Open Data Distribution Service on Linux

Disclaimer:	OpenDDS and other dependent modules (ACE/TAO, Xerces) are OSS modules provided as-is. NVIDIA does not make any security or safety warranties. DDS included in DRIVE OS is not for production and may have secure vulnerabilities that NVIDIA is not addressing. It is not intended for the Safety use case.
Note:	OpenDDS, the dependent libraries, and sample applications are built using c++14. OpenDDS is not available in Safety Build.
Note:	Starting in 5.2.3, OpenDDS is installed as an option through SDKM; there is no need to compile for the prebuilt binaries. OpenDDS is not intended for production usage.

Data Distribution Service (DDS) is networking middleware for data exchanges using the publish-subscribe pattern for real time distributed applications. DDS ensures interoperability (across different vendors), portability of applications, and high performance.

DDS enables publisher and subscriber nodes to:

- Send and receive messages
- Send and receive events and commands based on topics

Additionally, DDS handles:

- Addressing
- Marshaling and unmarshaling data
- Internal flow control
- Discovery of services

Applications can specify the Quality of Service (QoS) for discovery and runtime behavior.

OpenDDS is an open source, C++ implementation of the OMG Data Distribution Service specification. OpenDDS is built on the ACE abstraction layer. DDS for DRIVE OS includes:

- OpenDDS and the dependent libraries and sample applications.
- Sample applications leverage DDS-based communication methods.

Applications that use DDS for communication services must:

- Include minimal design for the publish-subscribe model of DDS.
- Define the exchange data types using IDL.
- Identify the QoS needs.
- Invoke DDS portable APIs that are independent of DDS implementations.

For the supported OpenDDS versions, see the *Release Notes*.

Within OpenDDS, the following terminology is used and defined as follows:

Cell Heading	Cell Heading
Domain	Represents a global data space. Each domain is uniquely identified by an integer domain ID. Domains are independent from each other. For two DDS applications to communicate with each other, they must join the same domain.
Domain Participant	A domain participant is the entry point for an application to interact within a particular domain. The domain participant is a factory for many of the objects involved in writing or reading data.
Topic	A topic is the most basic description of data to be published or subscribed to. Each topic describes a data stream in your system. A topic is identified by its name, which is a string that must be unique in the whole domain.
Publisher	The publisher is responsible for taking the published data and disseminating it to all relevant subscribers in the domain.
Subscriber	The subscriber receives the data from the published data and disseminates it to all relevant subscribers in the domain.
Data Writer	Data writer objects are responsible for sending type-specific data to one or more data readers. A data writer is created with a topic, which gives a name to the data stream and associates the data writer with a data type.
Data Reader	Data reader objects are responsible for receiving type-specific data sent by a data writer. Data writers and readers are associated with a topic.
QoS Policies	The entities of a domain have their own set of Quality of Service policies that determine the behavior of the transfer of data and the compatibility between data writers and readers. The entities include: • Domain participant • Topic • Publisher • Subscriber • Data writer • Data reader
Sample	A sample is a single data update received over DDS.
Interface Definition Language (IDL)	The Interface definition language is used to specify the interface between the client and the server so that the Remote Procedure Call (RPC) mechanism can create the code stubs required to call functions across the network.

Installing OpenDDS

The OpenDDS source and binary files are included as part of the DRIVE OS release package.

Use either the SDK Manager or manually extract in sequence the SDK RUN files.

For the branch and build number, see the *Release Notes*.

```
```

Where:

- `<top>` is the directory where you installed DRIVE OS software package.
- `drive-oss-src/dds` contains the sources for DDS, TAO, and sample code.
- `drive-oss-src/dds/install_static` contains the static libraries for DDS and TAO.
- `sample code.drive-t186ref-linux/targetfs/home/nvidia/drive-t186ref-linux/samples/dds` contains the pre-built samples using dynamic libraries.
- `drive-t186ref-linux/targetfs/usr/lib` contains the libraries on the target system. These libraries are flashed at `/usr/lib`.

Using Sample Applications

The complex IDL structures application transmits ten messages from the publisher to the subscriber. The data object transferred is a sample object detection metadata used in computer vision/imaging.

The pre-compiled binaries are available at:

```
<top>/drive-t186ref-linux/home/nvidia/drive-t186ref-linux/samples/dds
```

Running the complex_idl_example Test App

The steps below apply for cross process mode. Create two different sessions for starting the publisher and subscriber so that you can see the subscriber receiving and printing data.

The default configuration file uses RTPS and TCP. For more information, see Discovery Model and Transport Model.

