Boost Development Tools
Quick Reference Guide

August, 2025

TenAsys Corporation
1400 NE Compton Drive, #301
Hillsboro, OR 97006 USA
+1 503 748-4720
fax +1 503 748-4730
info@tenasys.com
www.tenasys.com

August, 2025 Edition

Boost Development Tools Quick Reference Guide

Contents

Welcome! — Before You Begin **4**
- Terms 4
- Supported Boost Version 4
- Supported C++ Version 4
- Requirements 4
- Third-Party Modules 5

Install Boost **6**
- Preferred Directory 6
- Install Git 6
- Clone the Boost Repository 7
  - Using Tortoise Git 7
  - Using the command prompt 8
- Check out the specific commit 9
  - Using Tortoise Git 9
  - Using the command prompt 11

Install INtime Boost SDK **12**
- Install Boost172sdk.msi 12
- INtime Boost headers and libraries 13
- Validating Boost Application Wizard in Visual Studio 14
  - Check for Boost Application Wizard 14
  - Troubleshooting Missing Boost Application Wizard 15
  - Setting Up a Visual Studio Boost Project with Property Manager 16

Configure an INtime Boost project **17**
- Creating an Empty Project 18
- Testing Requirements: Installation_Paths.txt 21
- The INtime Boost Solution Directory Overview 22

Visual Studio Property Sheets **23**
- Property Sheets 24
- Selecting a boostxx.props sheet 25
- Set PRPD within the prop. sheets 27
  - Submodule : boost_submodules_location 27
  - Submodule : Boost_mod_root_location 27
  - Submodule : INtime_include 28
- Setting up for Third-Party Submodules 29
  - Using wolfSSL 29
  - Using zlib 30
  - Using websocketpp 31

Changing include paths 32
Changing preprocessor definitions and macros 33
Macros to disable warnings..... 34
Changing linker properties 35

Setting up for Boost Tests.....36

Setting up Boost Examples.....37

Testing your project.....38
Empty Project..... 38
Running Examples 38
Running Tests..... 38

Setting up Test Automation.....39
Editing the Test Commands..... 40

Figures

Figure 1: Context menu to **Clone…** button 7
Figure 2: ***Git clone - TortoiseGit*** configuration: 8
Figure 3: Context menu to **Switch/Checkout** button 9
Figure 4: Entering the repository version 10
Figure 5: Entering the Boost version update window..... 10
Figure 6: Entering Submodule Update..... 11
Figure 7: Entering Checked-out the repository..... 11
Figure 8: **Boost SDK** Download Page..... 12
Figure 9: Files in Boost172sdk-xxxxx-x.zip 12
Figure 10: **Boost Application Wizard** 14
Figure 11: **INtime Development Environment Configuration** 15
Figure 12: Types of INtime/Boost projects 17
Figure 13: **Place solution and project in same directory**..... 18
Figure 14: An empty project 19
Figure 15: Boost configuration..... 19
Figure 16: Options that affect the whole Boost process 20
Figure 17: Overview of this Boost project..... 20
Figure 18: The INtime Boost Solution Directory 22
Figure 19: Property Manager..... 25
Figure 20: Add Existing Property Sheet..... 26

Welcome! — Before You Begin

INtime Boost is a library that utilizes Boost headers but is configured to use INtime Libraries by overriding files and using project preprocessor definitions.

You can read all the available Boost features/submodules supported in INtime under the link: Boost Library - TenAsys.

This guide helps you with setting up Boost development environments along with the INtime® Software Development Kit.

Terms

- PRPD: project property definitions.
- .props sheets: Visual Studio Property Sheets are configuration files

Supported Boost Version

Boost Version	Comment
1.67	The Original Boost Repository April 16 2018 - version 1.67, is phased out as of INtime 7.
1.72	The Boost repository is supported in INtime 7.x.

Note: Many of the Boost library modules are actively maintained and improved, so backward compatibility with prior versions isn't always possible.

Supported C++ Version

INtime Boost supports cpp11, cpp17 and cpp20 features for INtime's cpp11, cpp17 and cpp20 libraries which may be used by VS2015 through VS2022.

Requirements

- INtime SDK
- Git is used to clone the Boost C++ libraries.

Third-Party Modules

You can add additional third-party modules.

Module	Description
wolfSSL	wolfSSL is a proprietary portable, C-language-based SSL/TLS library. See the link: wolfSSL – Embedded SSL/TLS Library
zclib	zlib is a widely-used, general-purpose, lossless data-compression library. See the link: GitHub - madler/zlib: A massively spiffy yet delicately unobtrusive compression library.
websocketpp	websocketpp is a high-performance C++ WebSocket client and server library. It supports RFC 6455. See the link: GitHub - zaphoyd/websocketpp: C++ websocket client/server library

Install Boost

Preferred Directory

To maintain a consistent directory structure, we recommend cloning all Git repositories into the preferred directory: C:\git.

We should ensure that the paths are correctly aligned. We will reference these repositories in Visual Studio props files.

Install Git

Download and install Git for Windows. You can choose your preferred tool. The example demonstrates here is with Tortoise Git.

Install Git from https://gitforwindows.org/.
Install Tortoise Git from https://tortoisegit.org/.

Clone the Boost Repository

Clone the Boost repository from https://github.com/Boostorg/Boost.git to your choosing directory.

Using Tortoise Git

- Navigate to the preferred git directory.
- Right-click on the Boost folder where you want to clone the repository.
- From the context menu, select TortoiseGit and select **Git Clone....**
- Configure the Clone.

