Getting Started Guide: ACE, TAO, and OpenDDS VxWorks Source Build Layers

OCI ACE+TAO 2.2a_p25 https://theaceorb.com
OpenDDS 3.23 https://opendds.org
Tested with Wind River VxWorks 21.03

By Object Computing, Inc. https://objectcomputing.com
Document Revision 10/28/2022

Table of Contents

Introduction

OCI (a Wind River® Partner) provides complimentary pre-built packages of ACE, TAO, and OpenDDS for VxWorks®, Wind River's leading real-time operating system. This document provides complete instructions for working with the packages. Additional help, including professional training, consulting, and support, is available by contacting OCI (see Resources and Documentation).

OpenDDS is an open source, C++ implementation of the Object Management Group (OMG) Data Distribution Service (DDS) for real-time systems specification. DDS defines a strongly typed, data-centric, publish-subscribe middleware for efficiently distributing application data between participants in a distributed application. OpenDDS leverages the ADAPTIVE Communication Environment (ACE) to provide portability and configurability across a wide variety of operating systems and deployment environments, including VxWorks®.

OpenDDS's quality of service (QoS) features include implementation of the full DDS specification and make it suitable for use in real-time and embedded systems. OpenDDS also supports the Real-time Publish-Subscribe Wire Protocol (DDSI-RTPS) specification, making it ready for secure, dynamic discovery and data distribution in highly distributed environments, and the DDS Security specification, providing applications with authentication, access control, and cryptographic operations.

Additional Dependencies

On Windows host systems, ActiveState Perl is required since the VSB build process for the ACE, TAO, and OpenDDS layers runs Perl scripts.
https://www.activestate.com/products/activeperl/downloads

In order to build the DDS Security optional feature of OpenDDS, Wind River's Cryptography Libraries for VxWorks 21.03 must be installed.

In order to build the DDS Security optional feature of OpenDDS, the Apache Xerces-C++ XML parser version 3.2.3 is required (http://archive.apache.org/dist/xerces/c/3/sources). Detailed build instructions for version 3.2.3 can be found below, note that they may not work with any other version. Xerces-C++ uses CMake as its build system. Install CMake from https://cmake.org/download or use the system package manager (if using a Linux host).

RPM Packages

The following RPMs are available: **oci_ace**, **oci_tao_host_linux**, **oci_tao_host_windows**, **oci_tao**, **oci_opendds_host_linux**, **oci_opendds_host_windows**, and **oci_opendds**.

The **oci_ace** package contains a VxWorks Layer that (when included in a VxWorks Source Build) builds a number of ACE libraries for VxWorks 21.03. Aside from the core ACE library (`libACE.a` or `libACE.so`), this package includes ACE's utilities for XML and ETCL parsing.

Building both TAO and OpenDDS requires the tao_idl compiler (and the ace_gperf utility that it uses). Those programs generate code at build-time so they are not in VxWorks Layers but instead they are separate RPMs (**oci_tao_host_linux** or **oci_tao_host_windows**) that install binaries to the `partners` directory in the top-level VxWorks installation.

The **oci_tao** package contains a VxWorks Layer that builds TAO's libraries, utilities, and ORB Services. The libraries and executables built include everything in the TAO_ACE workspace (TAO_ACE.mwc) that's applicable to the VxWorks target environment.

Since OpenDDS uses a separate code generator, opendds_idl, in addition to TAO's code generation tools, there are separate host-side OpenDDS packages (**oci_opendds_host_linux** or **oci_opendds_host_windows**).

The **oci_opendds** package contains a VxWorks Layer that builds the OpenDDS middleware libraries along with some executable tools and examples (based on OpenDDS_no_tests.mwc).

The three packages that install VxWorks Layers use the "net" subdirectory so they can be found at `{WIND_PKGS}/net/{package}-{version}`.

The version numbers for these RPMs contain the released product version numbers (first three digits) followed by an RPM revision number (final digit). Note that OCI ACE+TAO, when treated as a unit, takes on TAO's version number. The corresponding ACE release is 4 major versions ahead. The third digit for OCI ACE and TAO RPMs is the patch level, therefore general release OCI ACE+TAO 2.2a_p25 becomes RPMs 6.2.25._ (ACE) and 2.2.25._ (TAO).

The ACE and TAO RPMs version 6.2.25.0/2.2.25.0 contain minor updates compared to the general release of ACE+TAO 2.2.a_p25. These changes are noted above the “ACE/TAO version X released” entry in the files ACE/OCIChangeLog and TAO/OCIChangeLog that are installed along with the source code in their respective layers. The OpenDDS RPM version 3.23.0.0 also contains minor updates compared to the general release of OpenDDS 3.23. These changes are noted in the OpenDDS/ChangeLog file within the installed layer directory.

