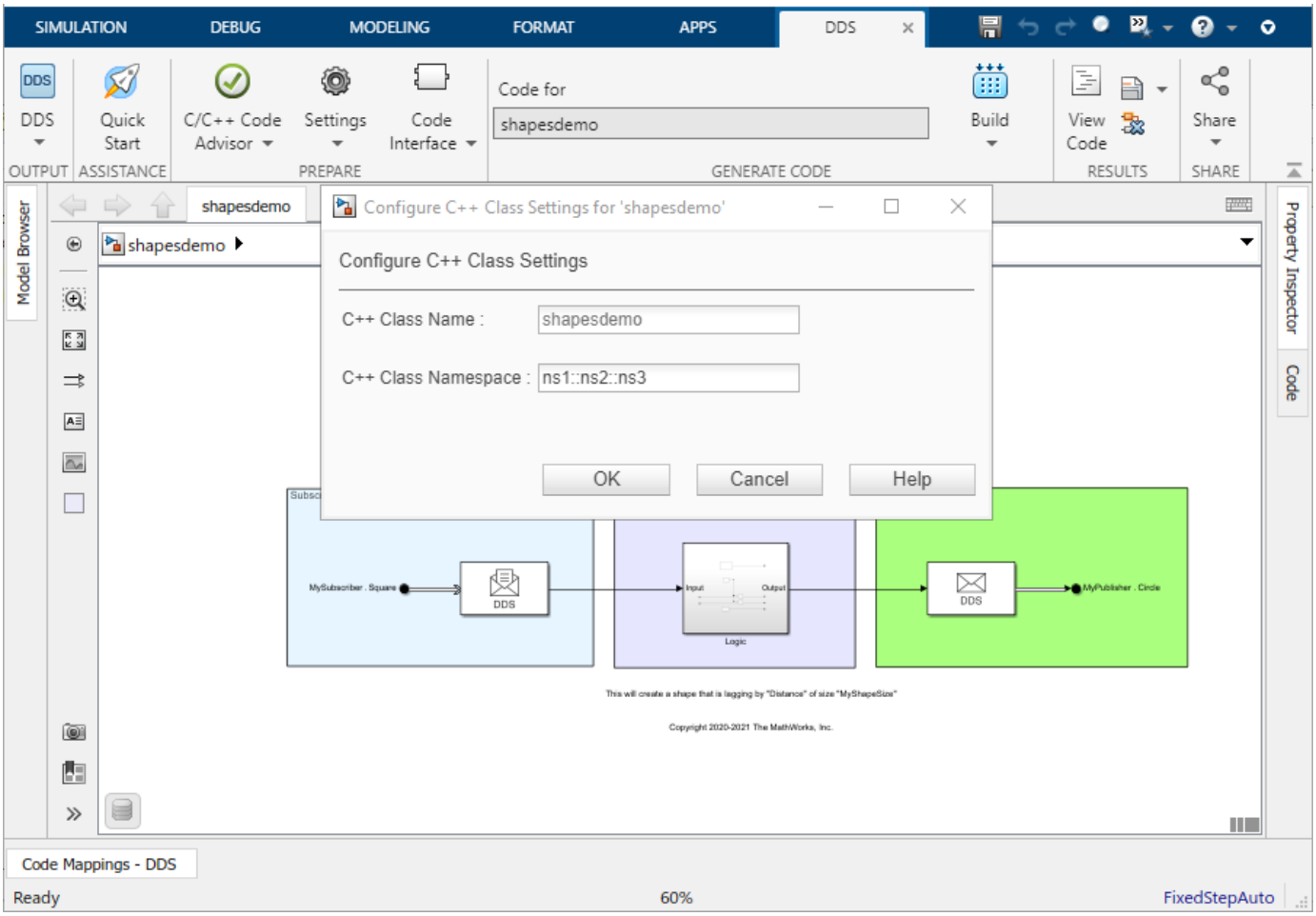
The implementation of DDS is specified by the Object Management Group (OMG) standard and implemented by several vendors in several different programming languages. The DDS Blockset provides out-of-the box integration with the DDS vendors RTI and eProsima. Specifically, the blockset supports the C++ implementation of the DDS standard provided by RTI and eProsima. If you are interested in these vendor APIs, refer to your vendor documentation.

The basic architecture of the generated C++ code is that the application is composed of message classes, vendor helper classes, and the main file. The message classes enable the application to send and receive data. The vendor helper classes are specific to the vendor and load the application profile, register the data types, create and initialize the DDS entities, and wrap the send and receive message classes specific to the vendor API. The main file then performs the application logic. If you would like to examine the generated C++ code, view the Embedded Coder build folder.

## Generated C++ Class Name and Namespace Customization

If you would like to customize the generated C++ code for your DDS application, you can control the generated class name and namespace for your DDS application model interactively or programmatically.

To interactively configure these aspects of the generated code, from an open model, on the **DDS** tab, click **Code Interface**, select **Class Name & Namespace**, and customize the names in the opened configuration dialog box.




To programmatically configure the class name and class namespace, use the Embedded Coder API functions `getClassName`, `setClassName`, `getClassNamespace`, and `setClassNamespace`. For more information, see Configure class namespace (Embedded Coder).

## Debug and Troubleshoot

A few common build issues you can troubleshoot are the following:

### Incorrect environment setup

- Description — If you select RTI as your vendor but do not install RTI Connext, then you are unable to deploy your application.

- Action — Download and install RTI Connext.

#### Missing or invalid mapping for inports and outports in application model

- Description — If the inports or outports have not been configured correctly the model does not build.
- Action — Map the inports and outports in the application model to DDS Topics and configure the ports with the corresponding DDS data types.

#### Inconsistent data management of the DDS definitions

- Description — If you map an inport or an outport to a Topic and then delete or change the data type for the Topic the model does not build.
- Action — Verify that the DDS definitions are available in the associated DDS Dictionary.

