1

Introduction

The DDS LabVIEW Integration provides users with DDS custom virtual instruments (VIs) to model DDS communication between LabVIEW and pure DDS applications.

1.1 DDS

What is DDS?

“The Data Distribution Service (DDS™) is a middleware protocol and API standard for data-centric connectivity from the Object Management Group® (OMG®). It integrates the components of a system together, providing low-latency data connectivity, extreme reliability, and a scalable architecture that business and mission-critical Internet of Things (IoT) applications need.”

“The main goal of DDS is to share the right data at the right place at the right time, even between time-decoupled publishers and consumers. DDS implements global data space by carefully replicating relevant portions of the logically shared dataspace.” DDS specification

Further Documentation

http://portals.omg.org/dds/
http://ist.adlinktech.com/
1.2 LabVIEW

What is LabVIEW?

“LabVIEW is systems engineering software for applications that require test, measurement, and control with rapid access to hardware and data insights. The LabVIEW programming environment simplifies hardware integration for engineering applications so that you have a consistent way to acquire data from NI and third-party hardware. The LabVIEW programming environment simplifies hardware integration for engineering applications so that you have a consistent way to acquire data from NI and third-party hardware. LabVIEW reduces the complexity of programming, so you can focus on your unique engineering problem. LabVIEW enables you to immediately visualize results with built-in, drag-and-drop engineering user interface creation and integrated data viewers. To turn your acquired data into real business results, you can develop algorithms for data analysis and advanced control with included math and signal processing IP or reuse your own libraries from a variety of tools. To ensure compatibility with other engineering tools, LabVIEW can interoperate with, and reuse libraries from, other software and open-source languages.”

2 Installation

This section describes the procedure to install the Vortex DDS LabVIEW Integration on a Linux or Windows platform.

2.1 System Requirements

- Operating System: Windows or Linux
- LabVIEW 2017 installed

2.2 OpenSplice (OSPL) and DDS LabVIEW Installation

Steps:

1. Install OSPL. The DDS LabVIEW Integration is included in this installer.
2. Setup OSPL license. Copy the license.lic file into the appropriate license directory.
   
   /INSTALLDIR/ADLINK/Vortex_v2/license

3. LabVIEW installation files are contained in a tools/labview folder.
   
   Example: /INSTALLDIR/ADLINK/Vortex_v2/Device/VortexOpenSplice/6.9.x/HDE/x86_64.linux/tools/labview

2.3 OpenSplice (OSPL) Configuration

By default OSPL uses single process configuration.

2.4 DDS LabVIEW Installation

2.4.1 Linux

1. Open a command shell and navigate to
   
   /INSTALLDIR/ADLINK/Vortex_v2/Device/VortexOpenSplice/6.9.x/HDE/x86_64.linux/tools/labview

2. Unzip the “adlink-dds-labview-linux-install.tar.gz”.

3. Run the install_vortex_dds_ubuntu.sh script as a super user.
   
   sudo ./install_vortex_dds_ubuntu.sh

   **NOTE:** The installer sets the default LabVIEW installation path to /usr/local/natinst/LabVIEW-2017-64. To override this installation directory, run the install script and pass the install directory as an argument:

   sudo ./install_vortex_dds_ubuntu.sh /path/to/your/LabVIEW/installation
4. LabVIEW will open and allow the installation Virtual Instrument (VI) run to completion.

5. After the installation is complete close LabVIEW. Installation takes effect the next time you start LabVIEW.

2.4.2 Windows

1. In a file browser, navigate to
   
   /INSTALLDIR/ADLINK/Vortex_v2/Device/VortexOpenSplice/6.9.x/HDE/x86_64.windows/tools/labview

2. Double click on the file “adlink_lib_vortexdds-1.0.0.1”. This will bring up the VI Package Manager installer dialog box. Select the LabVIEW version to install (32-bit or 64-bit). Select **Install**.