To run the complex_idl_example application

1. Set the environment for both sessions:

```
export DDS_ROOT=/home/nvidia/drive-t186ref-linux/samples/dds
export PATH=$DDS_ROOT:$PATH
```

2. Start the publisher and subscriber in different sessions:
   - Publisher session:

```
compidl_publisher -DCPSConfigFile generic_config.ini
```

   - Subscriber session:

```
compidl_subscriber -DCPSConfigFile generic_config.ini
```

The expected output contains:

- Four lines with vertices of rectangles
- Two transformation matrices are printed 10 times with different values in the subscriber session

Sample output is as follows:

```
Object Detection:
EnableBoundingBoxClipping = 1
EnableFuseObjects = 1
MaxNumImages = 10
 ROIs:
 h:      = 200 1100
 w:      = 300 2100
 x:      = 400 3100
 y:      = 500 4100
 Transformations:
```

```
  | 225 325 425 |
  | 525 625 725 |
  | 825 925 1025 |
  | 250 350 450 |
  | 550 650 750 |
  | 850 950 1050 |
```

Complex IDL with Static Discovery

Static discovery occurs when predefined endpoints with a predefined IP and port location are specified in the configuration file.

The complex IDL example remains the same, with the same expected output. However, since there are code changes required and a different configuration file is used, different binaries for the publisher and subscriber are needed. The environment setup and compile steps are the same, but the way the applications are launched is as follows:

- Publisher session:

```
static_publisher -DCPSConfigFile static_discovery.ini
```

- Subscriber session:

```
static_subscriber -DCPSConfigFile static_discovery.ini
```

Note:	Static discovery supports rtps_udp as the mode of transport. Start the subscriber first, before the publisher.

There is a known issue with the static discovery application. The data transfers successfully, but there is a 60 second timeout and the following message is displayed:

```
ERROR: Subscriber_static.cpp:146: main() - wait failed!
(1663031|1) WARNING: DataLink[101f4100]::~DataLink() - link still in use by 1
entities when deleted!
(1663031|1) ERROR: SubscriberImpl::~SubscriberImpl, 1 datareaders still exist.
```

Use Cases

Use cases have been classified into three types:

- Intra-SoC or single VM
- Inter-VM
- Inter-SoC

OpenDDS supports three types of discoveries and five types of transports, which are changeable in the `generic_config.ini` file.

For more information, see Discovery Model and Transport Model.

The default configuration file uses RTPS discovery and TCP transport. There are other configuration files in the `<top>/drive-t186ref-linux/targetfs/home/nvidia/drive-t186ref-linux/samples/dds` folder. These can be used instead of the current `generic_config.ini`. Note that the same config file must be used for the publisher and the subscriber.

Single VM/Intra-SoC Use Cases

The following table lists out the configuration files in the package for single VM use cases, and what transport mode and discovery mode they use.

Configuration File	Discovery Mode	Transport Mode
generic_config.ini (default)	RTPS	tcp
static_discovery.ini	Static	rtps_udp
shmem.ini	RTPS	shmem
rtps_multicast.ini	RTPS	rtps_udp

Inter-VM and Inter-SoC Use Cases

Static Discovery in Inter-VM/Inter-SoC

DDS can be used to transfer data from one VM to another in a multi-virtual machine environment. The configuration of these use cases is similar to intra-VM use cases, except the two different sessions for publisher and subscriber belong to different VMs or SoCs.

The transport and discovery mode limitation:

- RTPS discovery only works if multicast support is enabled.

The configuration file `rtps_multicast.ini` uses RTPS discovery and `rtps_upd` transport. Modify the following line according to the interface where the Virtual Machine must connect to other Virtual Machines.

```
MulticastInterface=hvX
```

- Where X=0,1,2, depending on the VM's bridge interface to the other VM.

For inter-VM communication, the bridge interface `hv0-hv1` is used between Linux and QNX.

To enable multicast support for RTPS discovery, you must modify the `tegra_t186ref_gnu_linux_defconfig` file (extracted from `oss_src.run`) and change this line:

```
CONFIG_IP_MULTICAST=y
```

Now recompile the kernel. For the steps, see Building the Flashing Kernel. If you do not recompile, the DDS application fails to launch with error: "unable to join multicast group".

For an RTPS based discovery mechanism, participants discover each other using Simple Participant Discovery Protocol (SPDP), which is based on multicast-UDP transport. The `rtps_multicast.ini` configuration file has an `InteropMulticastOverride` field to override the default multicast address `239.255.0.1`.

Ensure the kernel IP routing table has an entry against the specified multicast address or is handled using the default gateway. If the multicast address is NOT provided, the DDS middleware reports the following error while sending SDPDP related messages:

```
no route to host
```

The following provides an example for adding an entry in the kernel IP routing table for inter-VM:

```
#route add 239.255.0.0 netmask 255.255.0.0 hv1
```

To verify:

```
#route -n
```

The endpoint discovery can be triggered without multicast, where the following changes can be made in the configuration file:

```
SedpMulticast=0
SedpLocalAddress=<Local ip:port>
```

Use the `complex_idl_example` test application. Follow the steps depending on which VM the publisher and subscriber are started.

For the inter-SoC use case, use the `rtps_multicast.ini` configuration file and follow the steps for inter-VM communication. Change the `MulticastInterface` according to the interfaces that connect.