## Considerations and Limitations

- RTI Connext Micro adapter — The default network interface names for your system may differ from the default interface names used in the RTI Micro Adapter, `RTIMicroAdapter.hpp`. In such cases, the participant may not initialize properly. Check and update the strings used for the names and manually rebuild the executable.
- RTI Connext Micro DataReader and DataWriter capacity limits — The default number of allocated DataReaders (remote_reader_allocation) and DataWriters (remote_writer_allocation) for an application is 48. Depending on other DDS applications running and your network conditions, the number of readers and writers my exceed 48. You can configure the `RTIMicroQosDefn.inl` to increase this limit or you can change to a Domain where less readers and writers are currently occupied.
- Folder Names with Special Characters — The build process might produce an error if a build-related folder path contains:
  - Unicode characters that do not belong to the system locale.
  - A Japanese (multibyte) character where the final byte is equal to the `5c` hexadecimal character. The make and compiler tools might incorrectly interpret the final byte as the `'\'` (backslash) character.
- DDS Target Specification — DDS Blockset does not support compiling the code generated from a DDS application model for a non-DDS application
- DDS Definitions — The DDS Topics and QoS for your application are retrieved from the DDS Dictionary associated with your application model. Ensure that this dictionary is on your MATLAB path to appropriately build your model.
- Code Generation Data Types — The generated C++ code does not provide support for certain data types. Multidimensional arrays are not supported for code generation.
- Security — There are security risks inherent in communication platforms. These risks include the potential of malicious users to attempt to listen to or spoof DDS communication. Additionally, late-joining readers can potentially access previously transmitted data. To increase protection against these security risks, download and use the secure version of your vendor. The version of eProsima included with DDS Blockset is not the secure version.

## See Also

## Related Examples

- “Build Process” (Embedded Coder)
- “Interactively Configure DDS Interface” on page 4-2
- “DDS Blockset System Requirements”

## External Websites

- RTI Connext

# Configure DDS Application Model for External Mode Simulation

DDS Blockset provides validation support by enabling you to configure and run your DDS Blockset application models in Simulink external mode. External mode establishes a TCP/IP communication channel between a Simulink model and the corresponding DDS application generated from the model.

With external mode, you can enable parameter tuning and signal monitoring, which allows you to modify or tune block parameters in real time. When you change a parameter value in your model, Simulink downloads the updated value to the executing DDS application, allowing you to monitor the system response.

To configure your DDS application model for external mode:

1. In Simulink, create or open a model configured for DDS.

   Use a model that contains message blocks that send and receive data samples from the DDS network, application logic, and inports and outports configured as DDS data types. For more information, see “Model DDS Applications” on page 3-2.
2. Open the **DDS Application Designer** app.
3. Open the Configuration Parameters. Under **Code Generation**, click **Interface** to display the options.
4. In the **Interface** pane:
   a. Select **External mode**.
   b. Under **External mode configuration**, set **Transport layer** to `XCP over TCP/IP`.

      Use the defaults for the rest of the external mode configuration settings.
   c. Click **Apply**.

   After you click **Apply**, the **Hardware** tab opens.
5. In the Simulink canvas, select signals in your model that you want to monitor, left-click the selected signals, and select `Log Selected Signals` to enable logging.
6. On the **Hardware** tab, click **Monitor and Tune**.

   The system generates code, deploys the code, and connects to the application.

You can use the Simulation Data Inspector or other monitoring methods to observe the signals that you selected in the model.

For more information about running and monitoring external mode simulations, see “External Mode Simulation by Using XCP Communication” (Embedded Coder).

## See Also

## Related Examples

- “External Mode Simulations for Parameter Tuning, Signal Monitoring, and Code Execution Profiling” (Embedded Coder)
- “Deploy DDS Applications” on page 5-2

# DDS Examples

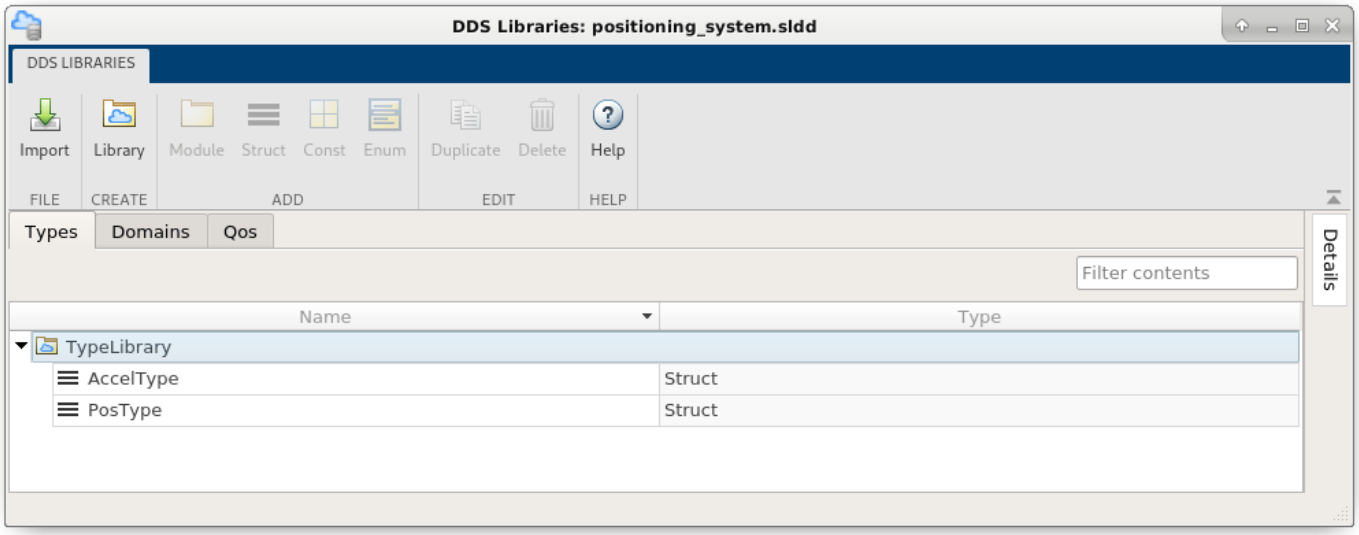
# DDS Positioning System Application

This example shows how to import, model, and deploy a DDS application. The example application is a multi-sensor positioning system designed to estimate the position of a vehicle. The positioning system is composed of three components: a sensor component, an estimation component, and a display component. In the positioning system, the sensors send data to the estimation component that calculates the vehicle position estimate that it sends to the display component to show as a visual representation of the estimated vehicle position.