Installing RPMs

RPMs can be directly installed using the process described in the Wind River Product Installation and Licensing Developer's Guide, 2.7.5 Appendix A. Follow the instructions in the section “Sample Command-Line Sequence: Using Yum to Install a Single RPM Package”. Instead of a single RPM Package, the entire set of related packages should be installed at the same time in order to meet the dependencies required. In place of <absolute path to rpm file> in Wind River’s instructions, provide all RPMs required (absolute or relative path should work fine). See the section RPM Packages above for the descriptions of the RPMs.

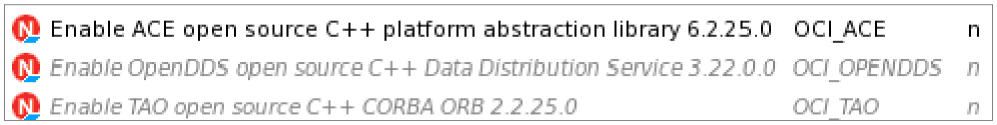
Configuring a VxWorks Source Build (VSB)

To start using ACE/TAO/OpenDDS, create a new VSB or configure an existing VSB to enable the required layers. See the Wind River VxWorks 21.03 document “Configuration and Build Guide” for more information on using the VSB Configuration tool in Wind River Workbench (or the command line).

ACE/TAO/OpenDDS on VxWorks are compiled with the LLVM Compiler Infrastructure (clang). This version of VxWorks includes LLVM 11.0.1.1.

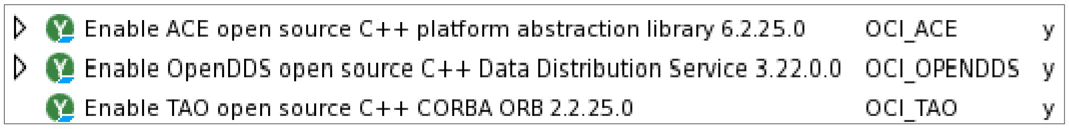
If you will be using Xerces-C++ (for DDS Security), build the VSB now with the ACE, TAO, and OpenDDS layers still excluded. This VSB can now be used as the basis for the Xerces-C++ build. Follow the steps in the section Building Xerces-C++ for VxWorks below and then return here to continue with the ACE/TAO/OpenDDS configuration.

The ACE, TAO, and OpenDDS layers start off as excluded. They can be found in the “net” category:



N	Enable ACE open source C++ platform abstraction library 6.2.25.0	OCI_ACE	n
N	*Enable OpenDDS open source C++ Data Distribution Service 3.22.0.0*	*OCI_OPENDDS*	*n*
N	*Enable TAO open source C++ CORBA ORB 2.2.25.0*	*OCI_TAO*	*n*

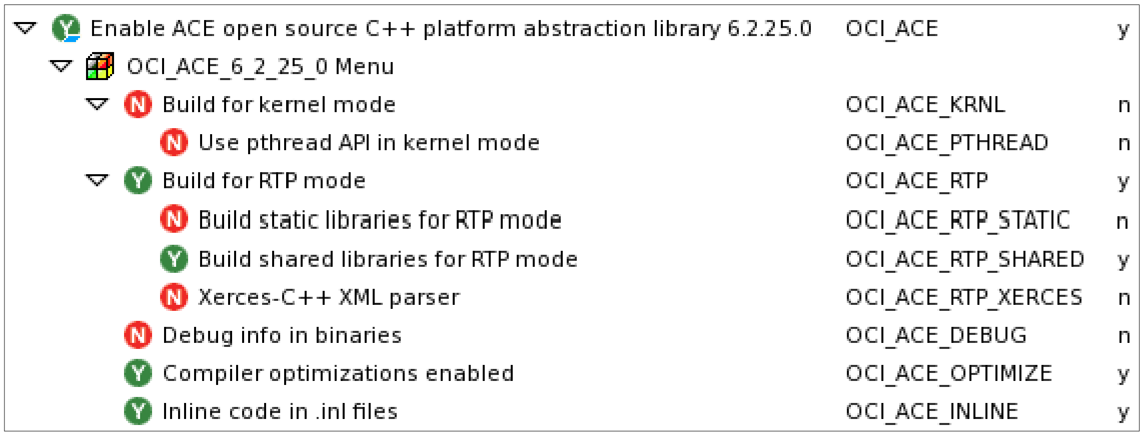
Since ACE is not currently included, TAO and OpenDDS (if installed) are showed as greyed-out and italicized. Select any layer to include, right-click it, and use the “Add with Dependencies” command from the context menu. In the example below, OpenDDS was added (which automatically added both ACE and TAO):



▷	Y	Enable ACE open source C++ platform abstraction library 6.2.25.0	OCI_ACE	y
▷	Y	Enable OpenDDS open source C++ Data Distribution Service 3.22.0.0	OCI_OPENDDS	y
	Y	Enable TAO open source C++ CORBA ORB 2.2.25.0	OCI_TAO	y

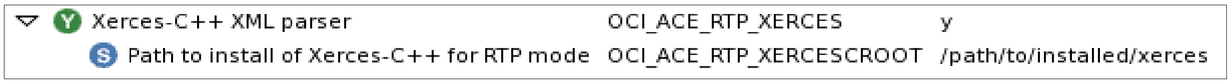
Note that the ACE and OpenDDS layers each have submenus, shown as an arrow icon to the left.