In all inter-VM/inter-SoC use cases, the transport mode must be specified with a local address. The local address must be an interface that can be pinged from the outside world, or at least from where the subscriber runs.

```
[transport/tcp1]
transport_type=tcp
local_address=IP:Port
## Substitute this with IP and Port for VM/SoC where publisher/subscriber is run
```

Static Discovery in Inter-VM/Inter-SoC

Use the `static_discovery.ini` configuration file for static discovery use cases. Recompiling the kernel is NOT required for static discovery. However, `static_publisher` and `static_subscriber` binaries must be used. Change the IP addresses as per the Virtual Machines or SoCs. Make the changes as follows:

```
[transport/rudp]                 ## Reader Transport
transport_type=rtps_udp
use_multicast=0
local_address=IP:Port
## Substitute this with IP and Port for VM/Tegra where subscriber is run

[transport/rtudp]               ## Writer Transport
transport_type=rtps_udp
use_multicast=0
local_address=IP:Port
## Substitute this with IP and Port for VM/Tegra where publisher is run
```

For inter-VM communication, the bridge interface hv0-hv1 is used between Linux and QNX.

- Linux fixed IP address is 192.168.2.2
- QNX fixed IP address is 192.168.2.1

For inter-SoC communication, external IPs are used instead.

Manually Compiling the Libraries

Compiling the libraries is required if any non-default configure features are used, if the toolchain is modified, or if any changes are made to the source or application.

To manually compile the libraries

1. Initiate a build with these commands:

```
cd <top>/drive-oss-src/dds/samples/complex_idl_example/build_scripts;
 ./build.sh clean
(Build the dynamic libraries)
 ./build.sh
(Build the standalone libraries)
 ./build.sh static
```

2. Once the build completes successfully, run the `copy_samples` script to copy the libraries to the correct directories.

```
cd <top>/drive-oss-src/samples/; ./copy_samples.sh
```

3. Proceed to Flashing the SoC.

Data Types/IDL

DDS applications send and receive messages that are strongly-typed. These types are defined by the application developer in Interface Definition Language (IDL). For example:

```
struct Data {
  long message_counter;
  string stock_name;
  double price;
};
```

Since OpenDDS is capable of exchanging messages between processes created in different languages, a structure must be defined that can be translated to all supported languages. For best practices, structures are defined in separate files with the extension `.idl`. These IDL files are written in C++ style, but do not have the same syntax. Since the file is created once, mismatch errors are avoided between publisher and subscriber defined data types. The IDL file is then compiled with OpenDDS tools to create source files for both publishers and subscribers.

The OpenDDS IDL compiler is invoked using the `opendds_idl` executable. This host side compilation requires the environment to have DDS_ROOT set as `<top>/drive-oss-src/dds/install/share/dds`. The IDL compiler:

- Parses a single IDL file.
- Generates the serialization and key support code that OpenDDS requires to marshal and de-marshal the types.
- Generates the type support code for the data readers and writers.

For each IDL file processed, such as `xyz.idl`, three files are generated:

- xyzTypeSupport.idl
- xyzTypeSupportImpl.h
- xyzTypeSupportImpl.cpp

Typically, `opendds_idl` is passed several options and the IDL file name as a parameter.

```
opendds_idl xyz.idl
```

For more details on options of `opendds_idl`, see the OpenDDS Developer Guide as provided in References.

The `xyz.idl` and `xyzTypeSupport.idl` are compiled by the TAO IDL compiler using the `tao_idl` executable. These client stubs are generated:

```
xyzC.h
 xyzC.inl
 xyzC.cpp
xyzTypeSupportC.h
xyzTypeSupportC.inl
xyzTypeSupportC.cpp
```

These server skeletons are generated:

```
xyzS.h
xyzS.cpp
xyzTypeSupportS.h
xyzTypeSupportS.cpp
```

- Pure client applications require `#include` and link with client stubs.
- Pure server applications require `#include` and link with server skeletons.

```
tao_idl xyz.idl
tao_idl -I<top>/drive-oss-src/dds/install xyzTypeSupport.idl
```

For more details, see the TAO IDL User's Guide as provided in References.

Discovery Model

OpenDDS provides three options for discovery:

- **Information Repository**: A centralized repository style that runs as a separate process allowing publishers and subscribers to discover one another centrally.
- **RTPS Discovery**: A peer-to-peer style of discovery that uses the RTPS protocol to advertise availability and location information.
- **Static Discovery**: A way of discovering which topic and endpoints, as well as QOS policies of all entities, are defined in the configuration file.

Transport Model

Transport Model

Transport Model

Transport mechanisms in OpenDDS are pluggable because they can be replaced by another mechanism through changes in a configuration `.ini` file. The transport framework is extensible to implement customized transports.