### Import DDS Definitions for Positioning System

For the positioning system, the DDS definitions (for example, Domains, Topics, Types, and QoS policies) were specified in XML. In the general DDS Blockset workflow, you use the DDS Application Quick Start to import these XML definitions. For this example, the XML has already been imported. To view the XML specifications for the positioning system, open the XML file `positioning_system.xml`. To view the representation of these definitions in the DDS Blockset, open the DDS Dictionary:

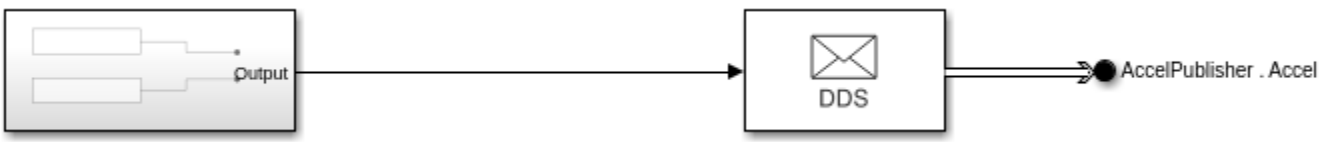
```
open positioning_system.sldd
```



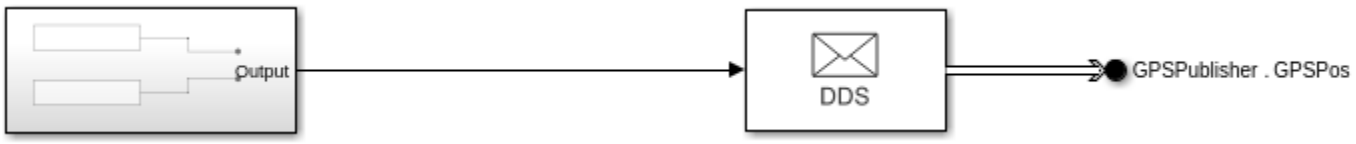

### Model and Configure Sensor Component

The first component in the positioning system is the sensors. The positioning system uses an accelerometer and GPS, each represented by a model, to estimate the vehicle position. Each sensor model is constructed as a Publisher. To view the model structure and interface, open the models:

```
open_system('ex_accelerometer');
```




```
open_system('ex_gps');
```




In each sensor model, you can view each aspect of a Publisher modeled in Simulink:

1. The Sensor model logic, which is composed of a Function block that simulates sensor input data.
2. The Write DDS Sample block that converts the sensor data from a Simulink data type to a DDS data type.
3. The Bus Element Out block that sends the DDS data type.

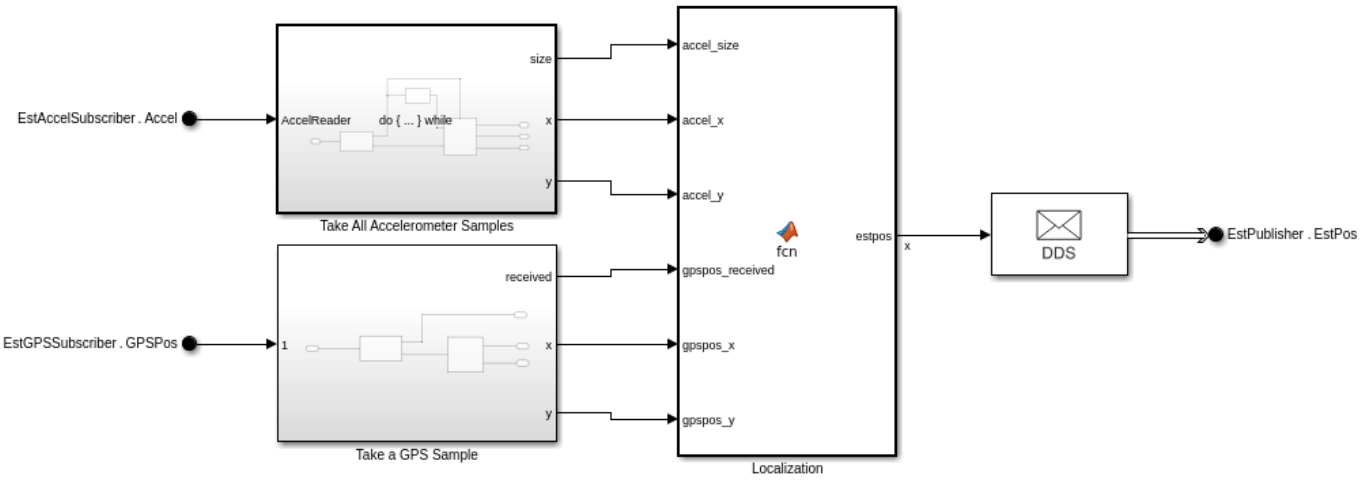
In addition to the modeling of the sensors, the DDS interface has been configured for each sensor model. You can view the DDS interface for each sensor in the Code Mappings editor:

1. On the toolstrip, click the **Code Interface** and select **Individual Elements Code Mappings**.
2. To view the imported Topic and DataWriter for each sensor, click on the **Outports** tab.

### Model and Configure Position Estimate Component

The second component is the estimate model. The estimate model receives data from the sensors and calculates the estimated position of the vehicle. The estimation model is constructed as a Subscriber to the sensors and as a Publisher to the display component. To view the model structure and interface, open the model:

```
open_system('ex_positionestimator');
```




In the estimation model, you can view each aspect of a Subscriber and Publisher modeled in Simulink:

1. The estimation model has two Bus Element In blocks configured as DDS data types to receive the sensor data.
2. The model has a Take DDS Sample block attached to each Bus Element In block to convert the DDS data types to Simulink data types.
3. The estimation model computes the logic to estimate the vehicle position.

