OpenDDS FOUNDATION™

WEBINAR

Getting Started as an OpenDDS Code Contributor

Webinar Host: Jen Wiese
Panelists: Fred Hornsey, Son Dinh, Adam Mitz, Justin Wilson, Tim Simpson

June 16, 2022

OpenDDS Foundation™
12140 Woodcrest Exec. Dr., Ste. 300
Saint Louis, MO 63141 USA

Limits of Liability and Disclaimer of Warranty

While every precaution has been taken in preparing this material, including research, development and testing, OpenDDS Foundation assumes no responsibility for errors or omissions. No liability is assumed by OpenDDS Foundation for any damages resulting from the use of this information.

Upcoming Events

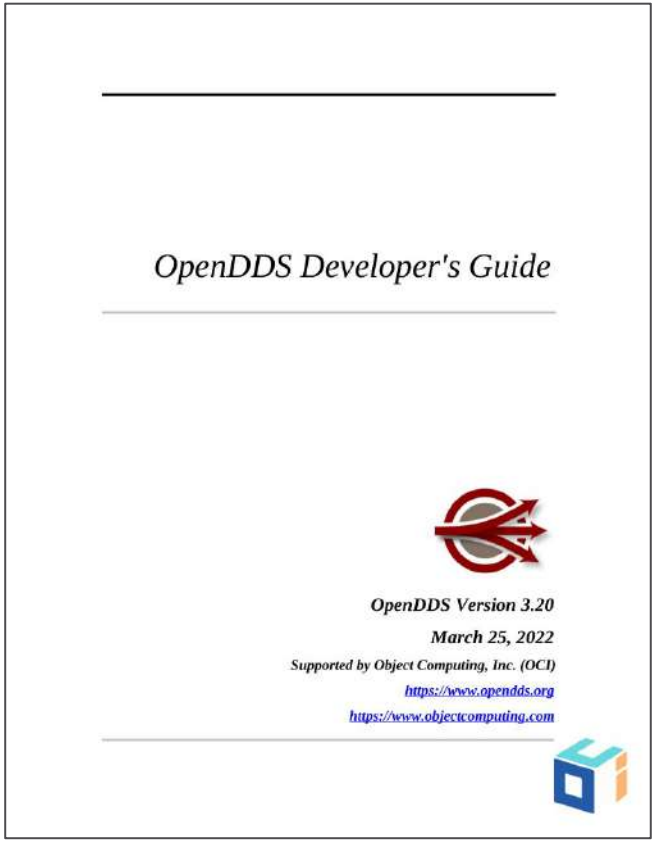
Online Training Classes:

- July 11-12
  - Introduction to OpenDDS Programming (C++, Java)
- July 13-14
  - Building OpenDDS Applications with DDS Security (C++, Java)
- July 18-19
  - OpenDDS Essentials I (C++, Java)
    - QoS, Keys and Instances, & Built-In Topics
- July 20-21
  - OpenDDS Essentials II (C++, Java)
    - Configuration, Listeners, Conditions, & Content-Subscription

https://objectcomputing.com/opendds-training

Agenda

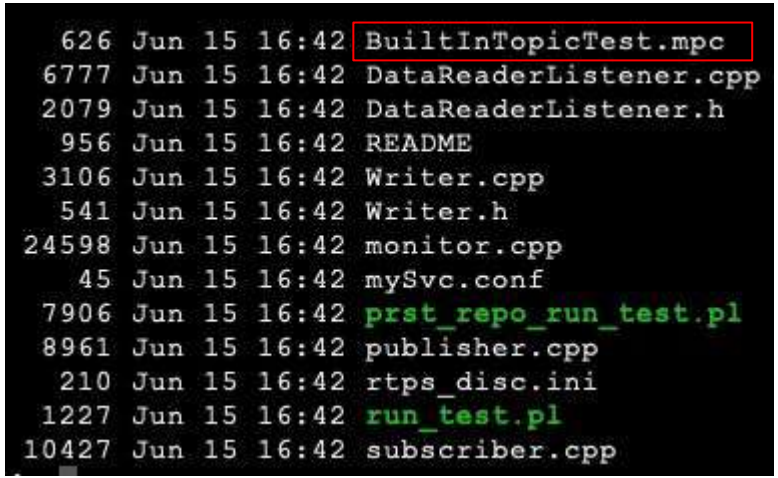
- Welcome and Introductions
- Development Environment Setup
- Guided Tour of Resources
- Testing
- Contributing via a Pull Request
- Contributing Documentation
- Contributing to Related Repos
- Q&A