OpenDDS transport implementations include:

- tcp
- udp
- multicast
- shmem
- rtps_udp

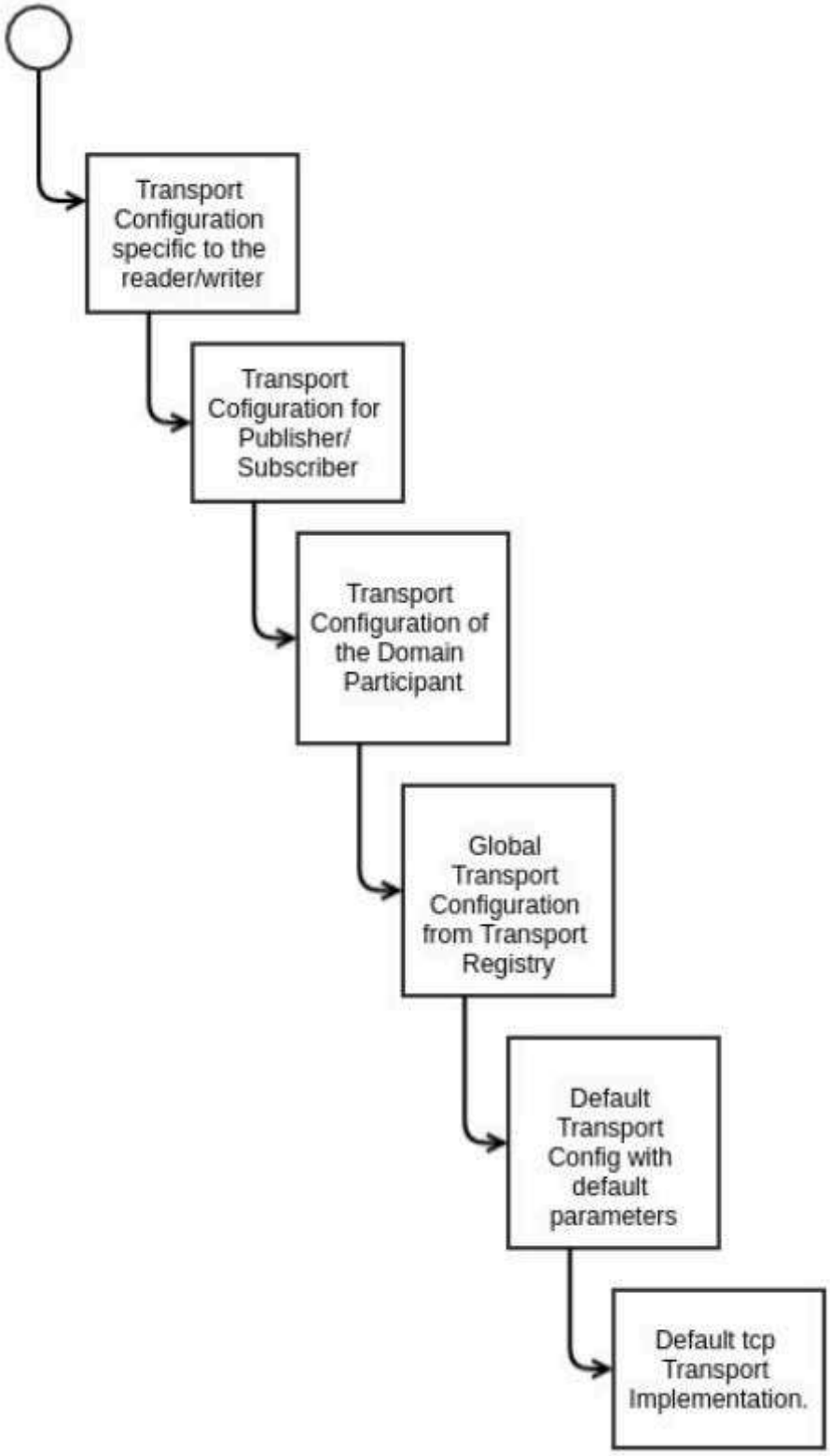
Transport configurations can be specified for:

- Domain participants
- Publishers
- Subscribers
- Data writers
- Data readers

Each transport configuration consists of transport instances, which are pre-configured transport implementations for a channel. The transport configuration and instances are managed in the transport registry, which is created using programming APIs or configuration files.

Transport Selection Hierarchy

Each data writer or reader follows a specific hierarchy in selecting a transport.