4 The model uses the Write DDS Sample block to convert the Simulink data types to DDS data types.

5 The estimate model uses an Bus Element Out block to send the estimate as a DDS data type to the display component.

In addition to the modeling of the estimation model, the DDS interface has been configured for the estimation model. You can view the DDS interface in the Code Mappings editor:

1 On the toolstrip, click the **Code Interface** and select **Individual Elements Code Mappings**.

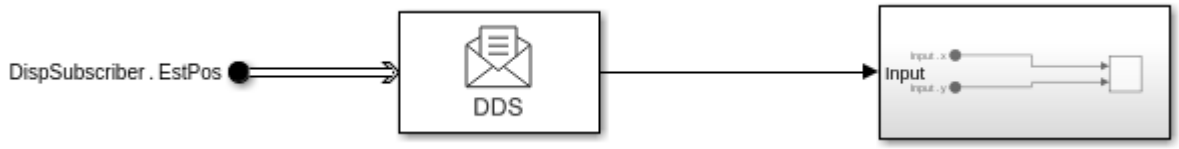
2 To view the imported Topics and DataReaders, click on the **Inports** tab.

3 To view the imported Topics and DataWriter, click on the **Outports** tab.

### Model and Configure Display Component

The third component is the display model. The display model receives and graphically displays the estimated position of the vehicle. The display model is constructed as a Subscriber to the estimation component. To view the model structure and interface, open the model:

```
open_system('ex_resultdisplay');
```




In the display model, you can view each aspect of an application that acts as a Subscriber modeled in Simulink:

1 The estimation model has an Bus Element In block configured as a DDS data type to receive the vehicle position estimate.

2 The model has a Take DDS Sample block to convert the DDS data type to the Simulink data type so that it can display the vehicle estimate.

3 The model logic of the display component graphically shows the estimate of the vehicle position.

In addition to the modeling the display model, the DDS interface has been configured. You can view the DDS interface in the Code Mappings editor:

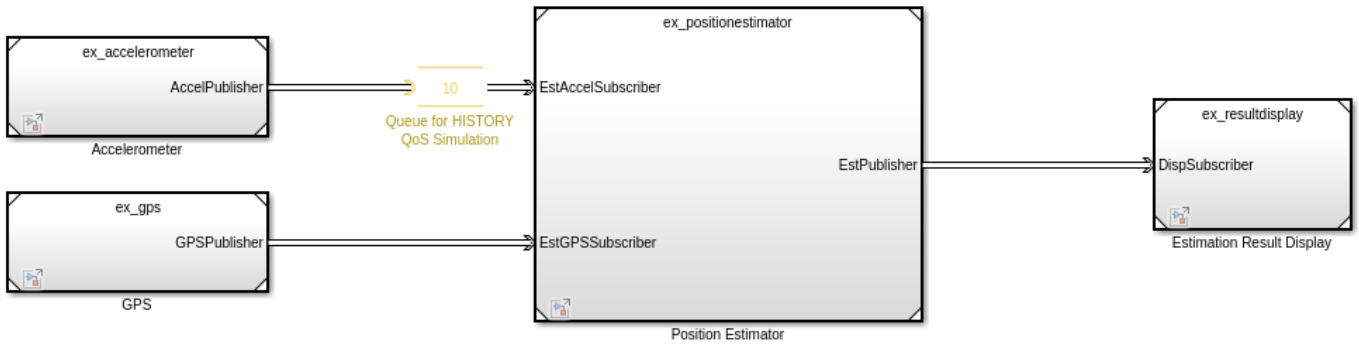
1 On the toolstrip, click the **Code Interface** and select **Individual Elements Code Mappings**.

2 To view the imported Topic and DataReader to recieve the estimate, click on the **Imports** tab.

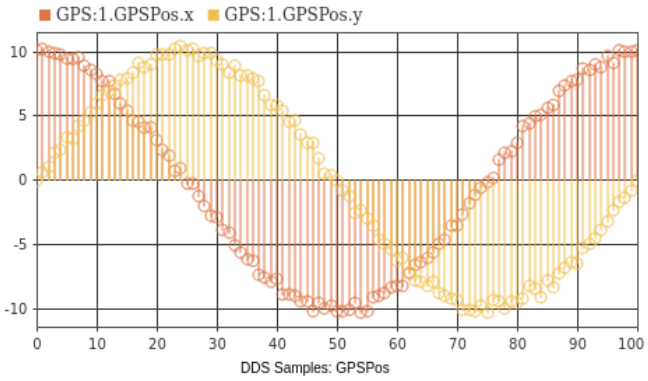
### Build and Deploy Positioning System

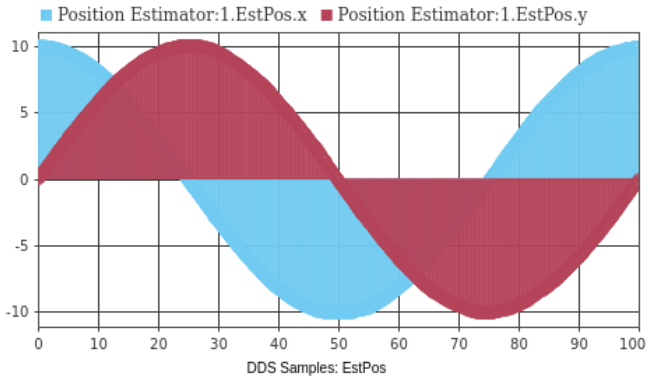
To visually show the DDS positioning system application, this example simulates the DDS network to show the results of the positioning system. A Queue block is placed between Accelerometer and Position Estimator model blocks to simulate HISTORY QoS. When you simulate the application, you can see on the dashboard that the position estimation system produces a more accurate position of the vehicle than the sensor input.

```
open_system('ex_positioningsystem');
```