There are additional configuration options available under ACE. Since these options change how ACE is compiled, they are controlled by the ACE layer and are not separately configurable under TAO and OpenDDS (though the settings also impact how TAO and OpenDDS are built).



▽ Y	Enable ACE open source C++ platform abstraction library 6.2.25.0	OCI_ACE	y
▽	OCI_ACE_6_2_25_0 Menu		
▽ N	Build for kernel mode	OCI_ACE_KRNL	n
N	Use pthread API in kernel mode	OCI_ACE_PTHREAD	n
▽ Y	Build for RTP mode	OCI_ACE_RTP	y
N	Build static libraries for RTP mode	OCI_ACE_RTP_STATIC	n
Y	Build shared libraries for RTP mode	OCI_ACE_RTP_SHARED	y
N	Xerces-C++ XML parser	OCI_ACE_RTP_XERCES	n
N	Debug info in binaries	OCI_ACE_DEBUG	n
Y	Compiler optimizations enabled	OCI_ACE_OPTIMIZE	y
Y	Inline code in .inl files	OCI_ACE_INLINE	y

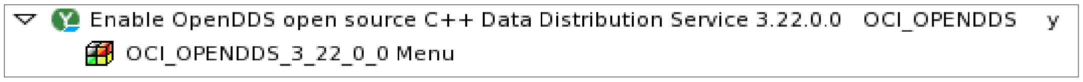
If Xerces-C++ was built (needed for DDS Security, see Building Xerces-C++ for VxWorks below), enable the OCI_ACE_RTP_XERCES option and provide the path to the installed Xerces-C++ in the option below it. This is the directory above the “include,” “lib,” and “shared” directories created by the Xerces-C++ installation process. Use forward slashes on all host platforms.



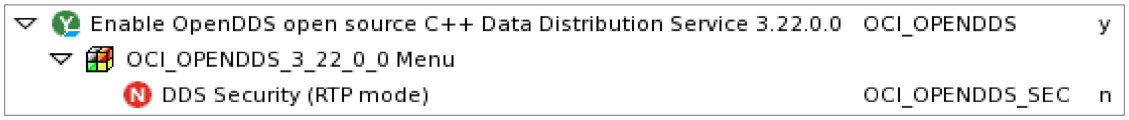
▽ Y	Xerces-C++ XML parser	OCI_ACE_RTP_XERCES	y
S	Path to install of Xerces-C++ for RTP mode	OCI_ACE_RTP_XERCESCROOT	/path/to/installed/xerces

The OpenDDS layer also has a submenu with an arrow icon to the left. It holds an additional configuration option to enable DDS Security. In addition to OCI_ACE_RTP_XERCES option described above, the OPENSSL layer (in the VSB’s “security” category) must be enabled before the DDS Security feature can be enabled. Note that using the default Source Build

Configuration view options, the OpenDDS Menu will be empty until the required layers and options are enabled:


```
▽ Enable OpenDDS open source C++ Data Distribution Service 3.22.0.0   OCI_OPENDDS   y
      OCI_OPENDDS_3_22_0_0 Menu
```

Once the required layers and options are enabled, the OpenDDS Menu will be populated:


```
▽ Enable OpenDDS open source C++ Data Distribution Service 3.22.0.0   OCI_OPENDDS        y
   ▽ OCI_OPENDDS_3_22_0_0 Menu
        DDS Security (RTP mode)                                          OCI_OPENDDS_SEC    n
```

With this menu now available, the DDS Security option is visible and can be enabled.

When the VSB is built, the selected ACE, TAO, and OpenDDS libraries and executables will be built alongside the VxWorks source code. Since there are interactions among the ACE, TAO, and OpenDDS layers, the “Enable Parallel Builds” option in workbench is not supported.