Programming Guidelines

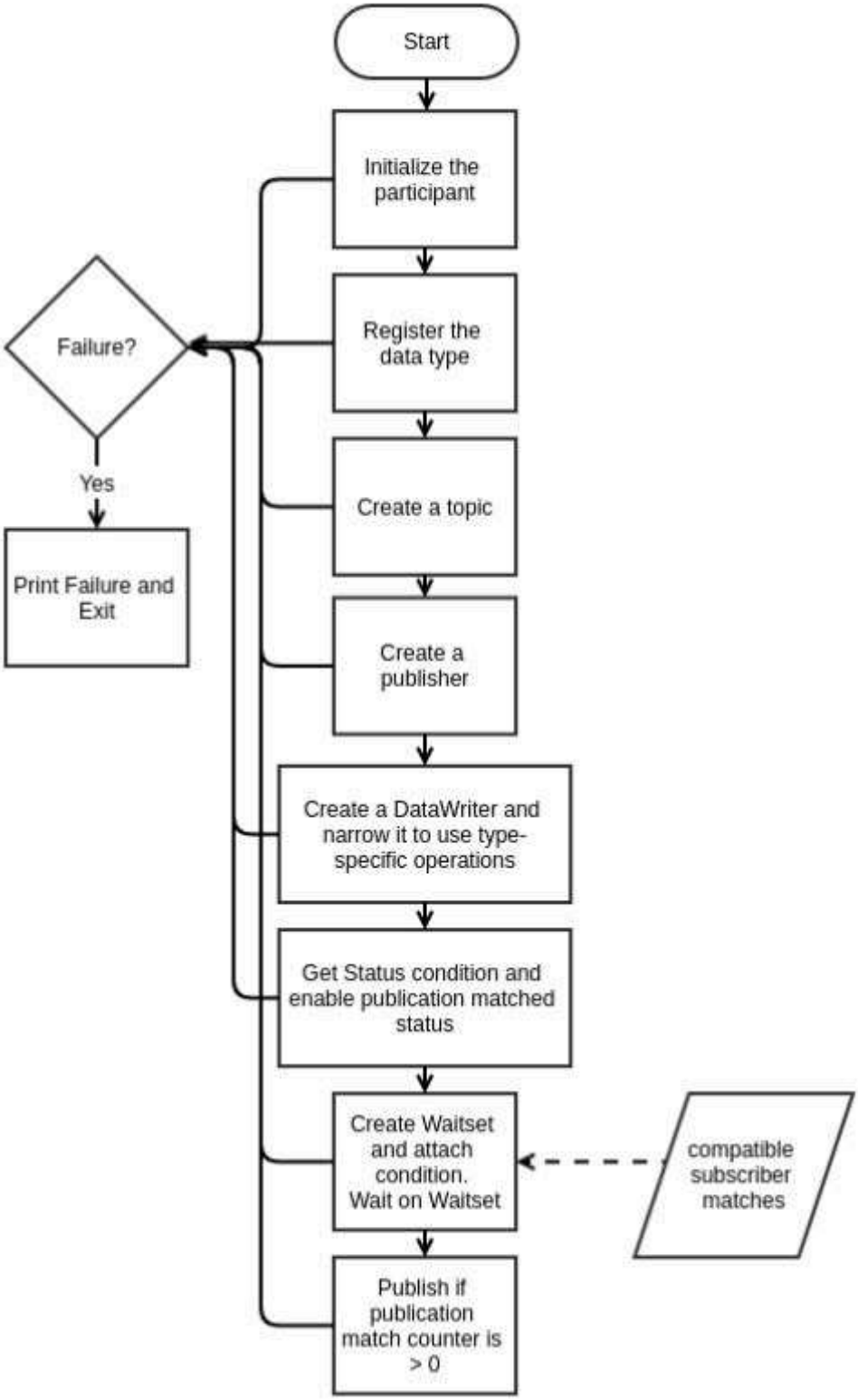
NVIDIA recommends that you utilize the source from drive-oss-src to configure OpenDDS for generating the libraries based on the build requirements. Any application written on top of the default libraries is expected to use the same flags as the ones used in corresponding library builds.

Writing an IDL File

Follow these guidelines for writing and compiling an IDL in the IDL Data Types section.