Figure 1: Context menu to **Clone…** button

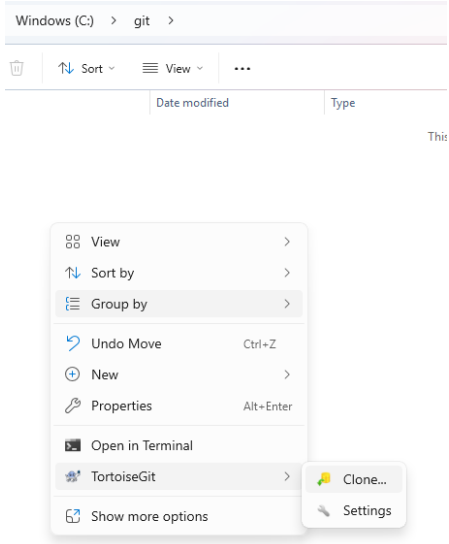
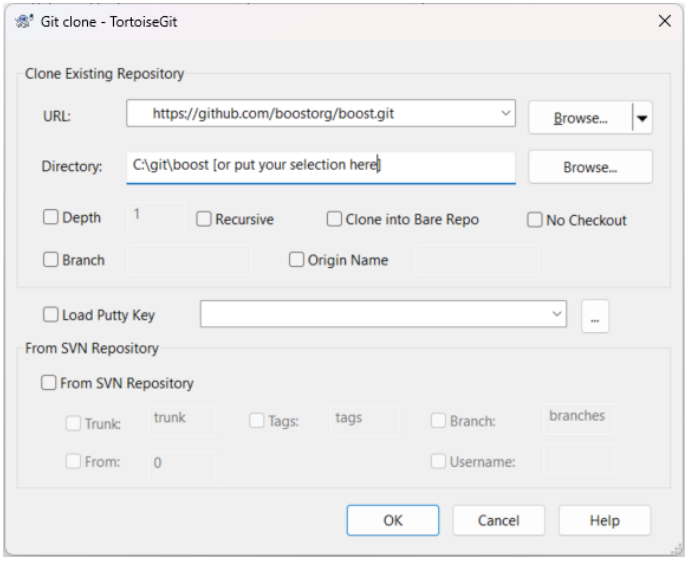


Figure 2: ***Git clone - TortoiseGit*** configuration:



Note*:* Check the link documentation for details: https://tortoisegit.org/docs/tortoisegit/

Using the command prompt

git clone --recursive https://github.com/Boostorg/Boost.git

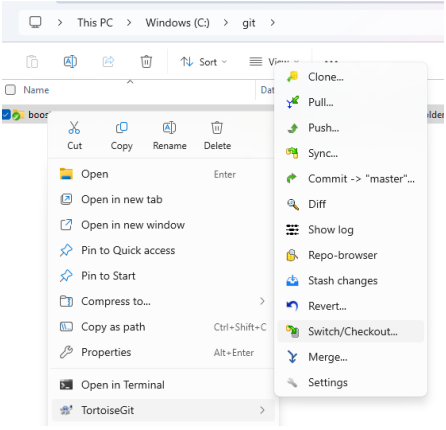
Check out the specific commit

We need the commit (4b0c5d768d645ded8f6a01ea4cf5a3ac0233df0a) to get the repository version.

Using Tortoise Git

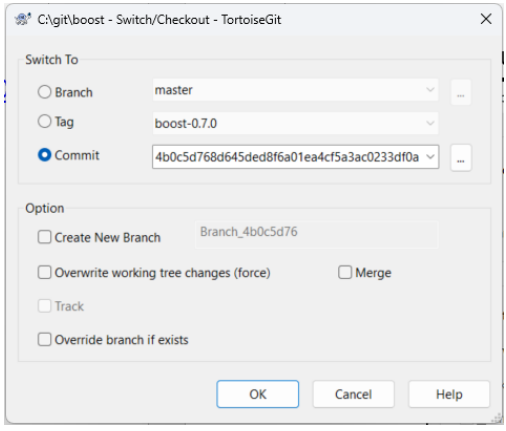
- Right-click on the folder where you want to clone the repository.
- From the context menu, select TortoiseGit and select **Switch/Checkout**

Figure 3: Context menu to **Switch/Checkout** button



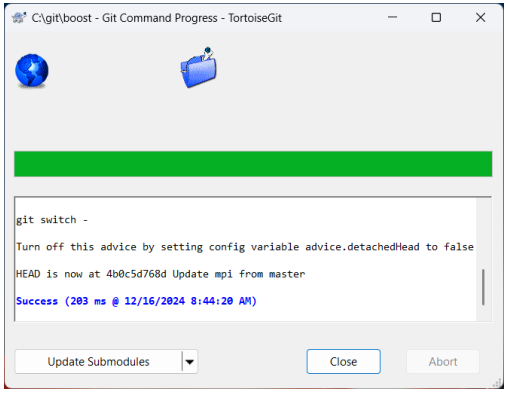
- Type the above commit value (4b0c….) to choose the repository version.

Figure 4: Entering the repository version



- Select **OK**.
- Select **Update Submodules**

Figure 5: Entering the Boost version update window



- Select Path: tool/check_build. Options: --init, Recursive, Force

Figure 6: Entering Submodule Update

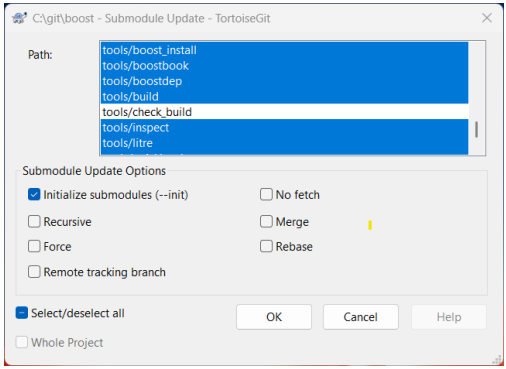
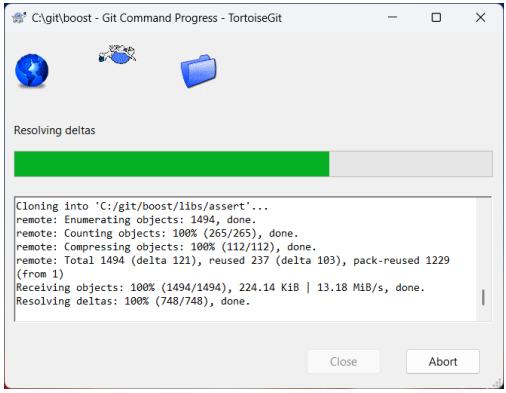


Figure 7: Entering Checked-out the repository



- Select **OK**.