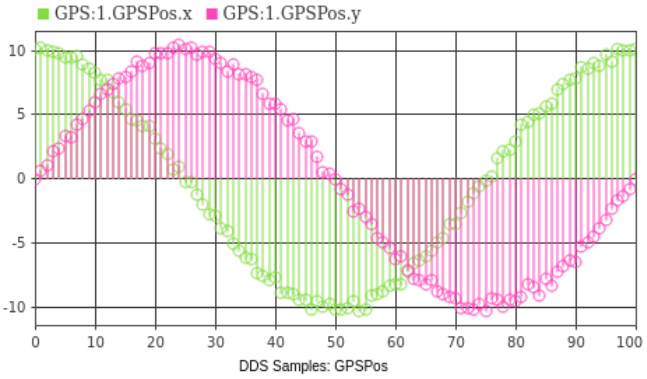

```
set_param('ex_positioningsystem/DDS Samples: GPSPos','TimeSpan','10');
set_param('ex_positioningsystem/DDS Samples: EstPos','TimeSpan','10');
sim('ex_positioningsystem');
set_param('ex_positioningsystem/DDS Samples: GPSPos','TimeSpan','100');
set_param('ex_positioningsystem/DDS Samples: EstPos','TimeSpan','100');
```

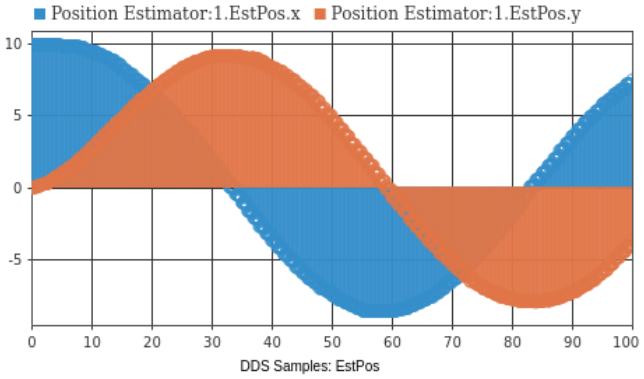



To see the effect of Quality of Service (QoS) on the estimate, if the QoS is not honored the results are inaccurate.

```
set_param(['ex_positioningsystem/Queue for HISTORY' newline 'QoS Simulation'], ...
    'Commented','through')
set_param('ex_positioningsystem/DDS Samples: GPSPos','TimeSpan','10');
set_param('ex_positioningsystem/DDS Samples: EstPos','TimeSpan','10');
sim('ex_positioningsystem');
set_param('ex_positioningsystem/DDS Samples: GPSPos','TimeSpan','100');
set_param('ex_positioningsystem/DDS Samples: EstPos','TimeSpan','100');
```




After reviewing the simulation of the positioning system, if you would like to deploy this example, you can build the `ex_positionestimator` model and use the executable to deploy this application on the DDS network.

## See Also

## Related Examples

- “DDS Blockset Shapes Demo”

## External Websites

- RTI Connext

# DDS Blockset Shapes Demo

This example shows how to import, configure, and deploy the Shapes Demo provided by DDS vendors to introduce DDS concepts. In the demo there are various shapes - Circles, Squares, and Triangles - that you can subscribe to watch their location as they move around a canvas. In this version of the demo, DDS Blockset provides an application that subscribes to a Square and then publishes the location of a Circle.

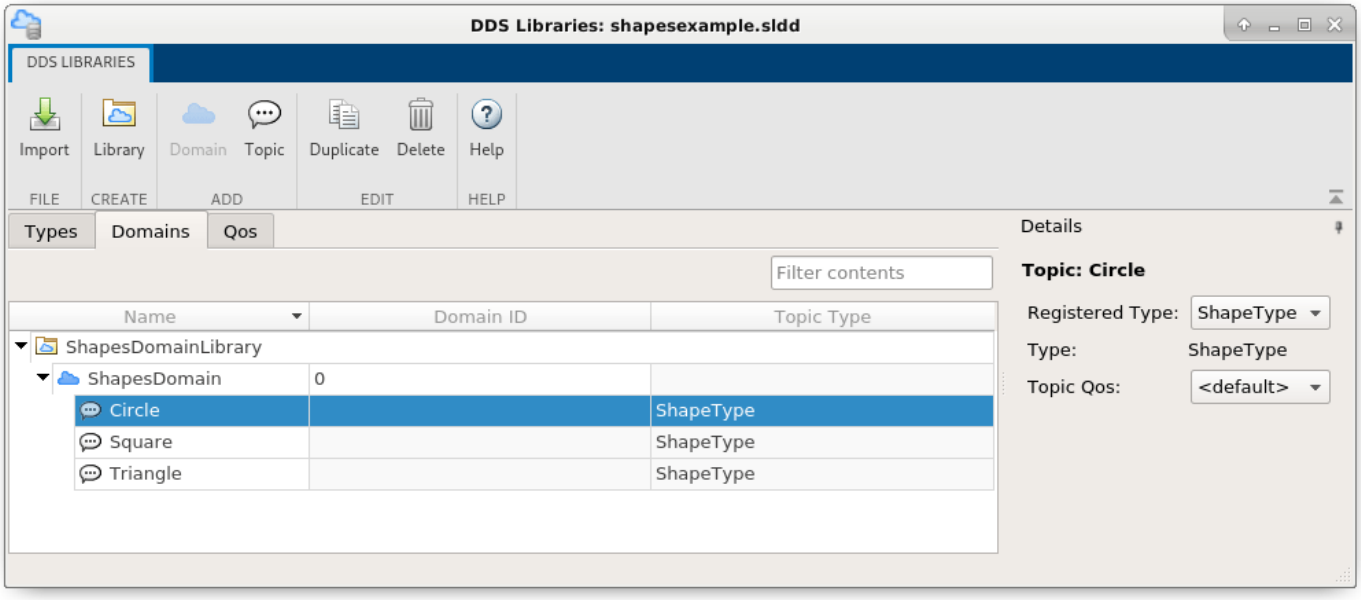
### Import DDS Definitions

To get started with the Shapes Demo, the DDS definitions of Domains, Topics, Types, and Quality of Service (QoS) have been imported from XML. To view the XML specifications, open `shapesexample.xml`.