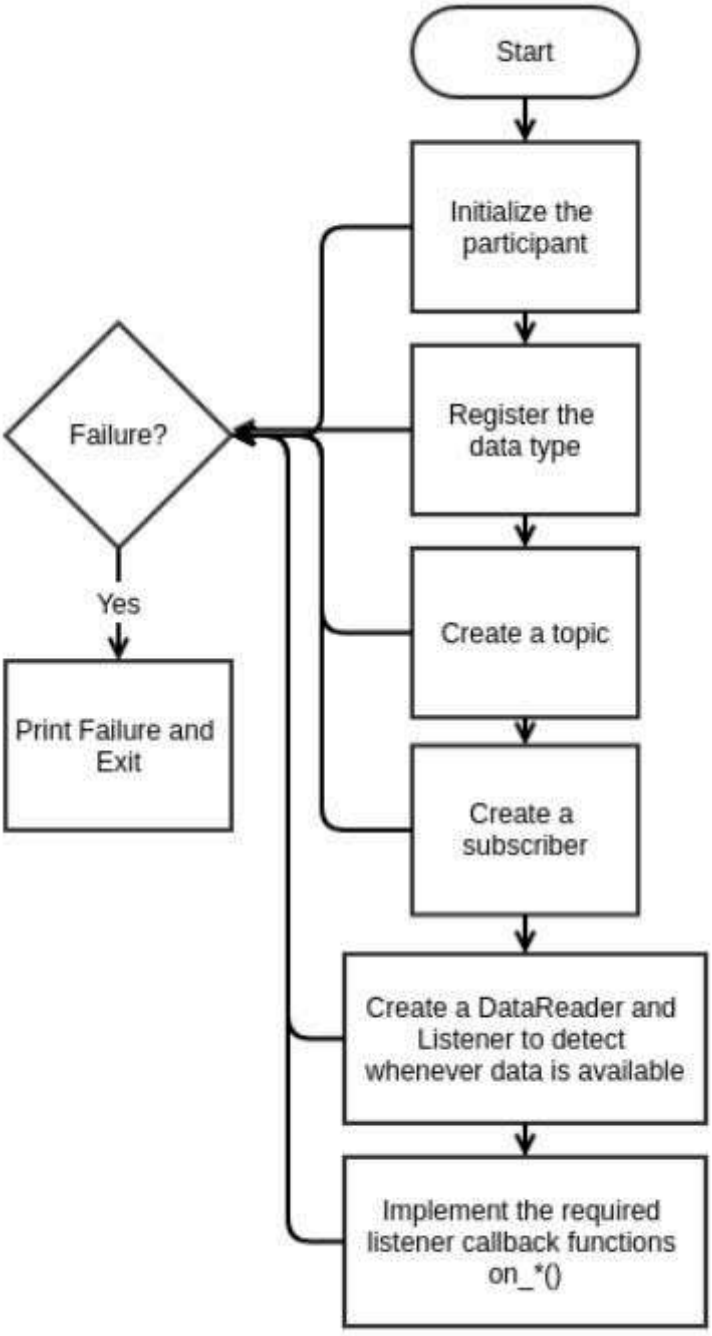
Writing a Publisher

For best practices, follow these guidelines on writing a publisher. Using these guidelines avoids the use of conditions and wait sets for a free-running publisher.



For more information about writing a publisher, see the OpenDDS Developer's Guide as provided in References.

Writing a Subscriber



For more information about writing a subscriber, see the OpenDDS Developer's Guide as provided in References.

Note:	The libraries packaged with DDS are generated with the OPENDDS_SECURITY flag enabled. All user applications are expected to be built with these flags enabled (whether the security plugin features are exercised or not), or you must generate the libraries without these flags, using their configure command on the packaged source.

Quality of Service for DDS Entities

OpenDDS QoS Policies

There are several Quality of Service policies available as a part of OpenDDS. Each specify a defined structure. Different sets of policies are applicable to:

- Domain participant
- Topic

- Publisher
- Subscriber
- Data writer
- Data reader

If the QoS is changed, existing associations are removed if they are no longer compatible and new associations are added if they become compatible.

OpenDDS QoS Policies

The OpenDDS QoS policies are as follows:

Cell Heading	Cell Heading
LIVELINESS	Determines whether participants are alive and reachable. Implemented using heartbeat or direction assertion.
RELIABILITY	Best effort vs. reliable. • Reliable specifies the service attempts to deliver all samples in its history. • Best effort indicates it is acceptable to not retry propagation of any samples.
HISTORY	Determines the number of samples to retain until the consumer acquires them.
DURABILITY	Determines if the data writers must maintain samples after they have been sent to subscribers.
DURABILITY SERVICE	Controls the deletion of samples in the TRANSIENT or PERSISTENT durability cache and provides a way to specify the history and resource limit for the sample cache.
RESOURCE LIMITS	Determines the amount of resources the service can consume to meet the requested QoS.
PARTITION	Creates a logical partition within a domain using a string name.
DEADLINE	Allows the application to detect when data is not written or read within a specified amount of time.
LIFESPAN	Allows the application to specify when a sample expires. Expired samples are NOT delivered to subscribers.
USER DATA	Can be set to any sequence that can be used to attach information to the created entity, such as security credentials for authentication.
TOPIC DATA	Can be set to attach additional information to the created topic.
GROUP DATA	Can be used to implement matching mechanisms similar to those of the PARTITION policy, except the decision is based on an application-defined policy.
TRANSPORT PRIORITY	Considered a hint to the transport layer to indicate at what priority to send messages. Higher values indicate a higher priority.
LATENCY BUDGET	Considered a hint to the transport layer to indicate the urgency of samples being sent and is used only for monitoring purposes at this time.
ENTITY FACTORY	Controls whether entities are automatically enabled when they are created.
PRESENTATION	Controls how changes to instances by publishers are presented to data readers. It affects the relative ordering of these changes and the scope of this ordering.

DESTINATION ORDER	Controls the order in which samples within a given instance are made available to a data reader.
WRITER DATA LIFECYCLE	Controls the lifecycle of data instances managed by a data writer.
READER DATA LIFECYCLE	Controls the lifecycle of data instances managed by a data reader.
TIME BASED FILTER	Controls how often a data reader may be interested in changes to values in a data instance.
OWNERSHIP	Determines whether more than one data writer can write samples for the same data-object instance.
OWNERSHIP STRENGTH	Used in conjunction with the OWNERSHIP policy, when the OWNERSHIP kind is set to EXCLUSIVE. The data writer with the highest value of strength is considered the owner of the data-object instance.