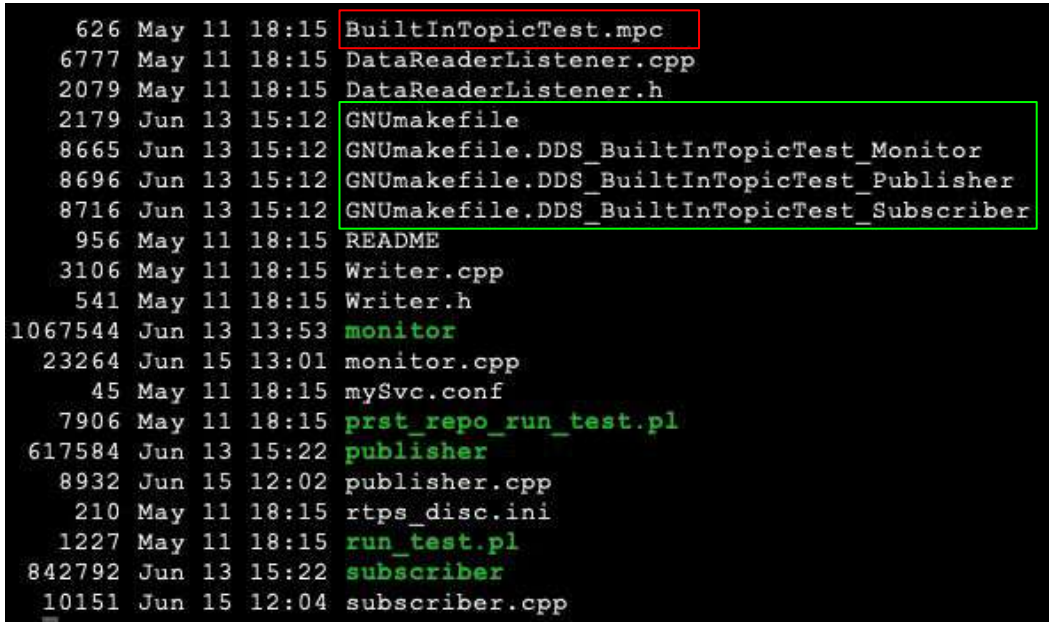


Development Environment Setup - MPC Overview

- The Makefile, Project and Workspace Creator (MPC) is used to generate input for specific build tools, e.g., Makefiles for GNU make or solution files for Visual Studio.
- Input to MPC is a file (mwc or mpc file) that describes the project being generated. E.g., whether the output is an executable or a shared library, the include paths, its dependencies, etc.
- MPC reads a common input file (mwc or mpc file) and generates the actual input files for a specific build tool being used on the target platform
  - The MPC input file is written once and works with different build tools/platforms
- GitHub repo: https://github.com/DOCGroup/MPC

Development Environment Setup - MPC Example





Development Environment Setup - Configure Script

- OpenDDS provides a Perl script, named `configure`, to configure its features, download and configure dependencies, generate build tool inputs.
  - Dependencies such as ACE, TAO, MPC can be downloaded or paths to existing dependencies can be specified.
  - Turn on/off different features: IPv6, security, output static lib, build tests, etc.
- It uses MPC internally to generate build tool input files, e.g., GNU makefiles, Visual Studio .sln files, etc.
- Run `configure --help` to see the available options.
- Build tool input files are generated and OpenDDS is ready to build
  - E.g., run `make` on Linux to start building.
- `setenv.sh` (on Linux) or `setenv.cmd` (on Windows) is generated which contains all environment variables needed to build OpenDDS.
  - E.g., on Linux, run `./setenv.sh` before running `make`.

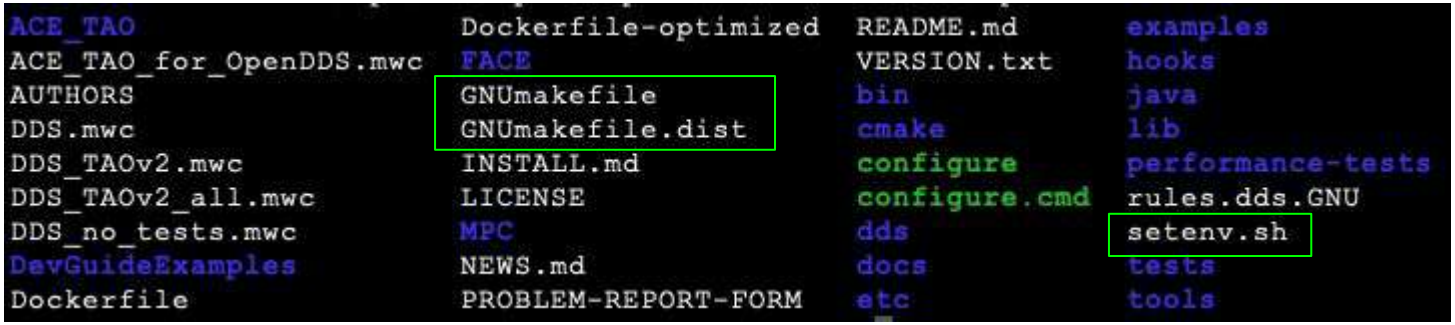
Development Environment Setup - Configure Example