3. After the installation is complete close LabVIEW. Installation takes effect the next time you start LabVIEW.
2.5 Running LabVIEW

Steps:

1. Open command shell and run script to setup environment variables.

   **Linux**
   - Open a Linux terminal.
   - Navigate to directory containing release.com file.
     `/INSTALLDIR/ADLINK/Vortex_v2/Device/VortexOpenSplice/6.9.x/HDE/x86_64.linux`
   - Run release.com. (Type in “. release.com” at command line.)

   **Windows**
   - Open a command prompt.
   - Navigate to directory containing release.bat file.
     `INSTALLDIR/ADLINK/Vortex_v2/Device/VortexOpenSplice/6.9.x/HDE/x86_64.win64`
   - Run release.bat. (Type in “release.bat” at command line.)

2. Start LabVIEW using the **SAME** command shell used in Step 1.

   **NOTE:** If LabVIEW is NOT started from a command shell with the correct OSPL environment variables set, errors will occur when attempting to use DDS LabVIEW virtual instruments.
3

Vortex DDS Virtual Instruments (VIs)

The DDS LabVIEW Integration provides a function palette with custom virtual instruments (VIs) to model reading and writing data with DDS.

The Vortex DDS LabVIEW VIs are included in VortexDDS functions palette.

The following DDS VIs are provided:

- create_participant.vi
- create_publisher.vi
- create_subscriber.vi
- create_writer.vi
- create_reader.vi
- wait_historical_data.vi
- delete_entity.vi

3.1 DDS VIs usage

The typical way to model a DDS application in LabVIEW is as follows:

- model your DDS topics using IDL
- using the LabVIEW IDLPP process generate DDS Topic, Read and Write VIs from the IDL file
- add the generated VIs to your LabVIEW project
- create a DDS LabVIEW application using the VortexDDS functions palette and the generated VIs from the previous step
3.2 QoS Profiles

In DDS - “The Data-Distribution Service (DDS) relies on the usage of QoS. A QoS (Quality of Service) is a set of characteristics that controls some aspect of the behavior of the DDS Service.”

Each DDS entity VI has an associated QoS profile. By default, the OSPL default profile is used. An XML file that specifies QoS profiles can be used to set the QoS of a DDS entity.

The QoS profile of an entity is set using the `qos_uri` and `qos_profile` terminals.

Please see section QoS Provider for more information.

3.3 create_participant.vi

The create_participant VI represents a DDS domain participant entity.

In DDS - “A domain participant represents the local membership of the application in a domain. A domain is a distributed concept that links all the applications able to communicate with each other. It represents a communication plane: only the publishers and subscribers attached to the same domain may interact.”

The domain id is the OSPL default domain id specified in the OSPL configuration file (file pointed by “OSPL_URI” environment variable).

<table>
<thead>
<tr>
<th>Terminal Type</th>
<th>Optional</th>
<th>Name</th>
<th>Description</th>
<th>Output consumed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>no</td>
<td>pp</td>
<td>DDS Domain Participant entity instance</td>
<td>create_publisher.vi create_subscriber.vi RegisterTopic.vi</td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>qos_uri</td>
<td>QoS file uri</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>qos_profile</td>
<td>Name of QoS profile</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>error in (no error)</td>
<td>Input Error cluster</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>yes</td>
<td>error out</td>
<td>Error out cluster</td>
<td></td>
</tr>
</tbody>
</table>
3.4 create_publisher.vi

The create_publisher VI represents a DDS publisher entity.

In DDS, a publisher is “an object responsible for data distribution. It may publish data of different data types.”

<table>
<thead>
<tr>
<th>Terminal Type</th>
<th>Optional</th>
<th>Name</th>
<th>Description</th>
<th>Output consumed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>no</td>
<td>pp</td>
<td>DDS Domain Participant entity instance</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>qos_uri</td>
<td>QoS file uri</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>qos_profile</td>
<td>Name of QoS profile</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>error in (no error)</td>
<td>Input Error cluster</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>no</td>
<td>pub</td>
<td>DDS publisher entity instance</td>
<td>create_writer.vi</td>
</tr>
<tr>
<td>Output</td>
<td>yes</td>
<td>error out</td>
<td>Error out cluster</td>
<td></td>
</tr>
</tbody>
</table>
3.5 create_subscriber.vi

The create_subscriber VI represents a DDS subscriber entity.