Recommended Policies for Use-cases

Recommended Policies for Use-cases

- Streaming Data
  - Deadline
  - Reliability
  - Time based filter
  - Durability
  - History
  - Transport priority
- Alarms/Events
  - Reliability
  - Durability
  - Liveliness
  - History
- Large Data
  - Reliability
  - Liveliness
  - Transport priority

Guidelines on integration with other Build Systems

OpenDDS does not support IDL to C mapping, but C++ mappings are supported. The OpenDDS header files expose RTTI functionality directly. If an application built by the NVIDIA build system includes these header files, it can result in build errors.

To overcome build error problems

- Use a C wrapper or a C++ shim layer over the publisher/subscriber application.
- Use OpenDDS build flags while compiling the application.

The C / C++ wrapper can then be built using the regular `tmake` build.

Migrating to Other DDS Implementation

The following sections describe specification compatibility and provide interoperability notes.

Specification Compatibility

OpenDDS states compliance with these specifications:

- Version 1.4 of OMG Data Distribution Service (DDS) for Real-Time Systems specification (formal/2015-04-10)
- Version 2.2 of the Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification (DDSI-RTPS) (formal/2014-09-01).

OpenDDS does NOT implement these specifications:

- DDS Security 1.1b (ptc/17-09-20)
- Extensible and Dynamic Topic Types for DDS Specification Version 1.2 (formal/17-08-01)
- Remote Procedure Calls over DDS Version 1.0 (formal/17-04-01)

Other DDS implementations may include these specifications, and cater to wider use cases than OpenDDS.

Interoperability between OpenDDS and commercially available DDS implementations is not affected, provided that RTPS interoperability protocol is used for discovery.

Design Effort for Migration to Different DDS Implementation

- **Type Representation**: OpenDDS supports OMG IDL spec version 3.1 from the CORBA specification. The latest IDL specification version is 4.2. Other DDS implementations may support a higher IDL version. In addition, other DDS implementations support other languages to represent types as specified in the DDS-XTYPES spec mentioned above, such as XML, XDR, etc., providing flexibility to represent DDS topic type support in a more readable form.

  If migration of type representation format is desired, the effort is proportional to the complexity and number of IDL files migrated.
- **Transport and Discovery Mechanisms**: Commercial distributions support a superset of transport mechanisms that include mandated transports and discovery methods of DDS. Consequently, applications using the recommended transport and discovery methods must work as-is. If a custom transport mechanism is developed, re-implementation may be required. The `DCPSInfoRepodiscovery` mechanism is specific to OpenDDS and must not be used if eventual migration or interoperability is desired.
- **Choosing QoS Policies**: Since QoS policies are specification driven, no change is expected to support these in other DDS implementations. Additional QoS policies may exist. For example, `TypeConsistencyEnforcementQosPolicy`, `DataRepresentationQoSPolicy`, etc.

Coding Effort

- **IDL Compiler**: OpenDDS supports C++ and Java bindings for data objects, while commercial versions may support more languages. Moreover, commercial versions may also have code generators that write boilerplate code for publisher and subscriber and its build environment, over and above compiling the IDL file. The generated publisher and subscriber code might look very different from what NVIDIA writes for OpenDDS, because the class definitions differ. Since the DDS specification platform independent model defines the APIs including the DDS entities, QoS policies, listeners, etc., those remain the same.
- **Publisher/Subscriber Code**: Commercial versions may have a boilerplate publisher, subscriber already written, and extension to that code for your application, if necessary. DDS users may, however, choose not to use the boilerplate code and port your application manually, wherein code refactoring might be required.

  For example, transport/discovery parameters for OpenDDS can be given via the config file, which may look very different than other distributions—or might not exist—and parameters are passed via the code itself.

Interoperability Issues

Due to a different interpretation of DDS specification by commercial DDS implementations, there may be some interoperability issues. For example, interpretation of CDR. Interoperability demonstrations have been successfully organized between commercial DDS implementations and OpenDDS using a common application, so migration and interoperability is possible with minimal tuning.

Minimal design change and coding adaptation is required to migrate to a commercial version due to a well-defined portable DDS specification.

References

- DDS Specification Version 1.4
- RTPS Specification Version 2.2
- DDS Security Specification Version 1.1 Beta 1
- OpenDDS Developer's Guide Version 3.13
- OpenDDS API Guide
- TAO IDL Compiler Users Guide

OpenDDS Third-Party Licenses

This topic provides license information about the third-party software libraries included in this NVIDIA product.

OpenDDS License

OpenDDS (Licensed Product) is protected by copyright, and is distributed under the following terms.

OpenDDS is an open source implementation of the Object Management Group (OMG) Data Distribution Service (DDS), developed and copyrighted by Object Computing Incorporated (OCI). OpenDDS is both a multi-language and multi-platform implementation. The OMG DDS specification is intended to be suitable for systems whose requirements include real-time, high volume, robustness, failure tolerant data distribution utilizing a publish and subscribe model.