- Example 1: `./configure --ace-github-latest --ipv6`
  - Download ACE and TAO from their GitHub repository and configure them. Also download MPC.
  - Turn on IPv6 feature.
- Example 2: `./configure --ace=/path/to/ACE --tao=/path/to/TAO --mpc=/path/to/MPC --security --tests`
  - Specify paths to existing ACE, TAO, MPC directories and use them.
  - Turn on security feature.
  - Build all tests and examples.
- Second run of `configure` may not work as expected because some feature config file may not be overwritten
  - Create a different clone and run `configure` in there
  - Cleanup all generated files, e.g., `git clean -xdf`, and run `configure` again

Development Environment Setup - Linux/macOS

- Clone OpenDDS:

  `git clone https://github.com/objectcomputing/OpenDDS.git`
- Run `configure` script with the desired configuration



- Run `setenv.sh` to export the required environment variables
  - Not needed if only running make from top-level directory
- Run `make`
  - E.g., `make -j4`

Development Environment Setup - Windows

- Open a Command Prompt for Visual Studio (x86 or x64)
- Clone OpenDDS:

  `git clone https://github.com/objectcomputing/OpenDDS.git`
- Run configure script with the desired configuration
  - E.g., `configure --ace-github-latest --tests --security --ipv6`
- Run setenv.cmd to export the required environment variables
  - Not needed if running from the same command prompt that ran configure
- Open the generated solution file, e.g., DDS_TAOv2.sln, from this command prompt
- From the Solution Explorer tab of the opened Visual Studio, right click and choose Build Solution

Development Environment Setup - Other Platforms

- OpenDDS supports other platforms, including Android, iOS, Raspberry Pi.
- For more information:

  https://github.com/objectcomputing/OpenDDS/blob/master/INSTALL.md#cross-compiling

  https://opendds.org/quickstart/

Guided Tour of Resources - opendds.org/documents

Documentation

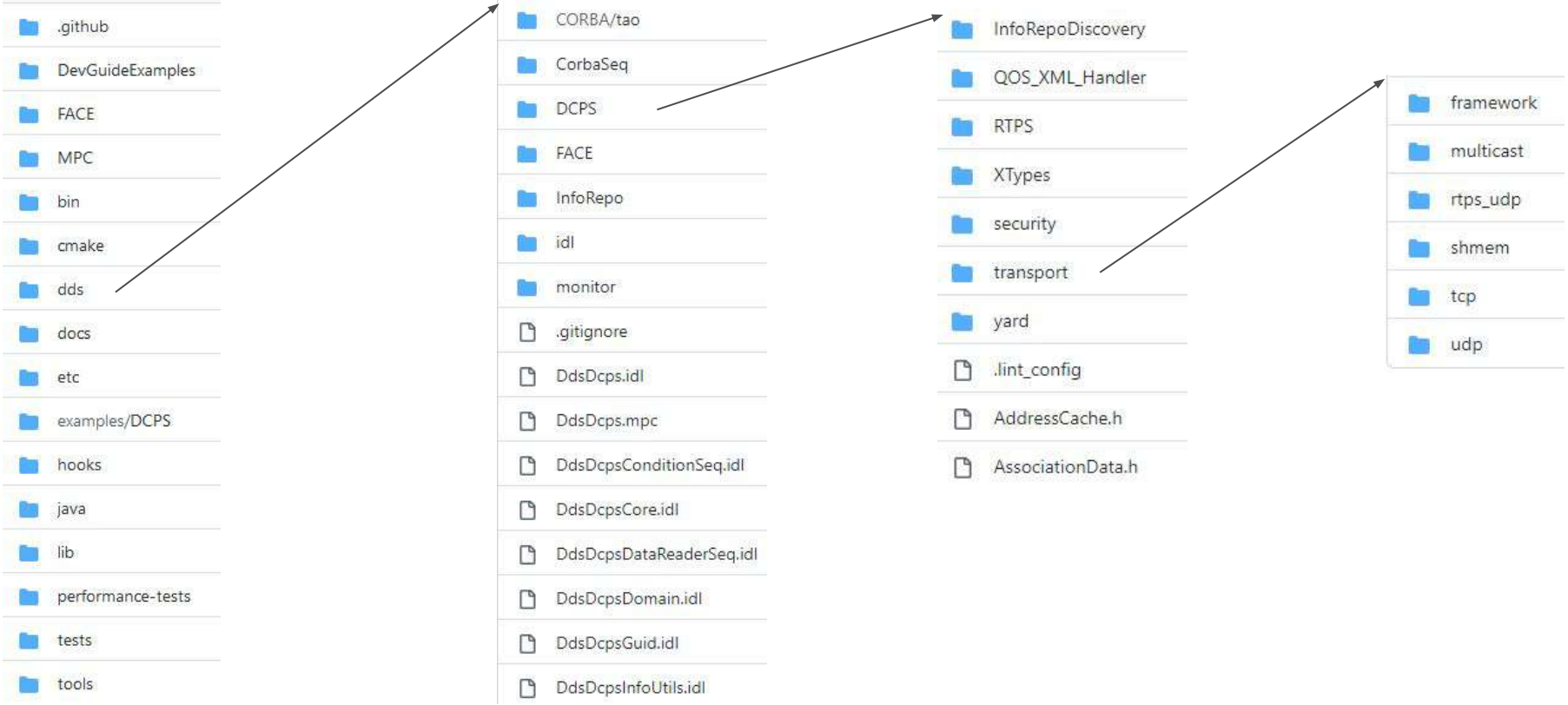
OMG DDS	Data Distribution Service 1.4 (external)
OMG DDSI-RTPS	The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification, OMG formal/19-04-03 (external)
OMG DDS Security	DDS Security 1.1, OMG formal/18-04-01 (external)
OMG DDS XTypes	DDS XTypes 1.3, OMG formal/20-02-04 (external)
Using OpenDDS	Developer's Guide [PDF]
	Building
	Examples
	OpenDDS-Bench
	OpenDDS-Monitor
	OpenDDS Real-Time Data (RTD) for Excel
OpenDDS Architecture	Architecture
OpenDDS Implementation	Summary
	Doxygen