In DDS, a subscriber is “an object responsible for receiving published data and making it available to the receiving application. It may receive and dispatch data of different specified types.”

<table>
<thead>
<tr>
<th>Terminal Type</th>
<th>Optional</th>
<th>Name</th>
<th>Description</th>
<th>Output consumed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>no</td>
<td>pp</td>
<td>DDS Domain Participant entity instance</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>qos_uri</td>
<td>QoS file uri</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>qos_profile</td>
<td>Name of QoS profile</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>error in (no error)</td>
<td>Input Error cluster</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>no</td>
<td>sub</td>
<td>DDS subscriber entity instance</td>
<td>create_reader.vi</td>
</tr>
<tr>
<td>Output</td>
<td>yes</td>
<td>error out</td>
<td>Error out cluster</td>
<td></td>
</tr>
</tbody>
</table>

3.6 create_writer.vi

The create_writer VI represents a DDS data writer entity.

In DDS - “The DataWriter is the object the application must use to communicate to a publisher the existence and value of data-objects of a given type.”

<table>
<thead>
<tr>
<th>Terminal Type</th>
<th>Optional</th>
<th>Name</th>
<th>Description</th>
<th>Output consumed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>no</td>
<td>pub</td>
<td>DDS publisher entity instance</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>no</td>
<td>topic</td>
<td>DDS Topic entity instance</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>qos_uri</td>
<td>QoS file uri</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>qos_profile</td>
<td>Name of QoS profile</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>error in (no error)</td>
<td>Input Error cluster</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>no</td>
<td>writer</td>
<td>DDS writer entity instance</td>
<td>write.vi</td>
</tr>
<tr>
<td>Output</td>
<td>yes</td>
<td>error out</td>
<td>Error out cluster</td>
<td></td>
</tr>
</tbody>
</table>
3.7 create_reader.vi

The create_reader VI represents a DDS data reader entity.

In DDS - “To access the received data, the application must use a typed DataReader attached to the subscriber.”

<table>
<thead>
<tr>
<th>Terminal Type</th>
<th>Optional</th>
<th>Name</th>
<th>Description</th>
<th>Output consumed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>no</td>
<td>sub</td>
<td>DDS subscriber entity instance</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>no</td>
<td>topic</td>
<td>DDS Topic entity instance</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>qos_uri</td>
<td>QoS file uri</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>qos_profile</td>
<td>Name of QoS profile</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>error in (no error)</td>
<td>Input Error cluster</td>
<td>read.vi</td>
</tr>
<tr>
<td>Output</td>
<td>no</td>
<td>reader</td>
<td>DDS reader entity instance</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>yes</td>
<td>error out</td>
<td>Error out cluster</td>
<td></td>
</tr>
</tbody>
</table>

3.8 wait_historical_data.vi

The wait_historical_data VI specifies that the Reader will wait for historical data to arrive. The timeout terminal is for setting time period (in seconds) determining how long the Reader should wait for the historical data. If the timeout is reached, then any remaining historical data may be interleaved with new data.

<table>
<thead>
<tr>
<th>Terminal Type</th>
<th>Optional</th>
<th>Name</th>
<th>Description</th>
<th>Output consumed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>no</td>
<td>reader</td>
<td>DDS Reader entity instance</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>historical_timeout</td>
<td>wait for historical data timeout (seconds)</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>error in (no error)</td>
<td>Input Error cluster</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>yes</td>
<td>error out</td>
<td>Error out cluster</td>
<td></td>
</tr>
</tbody>
</table>
3.9 delete_entity.vi

The delete_entity VI is used to delete a DDS entity. Connect the DDS participant to the entity terminal to delete the participant (pp) in a LabVIEW DDS application.