Using the command prompt

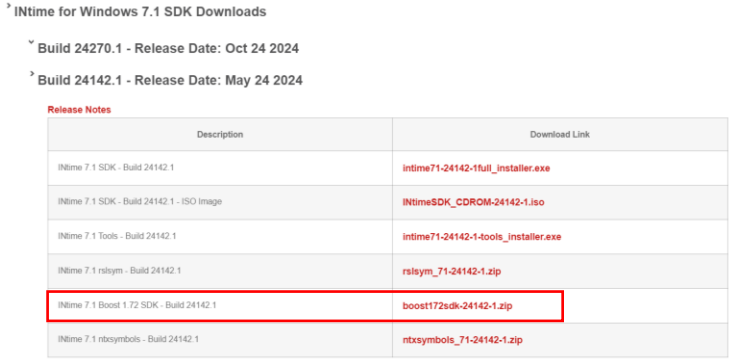
- cd <Boost folder>
- git checkout 4b0c5d768d645ded8f6a01ea4cf5a3ac0233df0a
- git submodule update --init --recursive -
  - ":(exclude)tool/check_build"

Install INtime Boost SDK

The Boost SDK provides the Boost application Wizard which sets the property manager, and project property definitions (PRPD) for you. You can select to configure for a Boost example or a Boost test.

As of intime7.x, the Boost SDK is available via the download page for each release of INtime. For example, Boost172sdk-24050-1.zip corresponds to Intime version 24050-1.

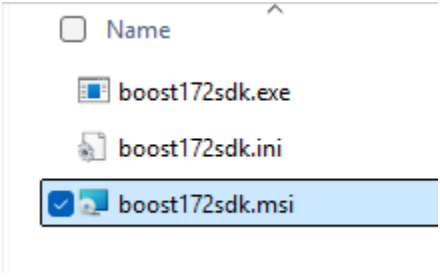
Figure 8: **Boost SDK** Download Page



Install Boost172sdk.msi

Unzip the Boost172sdk-xxxxx-x.zip. Install **Boost172sdk.msi**.

Figure 9: Files in Boost172sdk-xxxxx-x.zip



INtime Boost headers and libraries

The Boost headers and libraries are installed in the below location.

Files	Location
INtime Boost headers	**%INTIME%**[1]**rt\include**
INtime Boost libraries	**%INTIME%**[1]**rt\lib**

1 Typically C:\Program Files\INtime. On 64-bit versions of Windows this is C:\Program Files (x86)\INtime. Make note of this directory so you can locate it again if you wish to inspect header files and other INtime files.

IMPORTANT: Please refer to **Boost_release_notes.tx**t (located in the **%INTIME%help** directory) for the required Boost submodule version to download.

Next, add the commit version to the **Commit** field.
Finally, overwrite the tree changes.

Validating Boost Application Wizard in Visual Studio

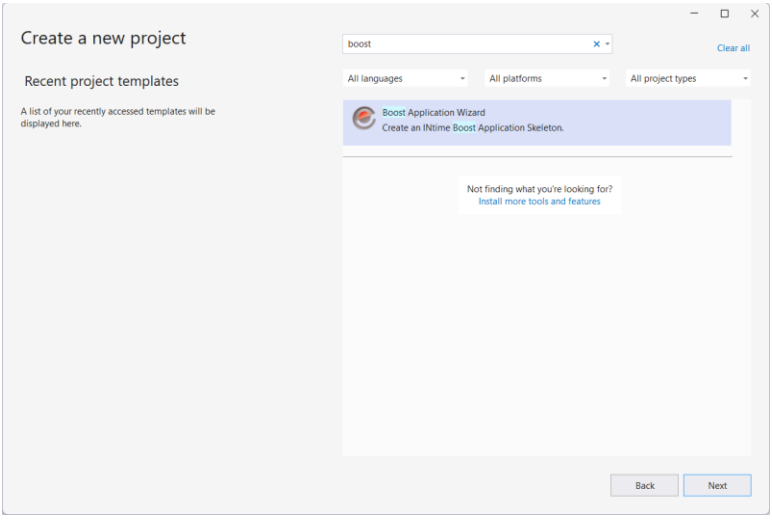
To ensure the Boost Application Wizard is enabled in Visual Studio, follow the steps below. This will help verify if the Boost installation was successful. If the wizard is not visible, reinstallation of the INtime Visual Studio development tools may be required.

Check for Boost Application Wizard

- Open Visual Studio.
- Click on File > New > Project.
- In the "Create a New Project" window, use the search bar to look for **Boost Application Wizard**.

If the Boost Application Wizard is listed, the installation was successful.

Figure 10: **Boost Application Wizard**

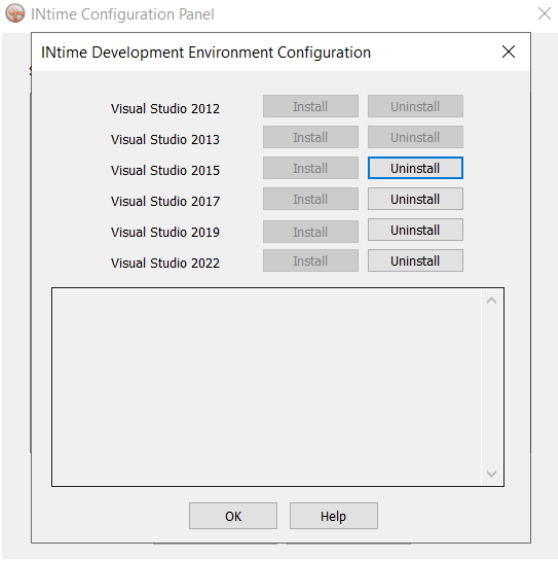


Troubleshooting Missing Boost Application Wizard

If the Boost Application Wizard is not visible, uninstall the platform tools for the visual studio version that you are using to develop and reinstall them. This will make sure that the wizards are installed in the correct locations.