Since OpenDDS is open source and free of licensing fees, you are free to use, modify, and distribute the source code, as long as you include this copyright statement.

In particular, you can use OpenDDS to build proprietary software and are under no obligation to redistribute any of your source code that is built using OpenDDS. Note however, that you may not do anything to the OpenDDS code, such as copyrighting it yourself or claiming authorship of the OpenDDS code, that will prevent OpenDDS from being distributed freely using an open source development model.

Warranty

LICENSED PRODUCT IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE WARRANTIES OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE.

Support

LICENSED PRODUCT IS PROVIDED WITH NO SUPPORT AND WITHOUT ANY OBLIGATION ON THE PART OF OCI OR ANY OF ITS SUBSIDIARIES OR AFFILIATES TO ASSIST IN ITS USE, CORRECTION, MODIFICATION OR ENHANCEMENT.

Support may be available from OCI to users who have agreed to a support contract.

Liability

OCI OR ANY OF ITS SUBSIDIARIES OR AFFILIATES SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY LICENSED PRODUCT OR ANY PART THEREOF.

IN NO EVENT WILL OCI OR ANY OF ITS SUBSIDIARIES OR AFFILIATES BE LIABLE FOR ANY LOST REVENUE OR PROFITS OR OTHER SPECIAL, INDIRECT AND CONSEQUENTIAL DAMAGES, EVEN IF OCI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

OpenDDS copyright OCI. St. Louis MO USA, 2005

ACE and TAO License

ACE(TM), TAO(TM), CIAO(TM), DAnCE>(TM), and CoSMIC(TM) (henceforth referred to as "DOC software") are copyrighted by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University, Copyright (c) 1993-2018, all rights reserved. Since DOC software is open-source, freely available software, you are free to use, modify, copy, and distribute--perpetually and irrevocably--the DOC software source code and object code produced from the source, as well as copy and distribute modified versions of this software. You must, however, include this copyright statement along with any code built using DOC software that you release. No copyright statement needs to be provided if you just ship binary executables of your software products.

You can use DOC software in commercial and/or binary software releases and are under no obligation to redistribute any of your source code that is built using DOC software. Note, however, that you may not misappropriate the DOC software code, such as copyrighting it yourself or claiming authorship of the DOC software code, in a way that will prevent DOC software from being distributed freely using an open-source development model. You needn't inform anyone that you're using DOC software in your software, though we encourage you to let us know so we can promote your project in the DOC software success stories.

The ACE, TAO, CIAO, DAnCE, and CoSMIC web sites are maintained by the DOC Group at the Institute for Software Integrated Systems (ISIS) and the Center for Distributed Object Computing of Washington University, St. Louis for the development of open-source software as part of the open-source software community. Submissions are provided by the submitter ``as is'' with no warranties whatsoever, including any warranty of merchantability, noninfringement of third party intellectual property, or fitness for any particular purpose. In no event shall the submitter be liable for any direct, indirect, special, exemplary, punitive, or consequential damages, including without limitation, lost profits, even if advised of the possibility of such damages. Likewise, DOC software is provided as is with no warranties of any kind, including the warranties of design, merchantability, and fitness for a particular purpose, noninfringement, or arising from a course of dealing, usage or trade practice. Washington University, UC Irvine, Vanderbilt University, their employees, and students shall have no liability with respect to the infringement of copyrights, trade secrets or any patents by DOC software or any part thereof. Moreover, in no event will Washington University, UC Irvine, or Vanderbilt University, their employees, or students be liable for any lost revenue or profits or other special, indirect and consequential damages.

DOC software is provided with no support and without any obligation on the part of Washington University, UC Irvine, Vanderbilt University, their employees, or students to assist in its use, correction, modification, or enhancement. A number of companies around the world provide commercial support for DOC software, however. DOC software is Y2K-compliant, as long as the underlying OS platform is Y2K-compliant. Likewise, DOC software is compliant with the new US daylight savings rule passed by Congress as "The Energy Policy Act of 2005," which established new daylight savings times (DST) rules for the United States that expand DST as of March 2007. Since DOC software obtains time/date and calendaring information from operating systems users will not be affected by the new DST rules as long as they upgrade their operating systems accordingly.

The names ACE(TM), TAO(TM), CIAO(TM), DAnCE(TM), CoSMIC(TM), Washington University, UC Irvine, and Vanderbilt University, may not be used to endorse or promote products or services derived from this source without express written permission from Washington University, UC Irvine, or Vanderbilt University. This license grants no permission to call products or services derived from this source ACE(TM), TAO(TM), CIAO(TM), DAnCE(TM), or CoSMIC(TM), nor does it grant permission for the name Washington University, UC Irvine, or Vanderbilt University to appear in their names.

If you have any suggestions, additions, comments, or questions, please let me know.

Douglas C. Schmidt

Xerces3 License

communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or

documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this