NOTE: If the user application VI stops due to an error and does not run to completion, the participant entity is not deleted and leaks occur. The participants are deleted once the user closes LabVIEW.

<table>
<thead>
<tr>
<th>Terminal Type</th>
<th>Optional</th>
<th>Name</th>
<th>Description</th>
<th>Output consumed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>no</td>
<td>entity</td>
<td>DDS entity instance</td>
<td></td>
</tr>
</tbody>
</table>
4

LabVIEW Generation from IDL

While creating a DDS application in LabVIEW, the user must create Topic VIs which map to DDS topic types. In addition to registering a topic, the user needs to create DDS Read and DDS Write VIs. LabVIEW data is represented in clusters. The DDS Read and Write VIs have terminals that require a LabVIEW cluster. On data writes, the LabVIEW clusters are converted to DDS topic types and on data reads, the DDS topic types are converted to LabVIEW clusters.

The user can generate the LabVIEW clusters and DDS VIs from an IDL file.

The DDS LabVIEW Integration supports generation of LabVIEW typedefs and VIs from IDL. This chapter describes the details of the IDL-LabVIEW binding.

4.1 Generating LabVIEW VIs from an IDL File IDLPP

DDS Topic Types can be described in an IDL file. The LabVIEW IDL generation is done using the Import_IDL Tools menu in LabVIEW.

Tools/VortexDDS/Import_IDL
In the IDL file, ensure that any topic structures have the OSPL specific annotation `pragma keylist` defined. This value is added as a `keylist constant` to the `key` terminal of the generated `RegisterTopic.vi`.

Select the IDL file and a folder for the generated VIs.

Upon successful generation the VIs are located in the folder that was chosen. A dialog box appears indicating the path of the generated VIs and the IDL file.
4.2 Generated Artifacts

The following table defines the LabVIEW artifacts generated from IDL concepts:

<table>
<thead>
<tr>
<th>IDL Concept</th>
<th>LabVIEW Concept</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>module</td>
<td>Appended to the name of each VI contained in the module</td>
<td></td>
</tr>
<tr>
<td>enum</td>
<td>enum</td>
<td>a LabVIEW .ctl file.</td>
</tr>
<tr>
<td>enum value</td>
<td>enum value</td>
<td>a LabVIEW .ctl file.</td>
</tr>
<tr>
<td>struct</td>
<td>cluster</td>
<td>a LabVIEW .ctl file.</td>
</tr>
<tr>
<td>field</td>
<td>cluster field</td>
<td></td>
</tr>
<tr>
<td>sequence</td>
<td>array</td>
<td></td>
</tr>
<tr>
<td>array</td>
<td>array</td>
<td></td>
</tr>
</tbody>
</table>

Datatype mappings

The following table shows the LabVIEW equivalents to IDL primitive types:

<table>
<thead>
<tr>
<th>DDS IDL</th>
<th>LabVIEW Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>char</td>
<td>int8</td>
</tr>
<tr>
<td>octet</td>
<td>uint8</td>
</tr>
<tr>
<td>short</td>
<td>int16</td>
</tr>
<tr>
<td>unsigned short</td>
<td>uint16</td>
</tr>
<tr>
<td>long</td>
<td>int32</td>
</tr>
<tr>
<td>unsigned long</td>
<td>uint32</td>
</tr>
<tr>
<td>long long</td>
<td>int64</td>
</tr>
<tr>
<td>unsigned long long</td>
<td>uint64</td>
</tr>
<tr>
<td>float</td>
<td>single-precision floating point</td>
</tr>
<tr>
<td>double</td>
<td>double-precision floating point</td>
</tr>
<tr>
<td>string</td>
<td>String</td>
</tr>
</tbody>
</table>

Unsupported DDS data types

- wchar not supported
- wstring not supported
- any not supported
- long double not supported
- union not supported
- inheritance not supported

Generated VIs and controls

For each struct in the IDL file, the following VIs and controls are generated:

- RegisterTopic.vi
- Write.vi
- Read.vi
- CicoTable.vi (Copy-in copy-out)
- Topic cluster.ctl (corresponds to each struct in IDL File)
- Enum.ctl (corresponds to each enum in IDL File)

The “moduleName_structName” is appended to the name of each VI and control that is generated.
4.3 RegisterTopic.vi

The RegisterTopic.vi represents a DDS topic type. The DDS topic corresponds to a single data type. In DDS, data is distributed by publishing and subscribing topic data samples.