OMG specs that OpenDDS implements

What do users of OpenDDS need to know to build applications?

What do users need to know to build OpenDDS ?

Guided Tour of Resources - GitHub repository



Guided Tour of Resources - Development Guidelines

docs/internal/dev_guidelines.rst

- Style Guide
- Conventions
- External dependencies
- Date and time types
- Logging, log levels, debug levels

OpenDDS Development Guidelines

This document organizes our current thoughts around development guidelines in a place that's readable and editable by the overall user and maintainer community. It's expected to evolve as different maintainers get a chance to review and contribute to it.

Although ideally all code in the repository would already follow these guidelines, in reality the code has evolved over many years by a diverse group of developers. At one point an automated re-formatter was run on the codebase, migrating from the GNU C style to the current, more conventional style, but automated tools can only cover a subset of the guidelines.

Repository

The repository is hosted on Github at objectcomputing/OpenDDS and is open for pull requests.

Automated Build Systems

Pull requests will be tested automatically and full CI builds of the master branch can be found at http://scoreboard.ociweb.com/oci-dds.html.

See :doc:`running_tests` for how tests are run in general. See :doc:`github_actions` for how building and testing is done with GitHub Actions.

Testing

The OpenDDS repository contains four kinds of tests:

1. Unit test - exercises a specific source code module X.(h|cpp|inl)
   `$DDS_ROOT/tests/unit-tests`
2. Integration test - exercises a feature (end-to-end) in a semi-realistic fashion
   `$DDS_ROOT/tests`
   `$DDS_ROOT/java/tests`
3. Stress test - a unit test or integration test that is repeatedly executed
   `$DDS_ROOT/tests/stress-tests`
4. Performance test - a test to measure the end-to-end performance
   `$DDS_ROOT/performance-tests/bench`

Adding a Unit Test

1. Pick the source code module that will be tested.
   `$DDS_ROOT/dds/DCPS/DisjointSequence.(h|cpp|inl)`
2. Create a unit test file for the source code module if one does not exist
   `$DDS_ROOT/tests/unit-tests/dds/DCPS/DisjointSequence.cpp`
3. Run MPC to add the new file.
4. Write tests
   a. Unit tests use gtest and gmock from Google Test
   b.
   ```
   TEST(TestSuiteName, TestName)
   {
     …
   }
   ```
   c. The TestSuiteName is important and should match the path to the source code module
   `TestSuiteName => dds_DCPS_DisjointSequence`
   d. The TestName should be descriptive.
5. Build

Adding a Unit Test

If the source code module is conditionally defined, the tests should also be conditionally defined.

All of the tests are combined in a single executable. Use anonymous namespaces for scoping.

See `$DDS_ROOT/docs/internal/unit_tests.rst` for more information.

Running the Unit Tests

- Execute `$DDS_ROOT/tests/unit-tests/UnitTests` directly
  a. Can pass --gtest_filter=TestCaseName to run tests for a specific source code module
- Execute `$DDS_ROOT/tests/unit-tests/run_test.pl`
- If you have a coverage build
  a. Execute `$DDS_ROOT/tools/scripts/unit_test_coverage.sh`

Adding an Integration Test

- $DDS_ROOT/tests
  - **DCPS - tests of user facing features**
  - FACE - tests related to the FACE specification
  - Utils - testing utilities
  - cmake - tests related to the use of cmake
  - dissector - tests for the Wireshark Dissector plugin
  - googletest - gtest/gmock submodule
  - security - tests related to the DDS Security specification
  - stress-tests - unit tests and integration tests for stressing certain components
  - transport - tests related to transports (participants, readers, writers, etc. are typically mocked)
  - unit-test - See previous.

Adding an Integration Test

The `$DDS_ROOT/tests/DCPS/HelloWorld` test is a starting point for adding an integration test.