- Open the **INtime Configuration Panel**.
- Click the **INtime Visual Studio Development Tools** option.
- Unistall the selected INtime Visual Studio Development Tools.
- Reinstall the selected one.

Figure 11: **INtime Development Environment Configuration**



Now, restart **Visual Studio** to ensure all changes take effect after reinstallation.

Note: If the Boost Application Wizard still does not appear, check that all dependencies that are required for INtime and Boost are installed. Contact the TenAsys support team.

Setting Up a Visual Studio Boost Project with Property Manager

Since the Boost project is vast, it is more efficient to use Property Manager and project property sheets to set up an INtime Boost project in Visual Studio. These property sheets provide **Project property definitions (PRPD)** for include directories, linker library directories, and linker input dependencies.

> **Note***:* Check the link documentation for details:
>
> Set C++ compiler and build properties in Visual Studio | Microsoft Learn
>
> Property inheritance in Visual Studio projects - C++ | Microsoft Learn

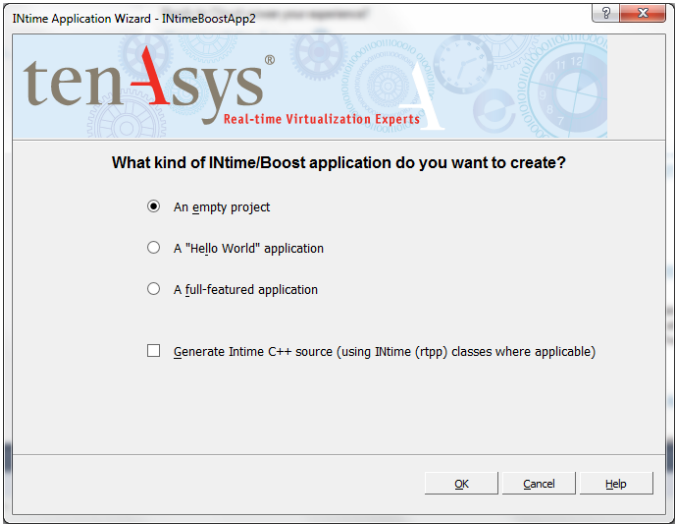
Configure an INtime Boost project

In Visual Studio, and INtime, you can use **Boost Application Wizard** to create an INtime Boost application. The wizard allows users to create three different types of INtime Boost projects:

- An empty project
- A “Hello World” application
- A full-featured application

There is also an option for enabling this option will use INtime C++ (rtpp) classes in the generated project where applicable.

Figure 12: Types of INtime/Boost projects

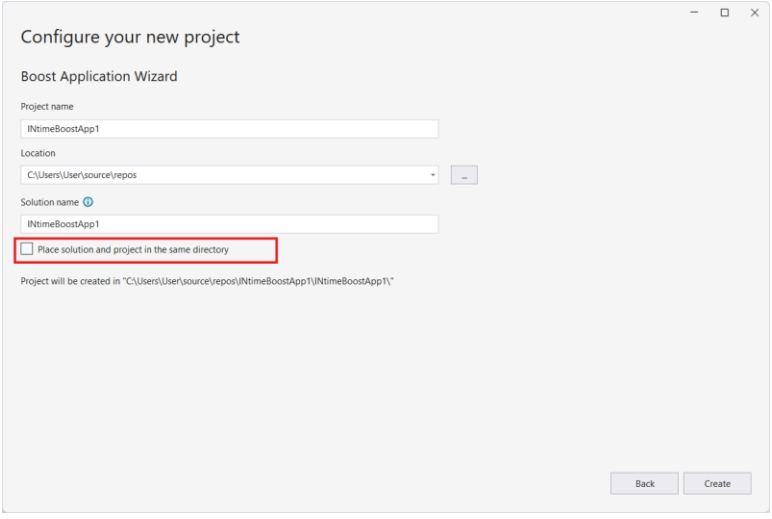


Creating an Empty Project

We will demonstrate creating an empty project as an example to walk through the Boost Application Wizard's continuing configurations.

- Open Visual Studio.
- Click on File > New > Project.
- In the "Create a New Project" window, use the search bar to look for **Boost Application Wizard**.
- Configure your new project. Make sure the checkbox **Place solution and project in same directory** is unchecked if running the supplied Boost tests and examples.

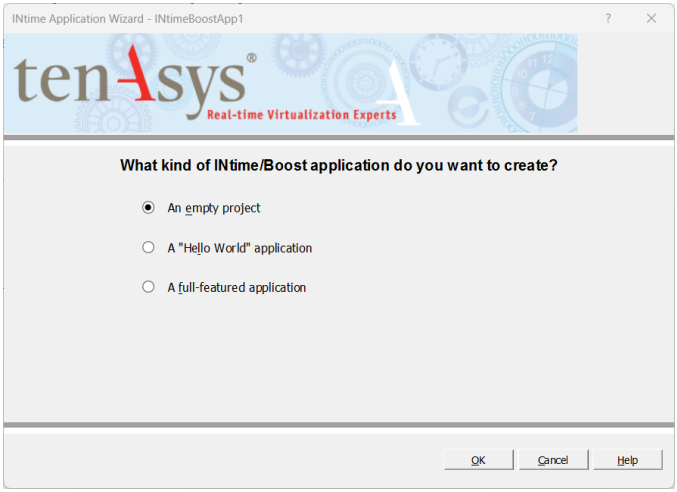
Figure 13: **Place solution and project in same directory**



IMPORTANT: If developing projects for testing, please uncheck the Place solution and project in same directory. The checkbox needs to be unchecked for this file to be found during runtime.