<table>
<thead>
<tr>
<th>Terminal Type</th>
<th>Optional</th>
<th>Name</th>
<th>Description</th>
<th>Output consumed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>no</td>
<td>DomainParticipantHandle</td>
<td>DDS Domain Participant entity instance</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>no</td>
<td>TopicName</td>
<td>DDS Topic Name</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>QosProfilePath</td>
<td>QoS file uri</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>QoSProfileName</td>
<td>Name of QoS profile</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>error in (no error)</td>
<td>Input Error cluster</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>no</td>
<td>topicHandle</td>
<td>DDS Topic entity instance</td>
<td>create_reader.vi</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>create_writer.vi</td>
</tr>
<tr>
<td>Output</td>
<td>yes</td>
<td>error out</td>
<td>Error out cluster</td>
<td></td>
</tr>
</tbody>
</table>
4.4 Read.vi

The DDS Read.vi is used to read DDS samples from a specific topic.

<table>
<thead>
<tr>
<th>Terminal Type</th>
<th>Optional</th>
<th>Name</th>
<th>Description</th>
<th>Output consumed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>no</td>
<td>readerHandle</td>
<td>DDS Reader entity instance</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>mask</td>
<td>read_condition masks LabVIEW cluster</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>max_samples</td>
<td>maximum number of samples to read</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>read_operation</td>
<td>READ or TAKE default operation is TAKE</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>query_expression</td>
<td>expression to filter samples based on a query</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>query_parameters</td>
<td>parameters for the query expression</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>wait_available_data_timeout</td>
<td>wait for data available timeout (seconds)</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>error in (no error)</td>
<td>Input Error cluster</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>yes</td>
<td>no_of_samples</td>
<td>Number of samples read</td>
<td>user</td>
</tr>
<tr>
<td>Output</td>
<td>no</td>
<td>samples</td>
<td>LabVIEW cluster</td>
<td>user</td>
</tr>
<tr>
<td>Output</td>
<td>yes</td>
<td>dds_sample_info_array</td>
<td>sample information</td>
<td>user</td>
</tr>
<tr>
<td>Output</td>
<td>yes</td>
<td>error out</td>
<td>Error out cluster</td>
<td></td>
</tr>
</tbody>
</table>
4.4.1 Filters

The filtering of incoming samples can happen based on a query and/or on a sample read condition(s).

**Query**

query_expression: The expression is a SQL condition.

query_parameters: Each parameter element must be an array element.

*Note: Query expressions are only validated at runtime. If they are incorrect, errors will occur while running the VI.*

**Read Condition**

The read condition mask specified will filter the samples that are read or take(n).

Example: For a reader, the Sample State has **Not Read** selected and **Read** deselected.

Only samples with a Sample State **Not Read** will be processed with read or take. Any samples with the **Read** sample state will not be read or take(n).
4.5 Write.vi

The DDS Write.vi is used to write DDS samples to a specific topic.

<table>
<thead>
<tr>
<th>Terminal Type</th>
<th>Optional</th>
<th>Name</th>
<th>Description</th>
<th>Output consumed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>no</td>
<td>writerHandle</td>
<td>DDS Writer entity instance</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>write_op</td>
<td>write operation: WRITE, DISPOSE, WRITE_DISPOSE</td>
<td>DDS</td>
</tr>
<tr>
<td>Input</td>
<td>no</td>
<td>data</td>
<td>samples LabVIEW cluster</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>wait_pub_matched_timeout</td>
<td>wait for publication matched timeout (seconds)</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>write_after_timeout</td>
<td>write samples after timeout</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>yes</td>
<td>error in (no error)</td>
<td>Input Error cluster</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>yes</td>
<td>error out</td>
<td>Error out cluster</td>
<td></td>
</tr>
</tbody>
</table>
Each Vortex DDS virtual instrument (VI) has a QoS file uri terminal and a QoS profile terminal. These terminals are used to set the QoS profile. By default, the OSPL default profile is used. In DDS - The Data-Distribution Service (DDS) relies on the usage of QoS. A QoS (Quality of Service) is a set of characteristics that controls some aspect of the behavior of the DDS Service.