### View and Edit Definitions

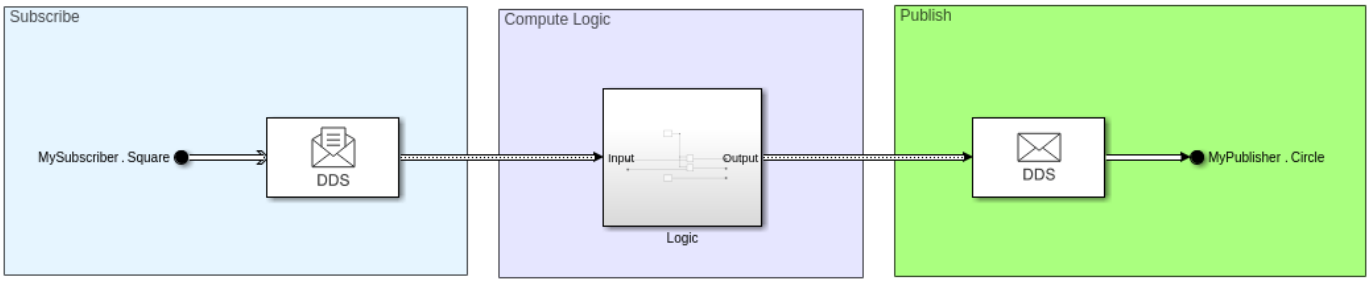
To view these DDS definitions in the DDS Dictionary, open the dictionary and view the **Types**, **Domains**, and **QoS** tabs.

```
open shapesexample.sldd
```




### View Publisher and Subscriber Model Construction

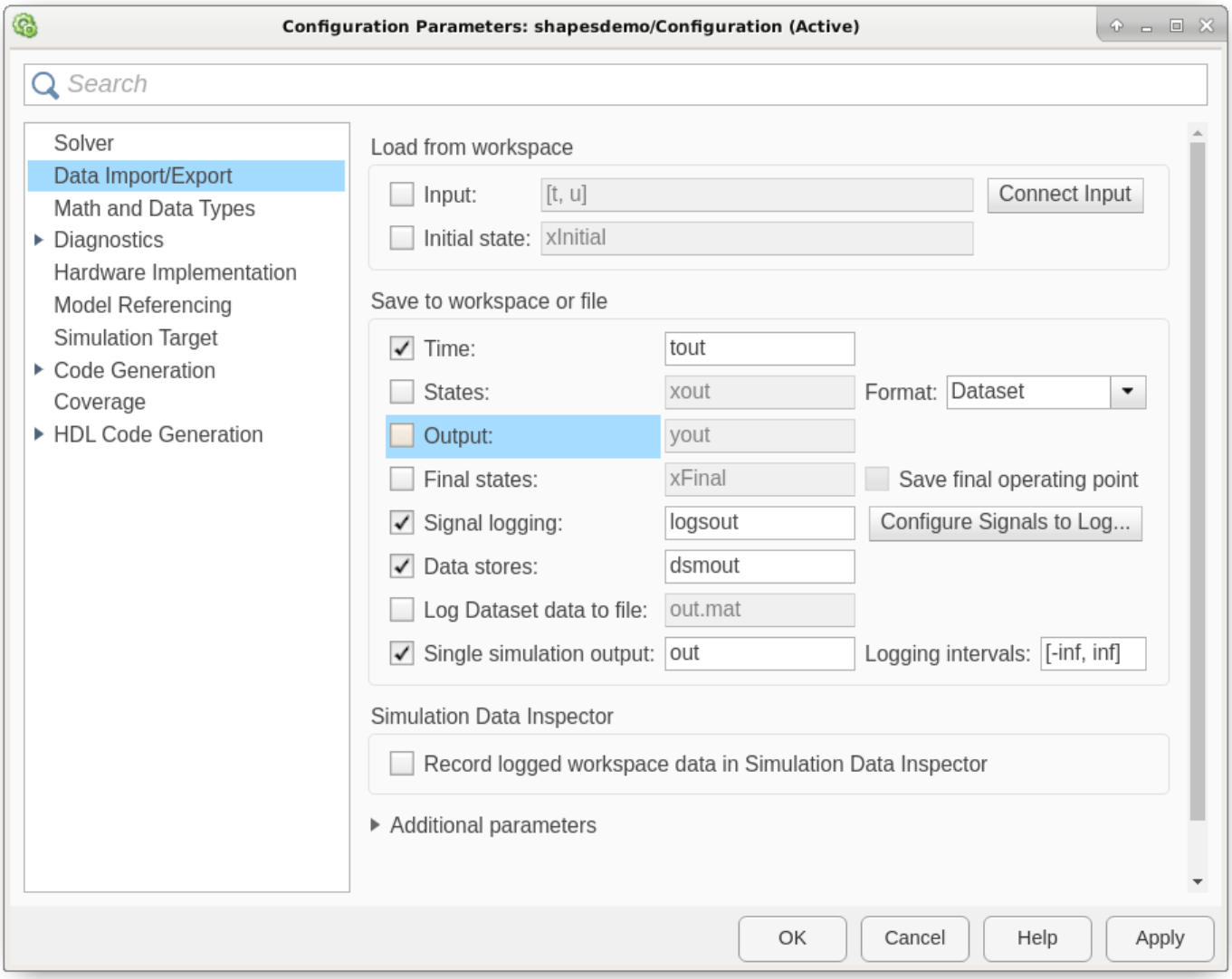
```
open_system('shapesdemo');
```




In the Simulink model for the Shapes Demo, view how the DDS blocks, Take DDS Sample and Write DDS Sample, are used to subscribe to and publish the location of the shape.

1. Click the Bus Element In block and view the DDS data type of the shape.
2. Click the Take DDS Sample block that converts the DDS data type to the Simulink data type.
3. Examine the logic component of the application model. The logic component uses the Simulink data types to compute the logic of the DDS application.
4. Click the Write DDS Sample block that converts the Simulink data types to a DDS data type.
5. Click the Bus Element Out block and view the DDS data type to send the shape position to the DDS network.

Also, note that root level output logging is unset for this model to use Bus Element Out block with virtual bus representing the DDS Publisher.