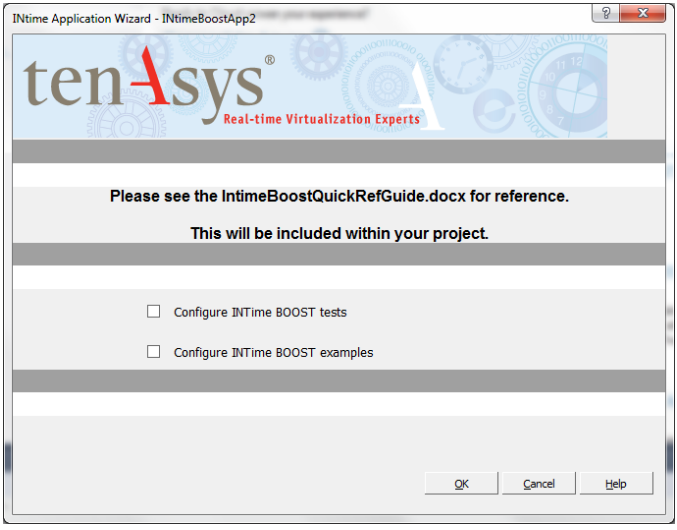
- Select **An empty project**
- Select **OK**.

Figure 14: An empty project



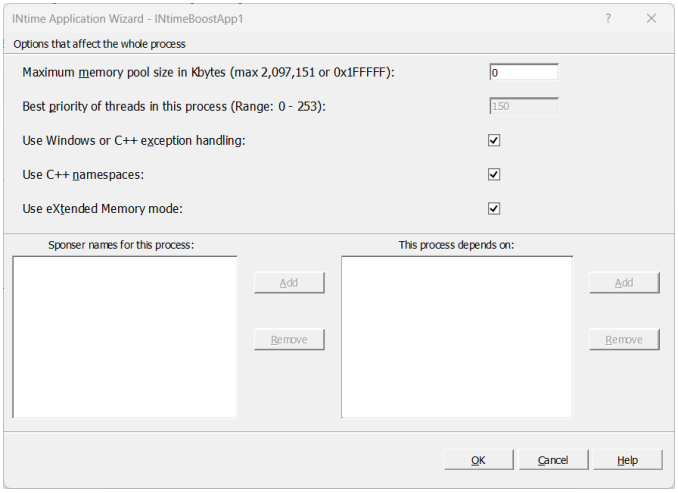
- Select Boost configuration for the testing purpose projects. If not, select **OK**.

Figure 15: Boost configuration



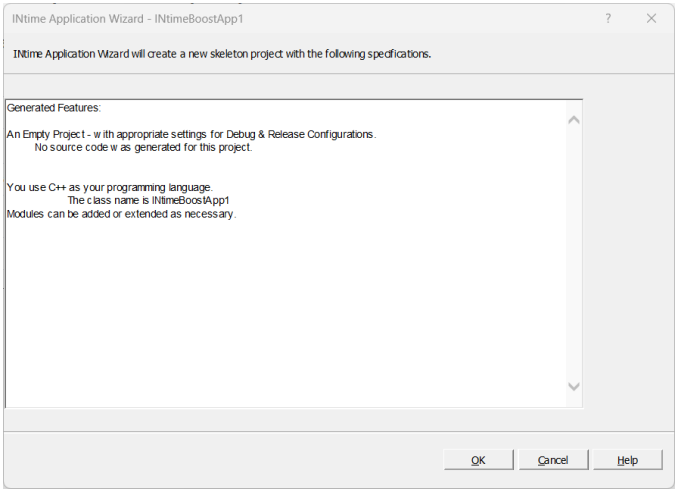
- Configure options that affect the whole Boost project.
- Select **OK**.

Figure 16: Options that affect the whole Boost process



- Check the overview of this project.
- Select **OK** to finish the wizard.

Figure 17: Overview of this Boost project



Testing Requirements: Installation_Paths.txt

The Boost projects for testing require files to run, which are dynamically read during execution. Those testing required files are read from **Installation_Paths.txt**. The file Installation_Paths.txt is located in the **support_folder**[1] located in the solution directory.

[1]Typically C:\Program Files\INtime\ vstudioxxx\wizards\boostprojects\support_folder. The folder vstudioxxx can be vstudio140, vstudio150, vstudio160, or vstudio170.

We will use the file to find the path of runtime files used by the tests. To use this the solution and project must be in separate directories.

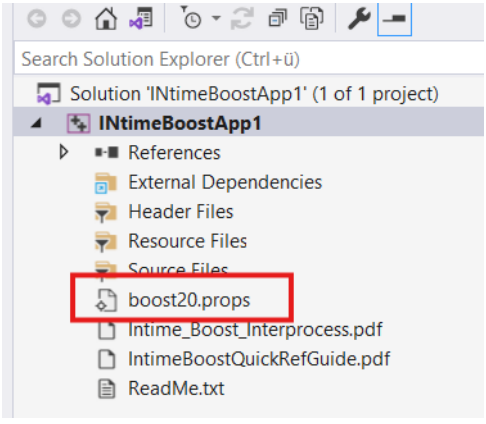
BOOST_EXAMPLES::C:/BoostTest/Boost_examples, or where you placed the examples.

BOOST_TESTS::C:/BoostTest/Boost_tests, or where you placed the tests.

The INtime Boost Solution Directory Overview

The INtime Boost Solution directory contains a property sheet **boostxx.props**. It is specifically used to manage project-wide Boost settings. The boostxx.props file plays a key role in managing the Boost configurations efficiently across multiple project configurations. You can reuse the property sheet for your other Boost projects in your development environment.

Figure 18: The INtime Boost Solution Directory



Note: The boostxx.props file will be described in the chapter Property Sheets.

Visual Studio Property Sheets

Visual Studio Property Sheets are configuration files (. props sheets) used in Visual Studio to manage and share build settings, properties, and configurations across multiple projects or solutions.

Since the Boost project is vast, we use property sheets to configure the Project property definitions (PRPD).

Property Sheets

The Boost wizard adds the following property sheets to the project.