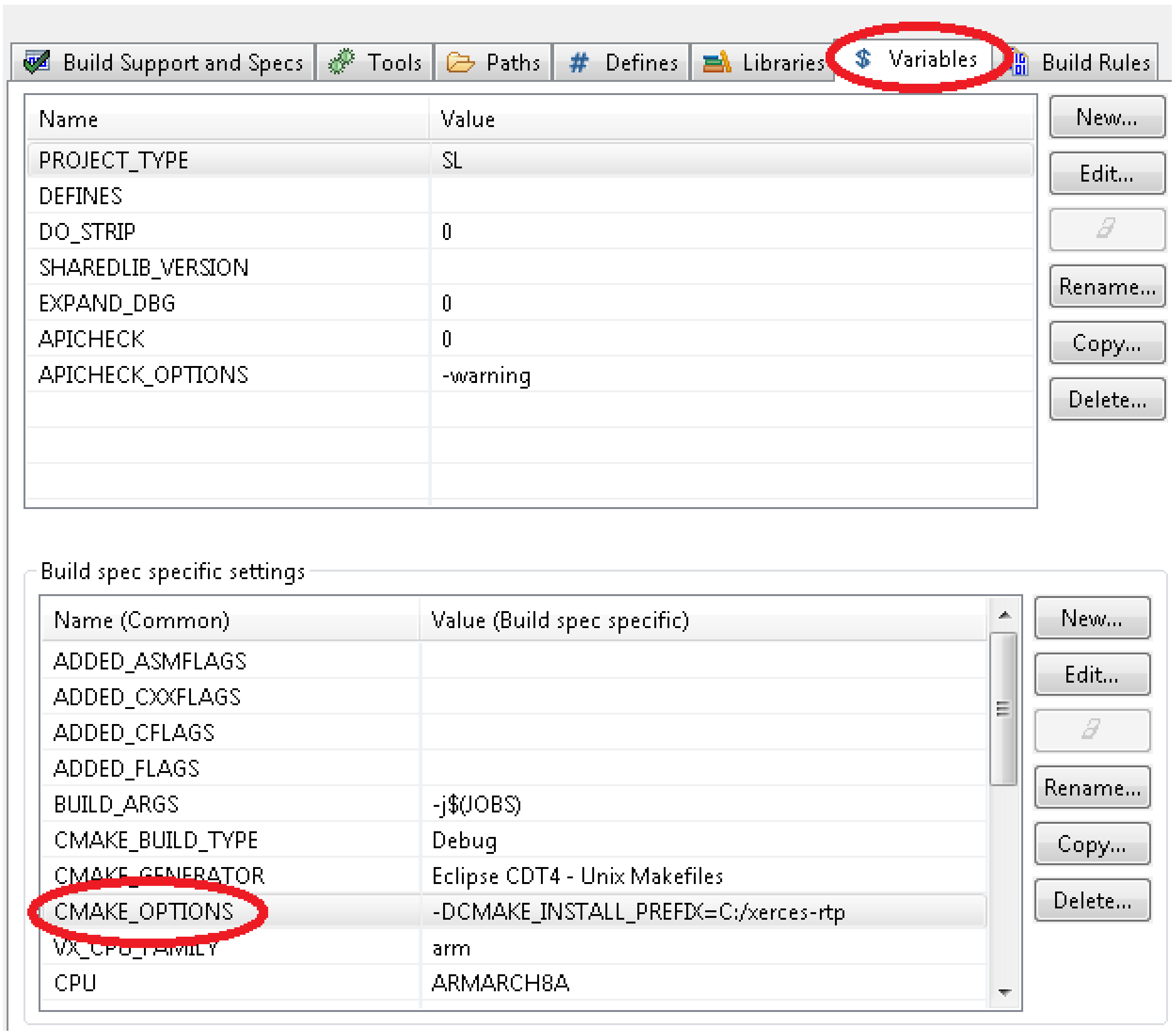
Building Xerces-C++ for VxWorks

Download and extract the source code for the Apache Xerces-C++ XML parser version 3.2.3 (http://archive.apache.org/dist/xerces/c/3/sources). These instructions will not work with any other version.

Xerces-C++ uses CMake as its build system. Install CMake from https://cmake.org/download so that the “cmake” executable is on the PATH. If PATH was changed, make sure to restart Wind River Workbench and any development shells so that they have the new PATH setting.

In Workbench, create a new project of type “CMake Shared User Library” with a name of your choice. This example will use the name “xerces_rtp” and will create the project in the workspace. The screenshot below shows the Project Setup required. The “Project context” points to the VSB that was built in the previous step. The “Import the content...” directory refers to the expanded Xerces-C++ source code. Note that these files are copied into the Workbench project and therefore are not modified.

Once the project is created, edit its Build Properties. Configure the build spec on the Variables tab by editing the CMAKE_OPTIONS variable. Set the variable’s value to “-DCMAKE_INSTALL_PREFIX=/path/to/installed/xerces”. This is a directory that doesn’t exist yet. Once Xerces-C++ is built and installed here, this is the directory that will be used to configure the ACE VSB layer so it can find Xerces-C++. On Windows use a full path starting with a drive letter and colon, and use forward slashes.