### View Configured DDS Interface

In addition to the modeling of the Shapes Demo, you can view the imported DDS interface configuration of the Topics, DataReaders, and DataWriters for the model:

1. Open the Code Mappings editor. On the toolstrip, click **Code Interface** and select **Individual Elements Code Mappings**.
2. To view the imported Topics and DataReaders, click on the **Inports** tab.
3. To view the imported Topics and DataWriters, click on the **Outports** tab.

### Build and Deploy the Shapes Demo

DDS Blockset enables out-of-the-box support for RTI and eProsima. To use this out-of-the-box support, build your application model and deploy your generated executable in your development

environment. For this example, click **Build** to subscribe to a Square and publish the location of a Circle Topic.

## See Also

## Related Examples

- “DDS Positioning System Application” on page 6-2

## External Websites

- RTI Connext Shapes Demo
- eProsima Shapes Demo

# Designing and Deploying Interoperable Applications for Heterogeneous Automated Driving Platforms

This example shows how to deploy the Automated Parking Valet application and interoperability between AUTOSAR Adaptive, DDS and ROS 2 over DDS network. This example extends the "Automated Parking Valet" (Automated Driving Toolbox) example in the ROS Toolbox and "Automated Parking Valet in Simulink" (Automated Driving Toolbox) example in the Automated Driving Toolbox™.

### Example Prerequisites

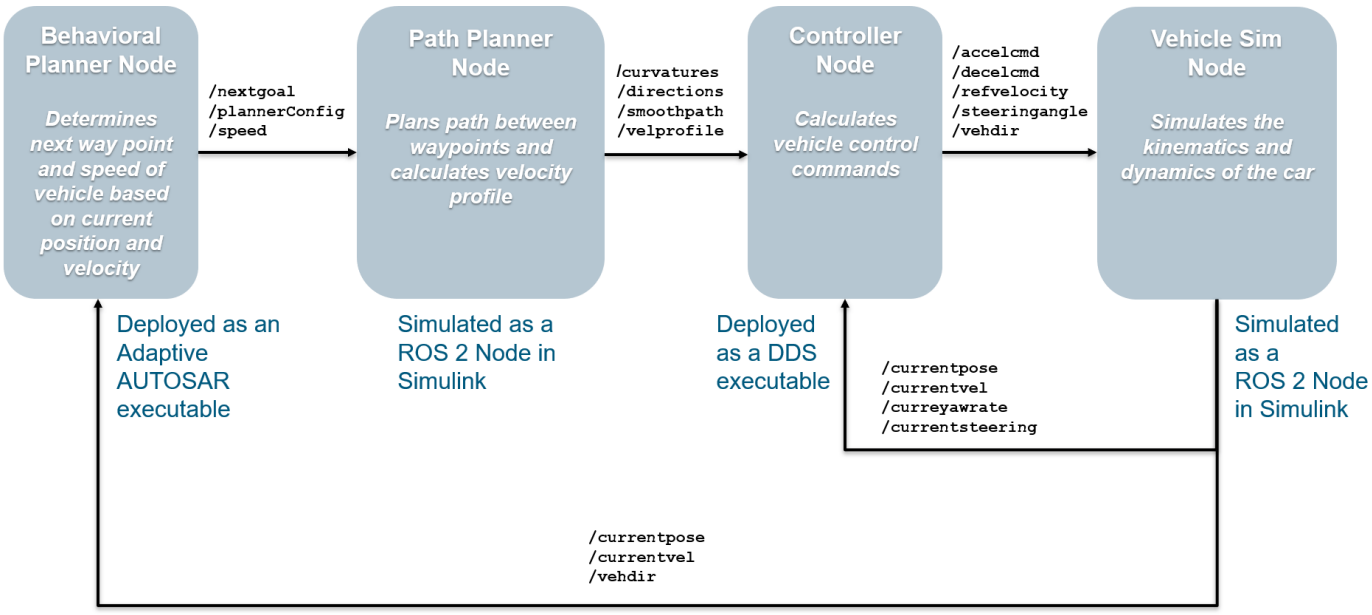
Before running this example, verify that your system meets the following requirements:

- You must have Visual Studio as your C++ compiler, installed RTI Connext® software, and the `NDDSHOME` environment variable defined. For more information, see "DDS Blockset System Requirements".
- You must have Python® installed. For more information, see "ROS Toolbox System Requirements" (ROS Toolbox).

Also, ensure you are familiar with the following examples:

- "DDS Blockset Shapes Demo"
- "Automated Parking Valet" (Automated Driving Toolbox)
- "ROS Custom Message Support" (ROS Toolbox)
- "Create and Configure AUTOSAR Adaptive Software Component" (AUTOSAR Blockset)
- "Setup Linux Target Computer" (Embedded Coder)
- "Build Simulink Model and Deploy Application" (Embedded Coder)

The example deploys the nodes as shown below.



### Register Custom Messages

Many of the existing ROS 2 messages use strings and may not be appropriate for deployment with various middleware. Since the string fields are not used in the application either, run `exampleHelperBuildCustomMessage` to register a custom message that is specific for the algorithm.

```
exampleHelperBuildCustomMessage
```

```
Identifying message files in folder 'C:/TEMP/Bdoc23a_2213998_3568/ib570499/10/tp6fc54e29/dds-ex54
Creating a Python virtual environment.Done.
Adding required Python packages to virtual environment.Done.
Copying include folders.Done.
Copying libraries.Done.
Done.
[1/1] Generating MATLAB interfaces for custom message packages... Done.
Running colcon build in folder 'C:/TEMP/Bdoc23a_2213998_3568/ib570499/10/tp6fc54e29/dds-ex546092
Build in progress. This may take several minutes...
Build succeeded.build log
Generating zip file in the folder 'C:/TEMP/Bdoc23a_2213998_3568/ib570499/10/tp6fc54e29/dds-ex546
```

Use the `ros2` command to verify that the custom message is built.

```
ros2 msg show valet/Float64ArrayFixed
```

```
#Using fixed size data in preparation for production
#Copyright 2022-2023 The MathWorks, Inc.
float64[1500] data
```

### Load Demo Data

The example loads localization data used for picking the parking spot. Use `exampleHelperDDSValetLoadLocalizationData` to load the pre-recorded localization map data. To park the car in a different spot, you can update the script.

```
exampleHelperDDSValetLoadLocalizationData;
```

### Check for NDDSHOME and Start `admin` Console

This example uses RTI Connext®. Other vendors can also be used.