PRPD project **property** definitions:

props for (PRPD)	Description
boostlib.props	This sheet sets to build the cpp11 - VS2015 project. Typically in C:\Program Files (x86)\INtime\vstudio140\wizards\boostprojects
boost17.props	This sheet sets to build the cpp17 - VS2017 project. Typically in C:\Program Files (x86)\INtime\vstudio150\wizards\boostprojects
boost20.props	This sheet sets to build the cpp20 - VS2019 project or VS2022 project. Typically for VS2019 in C:\Program Files (x86)\INtime\vstudio10\wizards\boostprojects Typically for VS2022 in C:\Program Files (x86)\INtime\vstudio170\wizards\boostprojects
boostlibtest.props	This sheet sets the (PRPD) for the test directories of INtime Boost. Typically in C:\Program Files (x86)\INtime\vstudioxx\wizards\boostprojects
boostlibExample.props	This sheet sets the (PRPD) for the example directories of Intime. Typically in C:\Program Files (x86)\INtime\vstudioxx\wizards\boostprojects

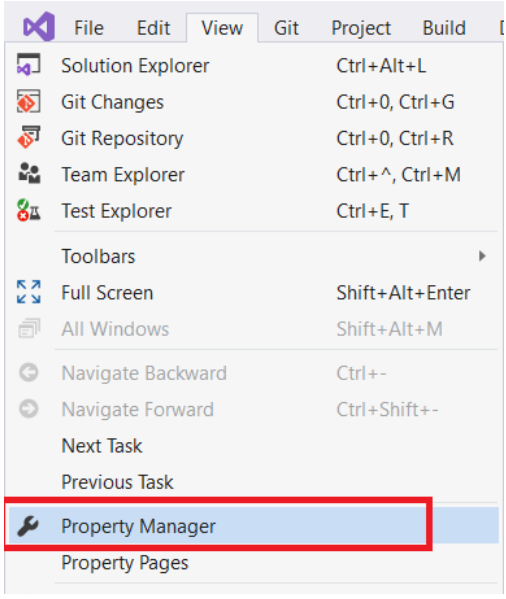
Selecting a boostxx.props sheet

If you want to configure a universal boostxx.props in your development environment, you can delete your local boostxx.props in your project and use the one from the **INtimeInstallDirectory**.

> **Note**: **INtimeInstallDirectory** is usually at C:\Program Files (x86)\INtime, and **vstudioVersion** for visual studio 2015 is vstudio140. Verify that steps 6 and 7 are what you expect and make changes when necessary.

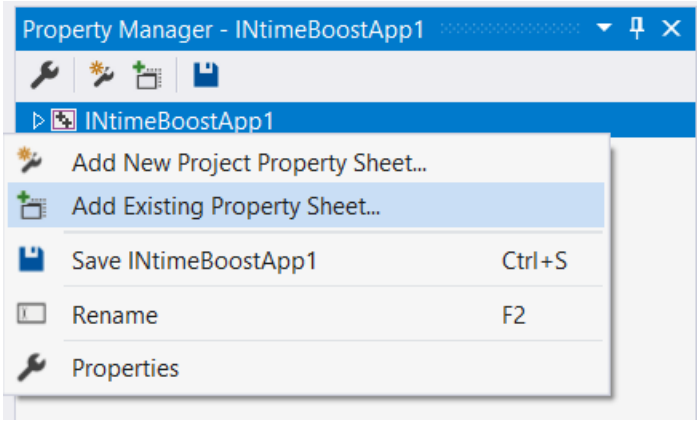
Within Visual Studio, click the **View** tab and then click **Property Manager**.

Figure 19: Property Manager



Right-click on the INtime Boost project in Property Manager. Click the button **Add Existing Property Sheet**. Select the **boostxx.props**. Check the section Property Sheets.

Figure 20: Add Existing Property Sheet



Set PRPD within the prop. sheets

After the property manager is set up, you may want to update the default settings. You can modify the **submodule** installation location within this placeholder.

Submodule : boost_submodules_location

You can edit the value in Boostlib.props / Boost17.props / Boost20.props. This is the location where you downloaded Boost including the \lib, so add the whole path there including the \lib. The preferred directory is C:\git\Boost\libs.

```
<!-- USER UPDATES HERE -->
<!-- Boost's Submodules Download Location -->
<!-- Put_boost_submodules_location_here -->
<boost_submodules_location>
[Put_Boost_submodules_location_here]
</boost_submodules_location>
```

Submodule : Boost_mod_root_location

If necessary, modify the INtime header locations for the INtime Boost here.

```
<boost_mod_root_location>$(Intime)\rt\include\boost
</boost_mod_root_location>
```

Submodule : INtime_include

If necessary, modify the INtime header locations here.

For Boostlib.props / cpp11 / VS2015:

```
<intime_include>
    $(Intime)\rt\include\cpp11;
    $(Intime)\rt\include\boost;
    $(intime_boost_root_path)\sys;
   $(wolfssl_location);
   $(zlib_location);
    $(Intime)\rt\include\network7;
    $(Intime)\rt\include\network7\sys;
    $(Intime)\rt\include;
</intime_include>
```

For Boost17.props / cpp17 / VS2017:

```
<intime_include>
    $(Intime)\rt\include\cpp17;
    $(Intime)\rt\include\boost;
    $(intime_boost_root_path)\sys;
   $(wolfssl_location);
   $(zlib_location);
    $(Intime)\rt\include\network7;
    $(Intime)\rt\include\network7\sys;
    $(Intime)\rt\include;
</intime_include>
```

For Boost20.props / cpp20 / VS2019 / VS2022:

```
<intime_include>
    $(Intime)\rt\include\cpp20;
    $(Intime)\rt\include\boost;
    $(intime_boost_root_path)\sys;
   $(wolfssl_location);
   $(zlib_location);
    $(Intime)\rt\include\network7;
    $(Intime)\rt\include\network7\sys;
    $(Intime)\rt\include;
</intime_include>
```

Setting up for Third-Party Submodules

Using wolfSSL

If the developer would like to use wolfSSL in their project, they must first contact wolfSSL to purchase a license. After obtaining the license, the following properties in the boostxx.props file must be updated.

```
<!-- If using wolf ssl uncomment these areas -->
<wolfssl_location>
    <!--
    C:\Intime\rt\include\wolfssl551;
    C:\Intime\rt\include\wolfssl551\wolfssl
    -->
</wolfssl_location>
<wolfssl_macros>
    <!--
    BOOST_ASIO_USE_WOLFSSL;
    WOLFSSL_ASIO;
    INTIME_RTOS;
    OPENSSL_EXTRA;
    OPENSSL_ALL;
    WOLFSSL_ALLOW_SSLV3;
    -->
</wolfssl_macros>
<wolfssl_lib>
    <!--
    libwolfssl551.lib
    -->
</wolfssl_lib>
```