1. Create a directory for the test `$DDS_ROOT/tests/DCPS/MyTest`
2. Write IDL if needed
   - Alternatively, use the `ConsolidatedMessengerIdl` (see the `Deadline` test for an example)
3. Write C++ programs for the different participants
   - Java tests belong in `$DDS_ROOT/java/tests` and will follow a similar pattern
4. Write an MPC file to compile the IDL and C++ programs
5. Write a driver script called `run_test.pl`
   - Adds libraries
   - Form command lines
   - Combines exit statuses to determine the overall pass/fail
6. Add the test to the list of automated tests (typically `tests/dcps_tests.lst`)
   - Command line: configuration flags
   - `tests/DCPS/Deadline/run_test.pl: !DCPS_MIN !OPENDDS_SAFETY_PROFILE`
   - `tests/DCPS/Deadline/run_test.pl rtps_disc: !DCPS_MIN !NO_MCAST RTPS`
   - `$DDS_ROOT/tests/auto_run_tests.pl --java --security --list-all-configs`
7. Write a README

Hints When Writing an Integration Test

Unequivocally, the biggest problem when writing tests is **coordination**.

- When possible, we suggest a single process test.
  - Can use locks, condition variable, semaphores, etc.
  - They are easier to debug.
- Use Reliability QoS Policy and other features to your advantage
- Some features are not implemented in all transports
  - E.g., wait_for_acknolwegements doesn't work in the multicast transport
- Do not sleep
  - Use Utils::wait* to wait for publishers, subscribers, samples, etc.
  - Use the Distributed Condition Set to synchronize between processes
- Use the TestFramework Perl module in `run_test.pl`

Assume your test will be running on an overutilized and underpowered VM (because it will be)

Running Integration Tests

To run a single test:

```
./run_test.pl [-Config FLAG …] ARGUMENTS
```

To run all of the tests:

```
$DDS_ROOT/tests/auto_run_tests.pl [-Config FLAG …]
```

Running and Interpreting the Lint Script

OpenDDS has a lint script for checking certains aspects of coding style.

To run the lint scripts:

```
./tools/scripts/lint.pl --color --ace
```

I introduced a tab into a header file and got this output:

```
NOTE Running OpenDDS Lint Checks: ace_condition eof_newline_count gettimeofday is_binary is_empty
missing_include_guard nonrelative_include_path path_has_whitespace tabs trailing_whitespace
whitespace_before_newline_in_string
ERROR: dds/DCPS/DisjointSequence.h:26: ·  DisjointSequence();
ERROR: dds/DCPS/DisjointSequence.h has failed the following checks:
  - tabs
    - Text file has tabs
    - Failed on line(s): 26
```

Contributing Via A Pull Request

Where to Start:

- GitHub Issues: https://github.com/objectcomputing/OpenDDS/issues
  - Many issues have been labeled *beginner*, *intermediate*, or *advanced*
  - Be sure to read full comment thread
    - Coordination with others / helpful discussion
  - Create a new issue for unlisted bugs / features
- GitHub Discussions
  - Good for general discussions about new features / roadmap
- Review other Pull Requests
  - Coordinate with other active changes

Contributing Via A Pull Request

General git / GitHub Workflow:

- Clone, Configure, Build OpenDDS
  - `git clone git@github.com:objectcomputing/OpenDDS.git`
- Fork the OpenDDS Repository on GitHub and add your fork as a git remote
  - `git remote add mine git@github.com:github-user/OpenDDS.git`
- Create a new branch from OpenDDS master
  - `git checkout -b my_pull_request_branch`
- Modify code and verify relevant tests
- Commit changes and push branch to your own repository
  - `git commit -a -m "A brief description of my changes"`
  - `git push mine my_pull_request_branch`
- Follow link from git output or use GitHub to open Pull Request