Set `NDDSHOME` and open `rtiadminconsole`.

```
NDDSHome = getenv('NDDSHOME');
if isempty(NDDSHome) || ~isfolder(NDDSHome)
    error('Ensure you have ''NDDSHOME'' set to your RTI Connext 6.x folder');
end
```

You can use the `admin` console to monitor messages.

```
%system(fullfile(NDDSHome,'bin','rtiadminconsole'));
```

### Open ROS2 Models

The following models make up a ROS 2 network comprised of four ROS 2 nodes.

- The `behavioralModel` reads the current vehicle information from ROS 2 network, sends the next goal and checks if the vehicle has reached the final pose of the segment.

- The `pathPlannerModel` plans a feasible path through the environment map using a `pathPlannerRRT` (Automated Driving Toolbox) object, which implements the *optimal rapidly exploring random tree* (RRT*) algorithm and sends the plan to the controller over the ROS 2 network.
- The `controllerModel` calculates and sends the steering and velocity commands over the ROS 2 network.
- The `vehicleModel` simulates the vehicle controller effects and sends the vehicle information over the ROS 2 network.

```
behavioralModel = 'ROS2ValetBehavioralPlannerExampleNV';
pathPlannerModel = 'ROS2ValetPathPlannerExampleNV';
controllerModel = 'ROS2ValetControllerExampleNV';
controllerModelOrig = 'ROS2ValetControllerExampleNV_orig';
vehicleModel = 'ROS2ValetVehicleExampleNV';
```

### Set Up Screen Space

```
% Get screen size and base params
scrSize = get(0, 'screensize');
h = scrSize(4);
w = scrSize(3);
```

### Use Fixed Size Arrays

The original ROS 2 models are simplified to not use variants and then updated to use the fixed size arrays in preparation for deployment.

```
open_system(controllerModelOrig);
open_system(controllerModel);

set_param(controllerModelOrig,'Location', [w/2 h/2+5 w h-5]);
set_param(controllerModel,'Location', [1 h/2+5 w/2-5 h-35]);
```

The `/velprofile` topic uses fixed-size arrays. Typically this is necessary for production.

```
open_system([controllerModel,'/Subscribe/Subscribe8']);
open_system([controllerModelOrig,'/Subscribe/Subscribe8']);

close_system(controllerModelOrig)
```

### Examine DDS Version of Controller System

The `ROS2ValetBehavioralPlannerExample_DDS` example model is converted to use DDS modeling elements. To step through the process of converting the ROS 2 model to DDS, use the `exampleHelperCreateDDSModels` example helper function. To optionally step through the process, uncomment the following line and run `exampleHelperCreateDDSModels` to transform the ROS 2 model to use DDS.

```
%exampleHelperCreateDDSModels({controllerModel, behavioralModel});
```

This conversion process analyzes the ROS 2 model, retrieves the messages and topics used by the model, and uses this information to populate a DDS Dictionary. Finally, it replaces ROS modeling semantics with DDS modeling semantics. For example, Subscribe blocks from the ROS Toolbox are replaced by Message Receive blocks from the Simulink® library.

```
open_system([controllerModel '_DDS']);
set_param([controllerModel '_DDS'],'Location', [w/2 h/2+5 w h-5]);
open_system(controllerModel);

close_system(controllerModel);
set_param([controllerModel '_DDS'],'Location', [1 h/2+5 w/2-5 h-5]);
```

### Create AUTOSAR Model

A version of the AUTOSAR model is available. If you optionally want to step through the process, uncomment the following line and run exampleHelperCreateAdaptive to transform the DDS model to use Adaptive AUTOSAR.

```
%exampleHelperCreateAdaptive([behavioralModel '_DDS']);
```

### Open AUTOSAR Model

```
open_system([behavioralModel '_DDS_adaptive']);
set_param([behavioralModel '_DDS_adaptive'],'Location', [1 50 w/2-5 h/2-5]);
```

### Open Other ROS Models

```
open_system(pathPlannerModel);
set_param(pathPlannerModel,'Location', [w/2+5 50 w-5 h/2-5]);

open_system(vehicleModel);
set_param(vehicleModel,'Location', [w/2+5 h/2+5 w-5 h-35]);
```

Move models to the front to examine the system: Path Planner and Vehicle models using ROS 2, the Controller model using DDS, and the Behavioral model using Adaptive AUTOSAR.

```
open_system([controllerModel '_DDS']);
set_param([controllerModel '_DDS'],'Location', [1 h/2+5 w/2-5 h-35]);

open_system([behavioralModel '_DDS_adaptive']);
set_param([behavioralModel '_DDS_adaptive'],'Location', [1 50 w/2-5 h/2-5]);
```

### Deploy DDS and AUTOSAR Models

This step of the example is supported for host Linux® or Windows® platforms only. Before deploying the Adaptive AUTOSAR model to a Linux platform, set up the Linux target. Based on your setup, update `tgtName` to your Linux® target name.

```
tgtName = 'LinuxTarget1';
```

Get the the target handle using `TargetManager`.

```
if ~ismac
    tgs = linux.Targets;
    tg = tgs.getTarget(tgtName);
else
    tg = [];
end
```

Use the following helper functions to generate C++ code and deploy the DDS and AUTOSAR models. When you run the system, the valet system consisting of the DDS, ROS 2, and AUTOSAR models automatically parks the car.

```
if ~isempty(tg) && ~isempty(tg.getIpAddress) && tg.isConnected
    exampleHelperDeployROS2AndDDSAndAutosar([behavioralModel '_DDS_adaptive'], ...
```

```
        pathPlannerModel, ...
        [controllerModel '_DDS'], ...
        vehicleModel, tg, false); %to rebuild set to true
end
bdclose all;
```