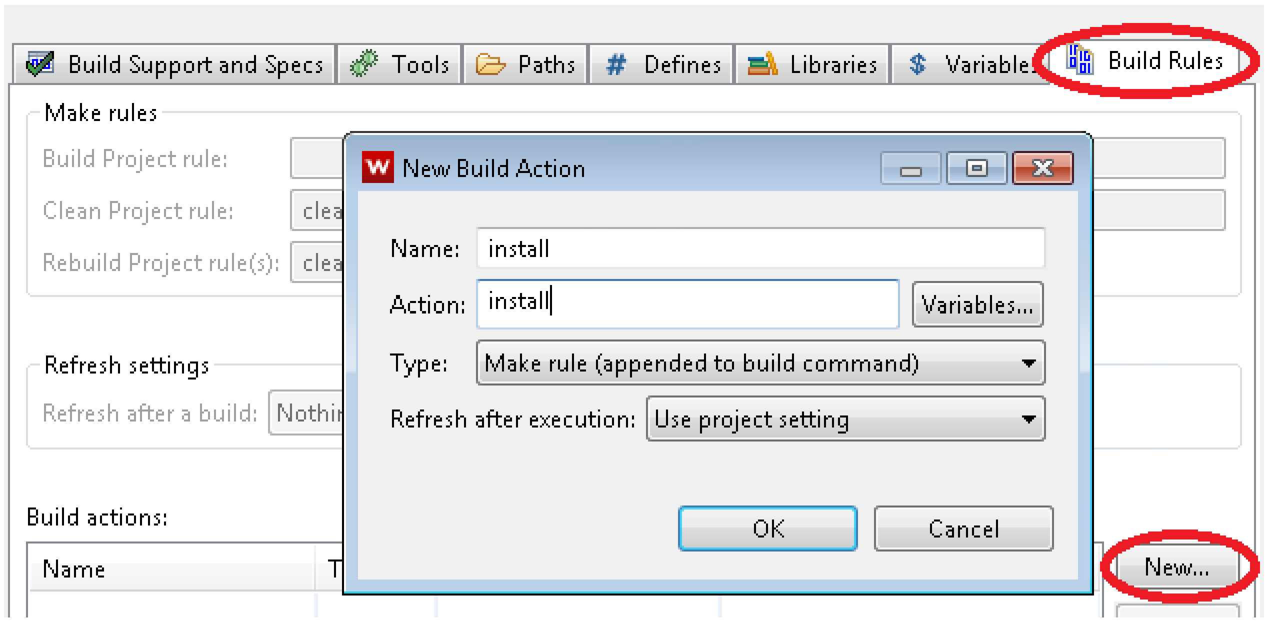
Build Support and Specs | Tools | Paths | Defines | Libraries | Variables | Build Rules

Name	Value
PROJECT_TYPE	SL
DEFINES	
DO_STRIP	0
SHAREDLIB_VERSION	
EXPAND_DBG	0
APICHECK	0
APICHECK_OPTIONS	-warning

Build spec specific settings

Name (Common)	Value (Build spec specific)
ADDED_ASMFLAGS	
ADDED_CXXFLAGS	
ADDED_CFLAGS	
ADDED_FLAGS	
BUILD_ARGS	-j$(JOBS)
CMAKE_BUILD_TYPE	Debug
CMAKE_GENERATOR	Eclipse CDT4 - Unix Makefiles
CMAKE_OPTIONS	-DCMAKE_INSTALL_PREFIX=C:/xerces-rtp
VX_CPU_FAMILY	arm
CPU	ARMARCH8A

Continue configuring the build spec: add a new build action for install:


Build Support and Specs | Tools | Paths | Defines | Libraries | Variables | Build Rules

Make rules

Build Project rule:

Clean Project rule: clea

Rebuild Project rule(s): clea

Refresh settings

Refresh after a build: Nothi

Build actions:

New Build Action

Name: install

Action: install

Type: Make rule (appended to build command)