Using zlib

If the developer would like to use zlib in their project they should first download the zlib repository from https://github.com/madler/zlib.git. We provide the library and is available in $(Intime)\rt\lib. The developer may then update following properties in the boostxx.props file.

```
<zlib_location>
    <!--
      C:\Intime\rt\include\zlib;
    -->
    <!--or-->
    <!--
      $(Intime)\rt\include\zlib;
    -->
</zlib_location>
<zlib_lib>
    <!--
    zlibstaticd.lib;
    -->
</zlib_lib>
```

Using websocketpp

If the developer would like to use websocketpp in their project, they should first download the websocketpp repository from https://github.com/zaphoyd/websocketpp.githttps://github.com/madler/zlib.git. Then create a websocketpp.prop and attach to the project.

```
<!-- Websocketpp's Download Location -->
<websocketpp_location>C:\git\websocketpp</websocket
pp_location>
<!-- Websocketpp Intime Modifications-->
<websocketpp_mod_root_location>C:\INtime\Components
\websocketpp\IntimeTest\logging_config;C:\INtime\Co
mponents\websocketpp</websocketpp_mod_root_location
>
<!-- Websocketpp Include Paths-->
<websocketpp_include_paths>$(websocketpp_mod_root_l
ocation);$(websocketpp_location);</websocketpp_incl
ude_paths>
```

Changing include paths

In **C++ General** group modify the **Additional Include Directories**:
Note since Boost overrides everything else, it's important to have $(Boost_lib_include_paths) as the first entry, so that it takes precedence. Otherwise, overrides could be missed and may result in compile errors. The developer may have to add additional directories for their project. In this case it is recommended to add them at the tail of the string. This is represented as $(other); below.

For generic Boost:

```
<!-- Boost Lib Include Paths-->
<boost_lib_include_paths>
$(intime_boost_windows_include_path);$(boost_modifi
cations_include_path);$(boost_include);$(libcxx_inc
lude);$(intime_include);$(pthreads_root)$(jpeg_incl
ude_path);$(submodule_spirit_path);$(other);
</boost_lib_include_paths>
```

For Boost Tests:

```
<boost_test_include_directories>
$(root_test_location);$(boost_test_include);$(tests
_unit_test_modifications_include_path);$(tests_unit
_test_submodule_headers);$(other);
</boost_test_include_directories>
```

For Boost Examples:

```
<boost_example_include_directories>
$(root_example_location);$(boost_example_include);$
(examples_unit_test_modifications_include_path);$(e
xamples_unit_test_submodule_headers);$(boost_lib_in
clude_paths);$(other);
</boost_example_include_directories>
```

Changing preprocessor definitions and macros

Within Visual Studio, click the View tab and then click **Solution Explorer**. Right-click on the INtime Boost project in Solution Explorer. Click the button **Properties**. In **C++** > **General** group, modify the **Preprocessor > Preprocessor Definitions**, you will at least want to have:

For Boostlib.props / cpp11 / VS2015:

```
__INTIME_BOOST__;__INTIME_CPP11;__INTIME_USE_IOCP_
_;__INTIME_PTHREAD__;_WIN32;_WIN32_WINNT=0x0501;_HA
S_NAMESPACE;_USRDLL;_USE_64BIT_TIME_T;__INTIME_TE
ST_TOOLS;_BOOST_GIL_NO_TIFF_LIB;BOOST_GIL_NO_PNG
_LIB;BOOST_THREAD_USES_CHRONO;BOOST_THREAD_TH
ROW_IF_PRECONDITION_NOT_SATISFIED;BOOST_USE_OW
N_EXCEPTIONS;$(wolfssl_macros);%(PreprocessorDefinitions)
```

For Boost17.props / cpp17 / VS2017:

```
__INTIME_BOOST__;__INTIME_CPP17;__INTIME_USE_IOCP_
_;__INTIME_PTHREAD__;_WIN32;_WIN32_WINNT=0x0501;_HA
S_NAMESPACE;_USRDLL;_USE_64BIT_TIME_T;__INTIME_TE
ST_TOOLS;BOOST_GIL_NO_TIFF_LIB;BOOST_GIL_NO_PNG_
LIB;BOOST_THREAD_USES_CHRONO;BOOST_THREAD_THR
OW_IF_PRECONDITION_NOT_SATISFIED;BOOST_USE_OWN
_EXCEPTIONS;SILENCE_ALL_CXX17_DEPRECATION_WARNI
NGS;_SILENCE_BOGUS_WARNINGS;__BYPASS_ITERATOR_
DEBUG_LEVEL_2__;$(wolfssl_macros);%(PreprocessorDefinition
s)
```

For Boost20.props / cpp20 / VS2019 / VS2022:

```
__INTIME_BOOST__;__INTIME_CPP20;__INTIME_USE_IOCP_
_;__INTIME_PTHREAD__;WIN32;_WIN32_WINNT=0x0501;_HAS
_NAMESPACE;_USRDLL;_USE_64BIT_TIME_T;__INTIME_TES
T_TOOLS;BOOST_GIL_NO_TIFF_LIB;BOOST_GIL_NO_PNG_LI
B;BOOST_THREAD_USES_CHRONO;BOOST_THREAD_THRO
W_IF_PRECONDITION_NOT_SATISFIED;BOOST_USE_OWN_E
XCEPTIONS;_SILENCE_ALL_CXX17_DEPRECATION_WARNIN
GS;_SILENCE_BOGUS_WARNINGS;__BYPASS_ITERATOR_D
EBUG_LEVEL_2__;$(wolfssl_macros);%(PreprocessorDefinitions)
```

Macros to disable warnings

Since Boost has many uninitialized variables, and some deprecated items that the developer may still want the use. A few macros are added to disable some of these warnings:

SILENCE_ALL_CXX17_DEPRECATION_WARNINGS and _SILENCE_BOGUS_WARNINGS;

<Boost/config.hpp> has the _SILENCE_BOGUS_WARNINGS and disables the following warnings:
#pragma warning(disable : 4018) // '<': signed/unsigned mismatch
#pragma warning(disable : 4100) // 'variable_name': unreferenced formal parameter
#pragma warning(disable : 4189) // 'variable name': local variable is initialized but not referenced
#pragma warning(disable : 4456) // declaration of 'variable name' hides previous local declaration
#pragma warning(disable : 4458) // declaration of 'variable name' hides class member
#pragma warning(disable : 4459) // declaration of 'variable name' hides global declaration
#pragma warning(disable : 4503) // 'variable name': decorated name length exceeded, name was truncated
#pragma warning(disable : 4702) // unreachable code
#pragma warning(disable : 4706) // assignment within conditional expression
#pragma warning(disable : 4709) // comma operator within arr index expression
#pragma warning(disable : 4714) // marked as __forceinline not inlined
#pragma warning(disable : 4718) // recursive call has no side effects, deleting
#pragma warning(disable : 4800) // unsigned int': forcing value to bool 'true' or 'false' (performance warning)
#pragma warning(disable : 4834) // discarding return value of function with 'nodiscard' attribute
#pragma warning(disable : 4326) // return type of 'main' should be 'int' instead of 'void'
#pragma warning(disable : 4700) // uninitialized local'variable name' used
#pragma warning(disable : 4701) // potentially uninitialized local variable

Changing linker properties

In **Linker General** group modify the **Additional Library Directories**, you will at least want to have:

For generic Boost:
```
;$(boost_lib_directories);
```

For Boost Tests:
```
;$(boost_test_dependencies);
```

For Boost Examples:
```
;$(boost_example_dependencies);
```

> **IMPORTANT**: Once there is an linkage error, please recheck the .prop sheets if the paths are correctly aligned.

Setting up for Boost Tests

The wizard attempts to setup most of the project for tests for you automatically, however, there are a few settings that need to be manually configured. Note the tests will be provided as requested.

Update the Installation_Paths.txt file BOOST_TESTS::C:/Boost_tests to the location you chose to extract them. This text file is used to allow for the project to run any test from any location. Set Project Property Definitions (PRPD) within the Boostlibtest.props property sheets. Modify your test location with this placeholder.

```
<Otherwise>
<!-- USER UPDATES HERE -->
<PropertyGroup Label="UserMacros">
<!-- Location where tests were installed -->
<root_test_location>'Put where you installed the
tests here'</root_test_location>
<!-- Location where unit test headers were
installed -->
<tests_root_unit_test_location>$(RELEASED_UNIT_TEST
_LIB)</tests_root_unit_test_location>
<tests_unit_test_modifications_location>$(tests_roo
t_unit_test_location)\intime_modifications</tests_u
nit_test_modifications_location>
</PropertyGroup>
</Otherwise>
```

Setting up Boost Examples

The wizard attempts to set up most of project examples for you automatically, however, there are a few settings that need to be manually configured. Note the examples will be provided as requested.

Update the Installation_Paths.txt file:

BOOST_EXAMPLES::C:/Boost_examples to the location you chose to extract them. This text file is used to allow for the project to run any example from any location. Set Project Property Definitions (PRPD) within the BoostlibExample.props property sheets. Modify your example location with this placeholder.

```
<Otherwise>
<!-- USER UPDATES HERE -->
<PropertyGroup Label="UserMacros">
<!-- Location where examples were installed -->
<root_example_location>'Put where you installed the
example here'</root_example_location>
<!-- Location where unit test headers were
installed -->
<examples_root_unit_test_location>$(RELEASED_UNIT_T
EST_LIB)</examples_root_unit_test_location>
<examples_unit_test_modifications_location>$(exampl
es_root_unit_test_location)\intime_modifications</e
xamples_unit_test_modifications_location>
</PropertyGroup>
</Otherwise>
```

Testing your project

After the setup is complete you may want to test your project. This may just be deciding if the project builds or if you would like to run the examples or the tests.

Empty Project

Recommend using an empty project. Add a blank .cpp file to the project.

Running Examples

Copy one of the examples into the .cpp file. Build the project and run it.

Running Tests

Copy one of the tests into the .cpp file. Build the project and run it.

Setting up Test Automation

The wizard Install an automation batch script to run tests automatically. However, this needs to be configured. Check the section: Testing Requirements: Installation_Paths.txt

Batch script	Description
TestOne.cmd / TestOnexx.cmd[1]	test one test case
TestOneWNodeReset.cmd/ TestOneWNodeResetxx.cmd[2]	
testdir.cmd	
testall.cmd	

1 Boost17 files have a TestOne17.cmd, and Boost20 files have TestOne20.cmd.

2 Boost17 files have a TestOneWNodeReset17.cmd, and Boost20 files have TestOneWNodeReset20.cmd.

- Use **set solutionName=** to give the solution name you chose with the wizard.
- Use **set projectName=** to give the project name you chose with the wizard.
- Use **set cppFile=** to add the file name that you added to the empty project

Editing the Test Commands

Edit the testall.cmd by changing the testdir parameter to where the installer placed the files. For example, set testdir=C:/Boost_tests/

Open a visual studio command prompt. i.e. Developer Command Prompt for VS2015. Set current directory to your project location. for example:

cd C:\Users\<User>\Documents\Visual Studio 2015\Projects\ INtimeBoostApp1>

Set this at the command prompt:

call testall.cmd C:\Boost_tests > C:\INtime\Tests\Boostlibtest\all_test_results.txt

Where C:\Boost_tests could be where you installed the examples or tests.

You will see the output log file which is called C:\INtime\Tests\Boostlibtest\all_test_results.txtis.

You may also use call testall_custom.cmd, to test a feature.

You may test a directory with call testdir.cmd

C:\Boost_tests\sub_directory > C:\INtime\Tests\Boostlibtest\sub_directory_results.txt.