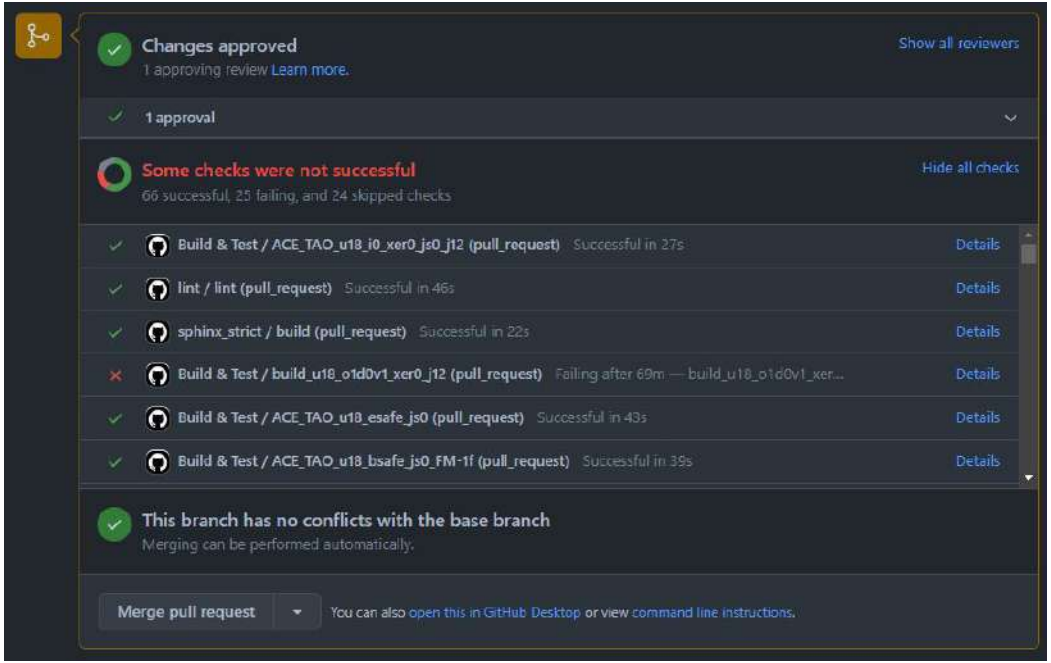
Contributing Via A Pull Request

GitHub Actions

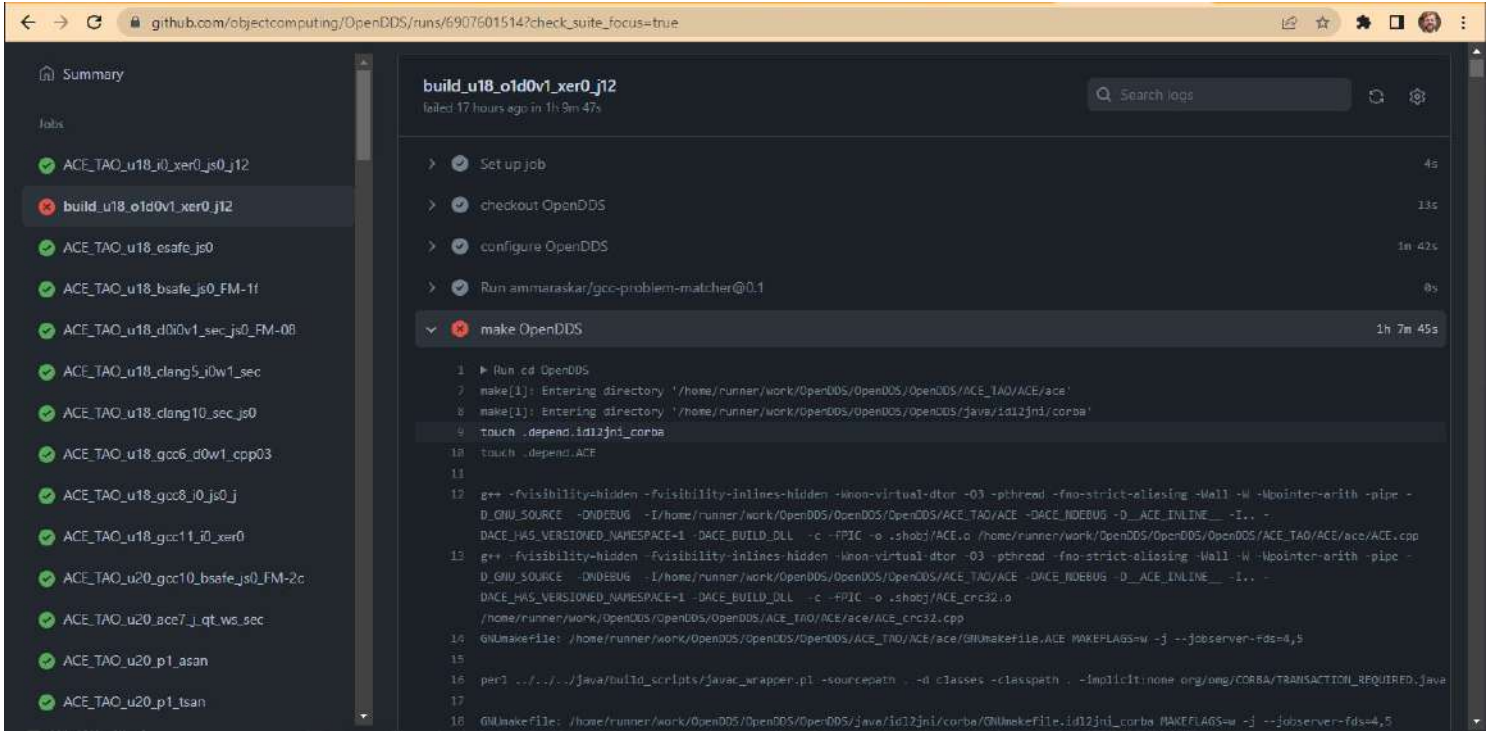
- Interpreting CI Results

Most Common Issues:

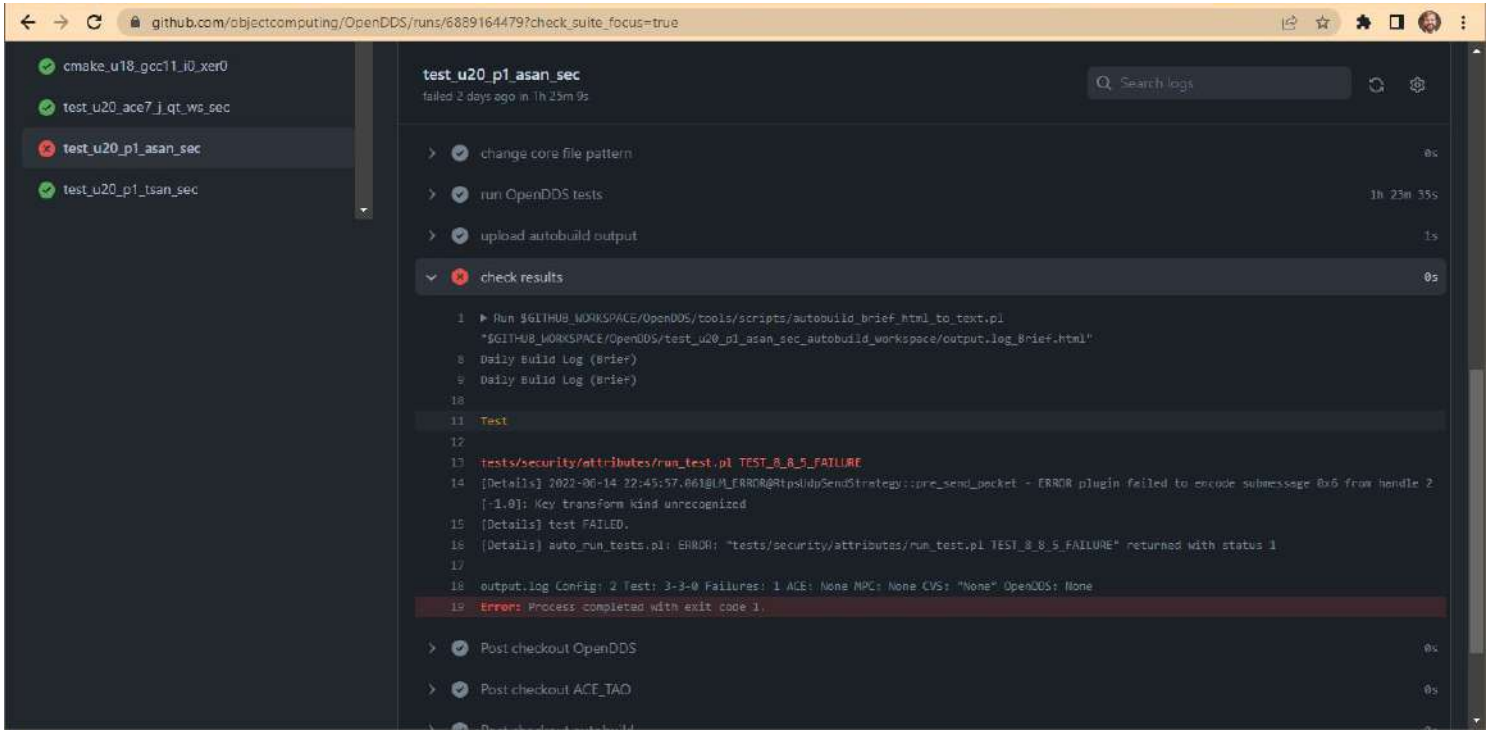
- Merge Conflict
- Lint Failures
- Build Failures
- Test Failures



Contributing Via A Pull Request



Contributing Via A Pull Request



Contributing Via A Pull Request

Post-Merge

- OpenDDS Scoreboard: http://scoreboard.ociweb.com/oci-dds.html
- OpenDDS Performance Dashboard: http://scoreboard.ociweb.com/bench2

DDS Build Scoreboard

Test Matrix | New Matrix | Footprint | Doxygen | Javadoc | Coverage

Also see DDS Builds at Vanderbilt (DOC Group)