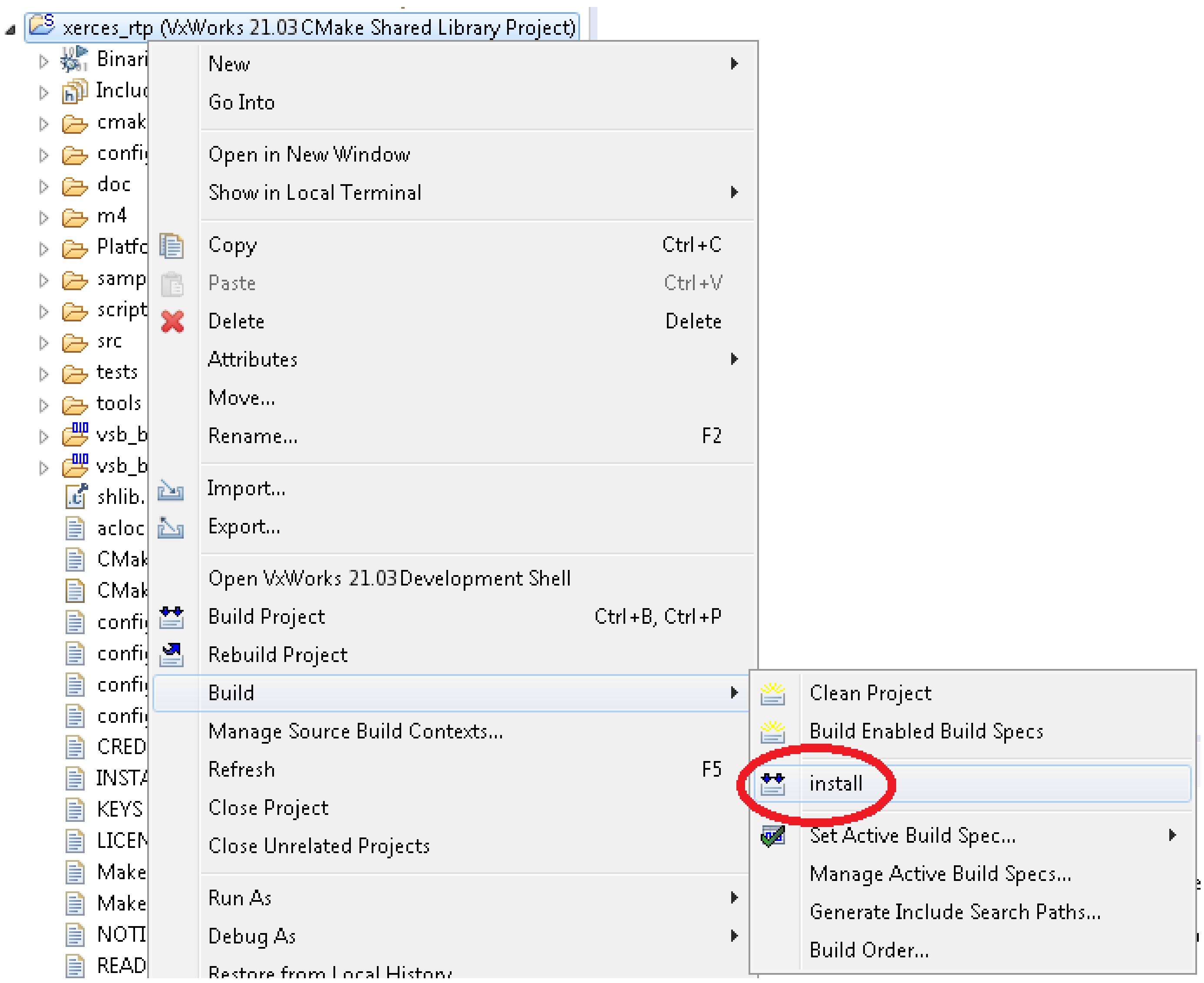
Refresh after execution: Use project setting

If the ACE/TAO/OpenDDS build will use RTP static libraries, create a new build spec for the static configuration. Return to the first tab “Build Support and Specs” and use the “Copy...” button (above “Delete...”) to create a new spec. Name the new spec with a suffix of “_static” added to the existing spec name. Change the Active build spec to this new one and edit its CMAKE_OPTIONS (on the Variables tab) by appending “ -DBUILD_SHARED_LIBS=OFF”. Restore the Active build spec to the original build spec (the one without _static).

Use a VxWorks Development Shell to run the included Perl script that prepares the Xerces-C++ source code and CMake environment for cross-compilation. This example assumes the Develpoment Shell starts in Workbench’s Workspace directory and changes to the Project directory for this project.

```
cd xerces_rtp
[Windows] perl %WIND_PKGS%\net\oci_ace-6.2.25.0\misc\xerces_vxworks.pl
[Linux] perl $WIND_PKGS/net/oci_ace-6.2.25.0/misc/xerces_vxworks.pl
```

Build the project using the “install” target:



If you have created an additional build spec for static libraries, activate that build spec and then build again using the “install” target.

Building Applications Using ACE, TAO, and OpenDDS

Include files for ACE, TAO, and OpenDDS are installed into the VSB. The C++ preprocessor include path should contain `{VSB_DIR}/krnl/h/public` or `{VSB_DIR}/usr/h/public` for kernel-mode or RTP-mode respectively.

Libraries for ACE, TAO, and OpenDDS are located in the VSB directory in `{VSB_DIR}/krnl/{CPU}/common` or `{VSB_DIR}/usr/lib/common`.

VxWorks executables for TAO and OpenDDS are located in `{VSB_DIR}/dkm` (Dynamic Kernel Modules) or `{VSB_DIR}/usr/root/llvm/bin` (RTPs). ACE doesn't include any VxWorks executables.