Each DDS VI has an associated QoS profile. By default, the OSPL default profile is used. An XML file that specifies QoS profiles can be used to set the QoS of a DDS block.

The following section explains how the QoS is set for a DDS entity using the QoS Provider.

### 5.1 QoS Provider File

Quality of Service for DDS entities is set using XML files based on the XML schema file DDS_QoSProfile.xsd. These XML files contain one or more QoS profiles for DDS entities.

**Note:** Sample QoS Profile XML files can be found in the LabVIEW DDS examples directories.

### 5.2 QoS Profile

A QoS profile consists of a name. The file contains QoS elements for one or more DDS entities. A skeleton file without any QoS values is displayed below to show the structure of the file.

```xml
     xsi:schemaLocation="file:DDS_QoSProfile.xsd">
  <qos_profile name="DDS QoS Profile Name">
    <datareader_qos></datareader_qos>
    <datawriter_qos></datawriter_qos>
    <domainparticipant_qos></domainparticipant_qos>
    <subscriber_qos></subscriber_qos>
    <publisher_qos></publisher_qos>
    <topic_qos></topic_qos>
  </qos_profile>
</dds>
```

**Example: Specify Publisher Partition**

The example below specifies the publisher’s partitions as A and B.

```xml
<publisher_qos>
  <partition>
    <name>
      <element>A</element>
    </name>
    <name>
      <element>B</element>
    </name>
  </partition>
</publisher_qos>
```
The QoS Profiles from the XML file can be obtained using the `List_qos_profiles` Tools menu in LabVIEW.

Tools/VortexDDS/List_qos_profiles
Steps to set the QoS Profile

1. A QoS Provider file can be selected by browsing to the XML file from the **List_qos_profiles** dialog box. Once a valid QoS file is chosen the **Available qos profiles** table is populated with the list of qos profiles that are available in the QoS XML file. If there are QoS profiles found in the file, then **Copy to Clipboard** button will be enabled.

2. Select the QoS Profile that you want to use and click on **Copy to Clipboard**.

![List_qos_profiles](image)

3. In your VI, create a String constant and press Ctrl + V. Connect this String constant to the DDS VI **qos_profile** terminal. Set the **qos_uri** as a LabVIEW control or constant and navigate to the path of the QoS Provider file.

![DDS PersistentQosProfile](image)

The **qos_profile** and **qos_uri** are optional terminals. If they are not set then the default QoS settings will be used.

**Note:** Seeing the QoS Profile in the list only guarantees the QoS Profile exists in the file. It does not mean the qos tag exists for the entity. The user is responsible for verifying the entity qos tag exists in the file.
Demo iShapes Example

A simple demo iShapes example is provided to demonstrate the basic capabilities of the LabVIEW DDS integration. It displays DDS communication between LabVIEW and pure DDS applications.

The demo_ishapes.vi example (LabVIEW application) can be found using the NI Example Finder in LabVIEW:

![NI Example Finder](image)

The demo_ishapes.exe (DDS application) can be found in the following directory.

```
OSPL_HOME/bin/demo_ishapes.exe
```
6.1 Example Files

An explanation of what each example file does is provided below.