DOC

Build Name	Last Finished	Rev	Config	Setup	Compile	Tests	Failures	Status
Remedy_Fedora_Versioned	06/15 11:57 pm	3439a81c	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	4	Inactive
Remedy_RHEL80_Versioned	06/16 12:36 am	3439a81c	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	1	Inactive
Remedy_VS2019	06/15 09:51 pm	3439a81c	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	2	Inactive
dds_doc_Latest_Micro_ant_linux_gcc_d1o0	06/13 10:41 am	3439a81c	[Config]	[Full]	[Full]	[Full]	0	Inactive
dds_doc_Latest_Micro_bee_win7_vc12_i1d0o1s1m1	06/13 03:17 pm	3439a81c	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	2	Inactive
dds_doc_Latest_Micro_spider_linux_gcc_d1o0	06/13 10:41 am	3439a81c	[Config]	[Full]	[Full]	[Full]	0	Inactive
dds_doc_ace6tao2_ant_linux-rpi_gcc_d0o1	06/14 10:38 am	3439a81c	[Config]	[Full]	[Full] [Brief]		0	Inactive
dds_doc_ace6tao2_ant_linux-sb_gcc_d0o1	06/14 10:11 am	3439a81c	[Config]	[Full]	[Full]	[Full] [Brief]	1	Inactive
dds_doc_ace6tao2_cricket_win2012r2-x64_vc10_d1o0	06/15 05:24 am	3439a81c	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	6	Inactive
dds_doc_ace6tao2_earwig_linux_gcc_d1o0	06/15 12:10 am	3439a81c	[Config]	[Full]	[Full] [Brief]	[Full]	0	Inactive
dds_doc_ace6tao2_firefly_android_clang_d1o0	06/15 03:51 am	3439a81c	[Config]	[Full]	[Full]		0	Inactive
dds_doc_ace6tao2_firefly_linux_gcc_d0o1i0e1	06/13 10:48 am	3439a81c	[Config]	[Full]	[Full]	[Full]	0	Inactive
dds_doc_ace6tao2_hornet_win7-x64_vc14_i1d1o0w1	06/15 03:44 am	3439a81c	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	15	Inactive
dds_doc_ace6tao2_katydid_win10-x64_vs2019_d1o0	06/15 01:17 am	3439a81c	[Config]	[Full]	[Full] [Brief]	[Full]	0	Inactive
dds_doc_ace6tao2_mantis_linux_gccasan_d1o0	06/14 10:44 am	3439a81c	[Config]	[Full]	[Full]	[Full] [Brief]	1	Inactive
dds_doc_ace6tao2_mosquito_linux_gcc_d0o1i0c1	06/14 09:05 am	3439a81c	[Config]	[Full]	[Full] [Brief]	[Full] [Brief]	2	Inactive
dds_doc_ace6tao2_mosquito_linux_inst_n1	06/13 12:05 pm	3439a81c	[Config]	[Full]	[Full]		0	Inactive

Contributing in other areas of OpenDDS

- Developer's Guide
  - Open Issue or Discussion on GitHub or bring it up on the mailing lists
- Sphinx Documentation
  - Hosted on Read the Docs: https://opendds.readthedocs.io/en/latest/
  - Uses Sphinx documentation framework
  - Content is written in reStructuredText
  - Located in the `docs` directory in the repo
  - Can be built locally using the `docs/build.py` script
- opendds.org Website
  - Uses Jekyll static website generator
  - Content is written in HTML and Markdown
  - Located in `gh-pages` branch of the repo
  - Can be built and served locally.

Contributing to Related Repositories

- MPC
  - MPC is the build system, used to configure the build and generate platform specific build files (Makefiles, VS solution files, etc.)
  - Written in Perl
  - gnuace, used on Linux and macOS, resides in ACE/TAO
- ACE/TAO
  - ACE is a library used for cross-platform compatibility, especially networking and event loops.
  - TAO is a C++ CORBA implementation built on ACE. Used for IDL parsing, IDL-to-C++ Mapping, and InfoRepo discovery.

Contributing to Related Repositories

- Node-OpenDDS
  - Interact with OpenDDS in Javascript running on NodeJS
- PyOpenDDS
  - Interact with OpenDDS in Python
- OpenDDSharp
  - Interact with OpenDDS in C#

Thank you!

Any Questions?

OpenDDS FOUNDATION™

Further Study

OCI training (beyond OpenDDS)

objectcomputing.com/training

OpenDDS project

opendds.org

Source repo

github.com/objectcomputing/OpenDDS

OpenDDS support, training, consulting, development

objectcomputing.com/products/opendds

OpenDDS 3.20 Release Notes

github.com/objectcomputing/OpenDDS/releases/tag/DDS-3.20

Further Study: On-Demand Webinars

Data Distribution with an Open and Secure DDS (DDS Security)

objectcomputing.com/resources/events/webinars/opendds-security

Designing a Distributed Application using DDS QoS

brighttalk.com/webcast/12231/281491

What's New in the 3.14 Release (IDL Annotations, C++11 support, and more)

objectcomputing.com/products/opendds/resources/introducing-opendds-3-14

XTypes in OpenDDS 3.16

objectcomputing.com/products/opendds/resources/introducing-xtypes

OpenDDS Foundation

OpenDDS Foundation is a not-for-profit organization that exists to support and collectively lead the open source OpenDDS® project. The Foundation is supported by a Technology Advisory Board that ensures the technology continues to reflect and serve its diverse and growing user community.

OpenDDS Foundation works to ensure technical innovation and advancement of the OpenDDS project, evangelize and promote the project as a leading technology in the data distribution space, and build and support an ecosystem of complementary documentation, functionality, and services.

As a not-for-profit organization, OpenDDS Foundation relies on the financial support of contributing members to support and grow the project. Businesses and community members are encouraged to actively participate in the project's success by becoming contributing members through one of our sponsorship programs.

LET'S CONNECT!


info@opendds.org


opendds.org/foundation


@OpenDDS


linkedin.com/showcase/opendds


github.com/objectcomputing/OpenDDS