Host executables (code generators) are installed in `{WIND_HOME}/partners/oci_tao-{VERSION}/{HOST_OS}/bin` and `{WIND_HOME}/partners/oci_opendds-{VERSION}/{HOST_OS}/bin`.

Using MPC to Generate Makefiles for Application Code

MPC: The Makefile, Project, and Workspace Creator (http://objectcomputing.com/products/mpc) is installed inside the oci_ace layer (in the MPC subdirectory). All of the makefiles included in the ACE, TAO, and OpenDDS layers were generated by MPC. Since ACE, TAO, and OpenDDS use MPC to generate their makefiles, the included MPC project base files (*.mpb) contain reusable bits of information that can be used to generate makefiles for your own codebase that uses ACE, TAO, and OpenDDS.

The following example shows how an OpenDDS application can be set up to use MPC with the VSB-based ACE, TAO, and OpenDDS. Since OpenDDS applications also implicitly use ACE and TAO, the example can be adapted to ACE-only and TAO-only applications. This example assumes the “wrenv” utility from Wind River has already been run to set up the VxWorks environment. This example is written for a Linux host but it can also be adapted to work on a Windows host.

Set up the environment variables:

Name	Value
`VSB_DIR`	Directory containing the VxWorks Source Build
`MPC_ROOT`	`$WIND_PKGS/net/oci_ace-6.2.25.0/MPC`
`ACE_ROOT`	`$VSB_DIR/src/oci_ace-6.2.25.0`
`TAO_ROOT`	`$VSB_DIR/src/oci_tao-2.2.25.0`
`DDS_ROOT`	`$VSB_DIR/src/oci_opendds-3.23.0.0`
`TAO_HOST_TOOLS`	`$WIND_HOME/partners/oci_tao-2.2.25/LINUX`
`OPENDDS_HOST_TOOLS`	`$WIND_HOME/partners/oci_opendds-3.23.0/LINUX`
`XERCESCROOT`	The directory containing Xerces-C++ (if using DDS Security)

The application's makefile is generated from its .mpc file:

```
project: dcpsexe, dcps_rtps_udp {
  TypeSupport_Files {
    Messenger.idl
  }

  // Add the following line for RTP mode:
  libpaths += $(VSB_DIR)/usr/lib/common

  // Or add the following line for Kernel mode:
  libpaths += $(VSB_DIR)/krnl/$(CPU)/common
}
```

In order to link directly to DDS Security libraries, add `opendds_security` to the list of base projects. This is required for static builds.

Run MPC to generate makefiles (without OpenDDS Security):

```
$ACE_ROOT/bin/mwc.pl -type gnuace
```

Run MPC to generate makefiles (with OpenDDS Security):

```
$ACE_ROOT/bin/mwc.pl -type gnuace -features \
  no_opendds_security=0,openssl=0,no_vxworks_openssl=0
```

Run make:

Determine additional arguments for “make” based on the ACE settings in the VSB configuration. If the VSB is configured as shown in the first column, set the makefile variable in the remainder of that row:

VSB Configuration Setting	**Makefile Variable**	**Value**	**Notes**
Build for RTP mode	rtp	1	
Use pthread API in kernel mode	pthread	1	
Build shared libraries for RTP mode	shared_libs_only	1	Use static_libs_only for *any* kernel build
NO - Debug info in binaries	debug	0	Defaults to 1, specify 0 if “NO”
NO - Compiler optimizations enabled	optimize	0	Defaults to 1, specify 0 if “NO”
NO - Inline code in .inl files	inline	0	Defaults to 1, specify 0 if “NO”

Add the following to the “make” command based on the DDS Security configuration:

- If DDS Security is enabled in the VSB

- `OPENDDS_SECURITY_MACRO=OPENDDS_SECURITY no_opendds_security=0`

For example, if the VSB was set to build for kernel mode, use the pthread API, no debugging info, with optimizations and with inline code:

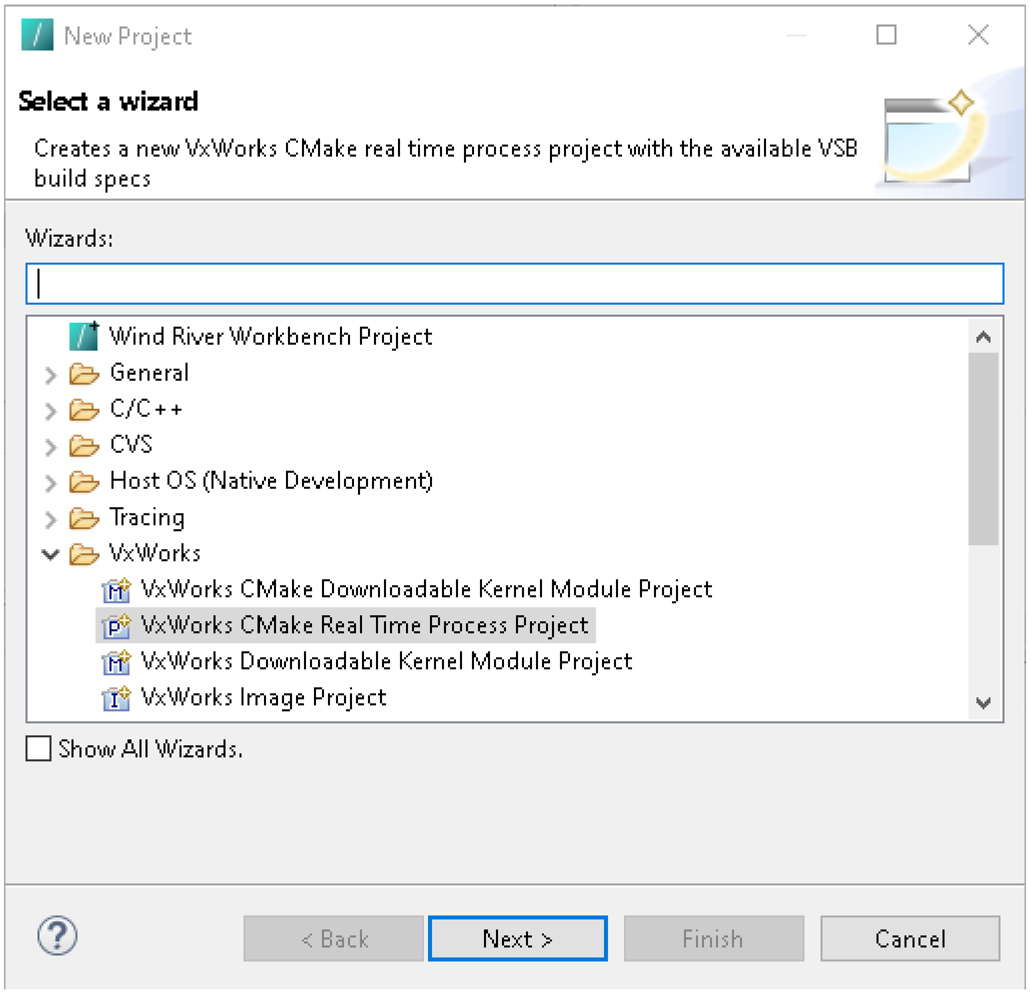
```
make rtp=0 pthread=1 static_libs_only=1 debug=0
```

To build for RTP mode with DDS Security enabled along with the same settings as above except for using shared libraries:

```
make debug=0 OPENDDS_SECURITY_MACRO=OPENDDS_SECURITY no_opendds_security=0
```

Using CMake for Application Code

Workbench has CMake support built into it, which can be used to build applications against ACE, TAO, and OpenDDS. To get started, create a new project and select “VxWorks CMake Real Time Project Project” from the list of wizards.


New Project
Select a wizard
Creates a new VxWorks CMake real time process project with the available VSB build specs
Wizards:
Wind River Workbench Project
General
C/C++
CVS
Host OS (Native Development)
Tracing
VxWorks
VxWorks CMake Downloadable Kernel Module Project
VxWorks CMake Real Time Process Project
VxWorks Downloadable Kernel Module Project
VxWorks Image Project
Show All Wizards.
< Back
Next >
Finish
Cancel

Once the project has been created, a few modifications to the generated CMakeLists.txt will need to be made.

1. Rename the source file from rtp.c to rtp.cpp.
2. Comment the line that adds -static to the CMAKE_EXE_LINKER_FLAGS.
3. Add a call to find_package() to find the TAO module. It should go directly after the project declaration.
4. Add a call to target_link_libraries() to link in the TAO libraries.

```
cmake_minimum_required(VERSION 2.8.7)

project(test C CXX ASM)

#set(CMAKE_EXE_LINKER_FLAGS "${CMAKE_EXE_LINKER_FLAGS} -static")
find_package(TAO REQUIRED)

SET(MY_TARGET ${PROJECT_NAME}.vxe)

add_executable(${MY_TARGET}
        rtp.cpp
)

target_link_libraries(${MY_TARGET} ${TAO_SERVER_LIBS})
```

Once you have renamed the rtp.c file to rtp.cpp by right clicking on the file within the project and selecting *Rename…*, you are ready to build the project. Right click on the project and select *Build Project*.

Resources and Documentation

- OCI ACE https://objectcomputing.com/products/ace
- OCI TAO https://objectcomputing.com/products/tao
    - TAO Developer's Guide available from the above link
- OpenDDS https://opendds.org/
    - OpenDDS Developer's Guide

https://download.objectcomputing.com/OpenDDS/OpenDDS-latest.pdf

- MPC https://github.com/objectcomputing/MPC
- OCI Commercial Support https://objectcomputing.com/products/open-source-support
- OCI Training https://objectcomputing.com/training/