**DDS_Data Folder**
This folder contains the idl file and artifacts generated from idlpp process.

**ishape.idl**
- Defines the ShapeType in idl
- Used to generate the LabVIEW DDS VIs via idlpp

**ishape.idl.xml**
- Defines the topic descriptor from idl file

**ShapeType_CicoTable.vi**
- Defines the copy-in, copy-out table information for mapping IDL to LabVIEW types

**ShapeType.ctl**
- Defines a ShapeType cluster in LabVIEW; generated from idlpp
- The ShapeType represents a DDS topic type
- ShapeType specifies 4 properties: color, x, y, shapesize

**ShapeType_Read.vi**
- DDS Read ShapeType samples

**ShapeType_Write.vi**
- DDS Write ShapeType samples

**ShapeType_Topic.vi**
- DDS Register ShapeType topic

**DDS_PersistentQoS_All.xml**
- XML file that specifies the DDS QoS (quality of service) settings for RegisterTopic

**DDS_VolatileQoS_All.xml**
- XML file that specifies the DDS QoS (quality of service) settings for Reader and Writer entities

**demo_ishapes.vi**
- Creates a participant on the default DDS domain
- Registers a ShapeType Topic to Read and to Write to one of the three topics: Circle, Square or Triangle
• Subscribes to the shape and color from demo_isshapes.exe DDS application
• As soon as they match, demo_isshapes.vi publishes to the DDS application and follows the subscribed shape
6.2 Steps to run example

Steps:

1. Open command shell and run script to setup environment variables.

   **Linux**
   - Open a Linux terminal.
   - Navigate to directory containing release.com file.
     
     `/INSTALLDIR/ADLINK/Vortex_v2/Device/VortexOpenSplice/6.9.x/HDE/x86_64.linux`
   - Run release.com. (Type in “. release.com” at command line.)

   **Windows**
   - Open a command prompt.
   - Navigate to directory containing release.bat file.
     
     `INSTALLDIR/ADLINK/Vortex_v2/Device/VortexOpenSplice/6.9.x/HDE/x86_64.win64`
   - Run release.bat. (Type in “release.bat” at command line.)

2. Navigate to the directory that contains demo_ishapes.exe DDS application and run the application using the command shell used in Step 1.

   **Linux**
   - Run demo_ishapes.exe (Type in “./demo_ishapes.exe &” at command line)

   **Windows**
   - Run demo_ishapes.exe (Type in “demo_ishapes.exe &” at command line)

3. Start LabVIEW using the **SAME** command shell used in Step 1. Open the demo_ishapes.vi using the **NI Example Finder** in LabVIEW.

   **NOTE:** If LabVIEW and the demo_ishapes.exe application are NOT started from a command shell with the correct OSPL environment variables set, the example will not work.
4. In the LabVIEW demo_ishapes.vi application make the following selections:
   Read Shape: Circle Track color: GREEN
   Write Shape: Triangle Write color: MAGENTA
5. In the demo_ishapes.exe DDS application make the following selections:
   Shape: Circle Color: Green Click **Publish**
   Shape: Triangle Click **Subscribe**

6. Run the LabVIEW demo_ishapes.vi application

7. To stop the LabVIEW application, click on **STOP** on the front panel of the demo_ishapes.vi.

### 6.3 Output

LabVIEW application publishes samples to the DDS application. In the demo_ishapes.exe application, the Triangle follows the Circle.
7

Contacts & Notices

7.1 Contacts

ADLINK Technology Corporation
400 TradeCenter
Suite 5900
Woburn, MA
01801
USA
Tel: +1 781 569 5819

ADLINK Technology Limited
The Edge
5th Avenue
Team Valley
Gateshead
NE11 0XA
UK
Tel: +44 (0)191 497 9900

ADLINK Technology SARL
28 rue Jean Rostand
91400 Orsay
France
Tel: +33 (1) 69 015354

Web: http://ist.adlinktech.com/
Contact: http://ist.adlinktech.com
E-mail: ist_info@adlinktech.com
LinkedIn: https://www.linkedin.com/company/79111/
Twitter: https://twitter.com/ADLINKTech_usa
Facebook: https://www.facebook.com/ADLINKTECH

7.2 Notices

Copyright © 2018 ADLINK Technology Limited. All rights reserved.