Boost Development Tools Quick Reference Guide

tenAsys
Embedded Virtualization Solutions

1 Install Boost

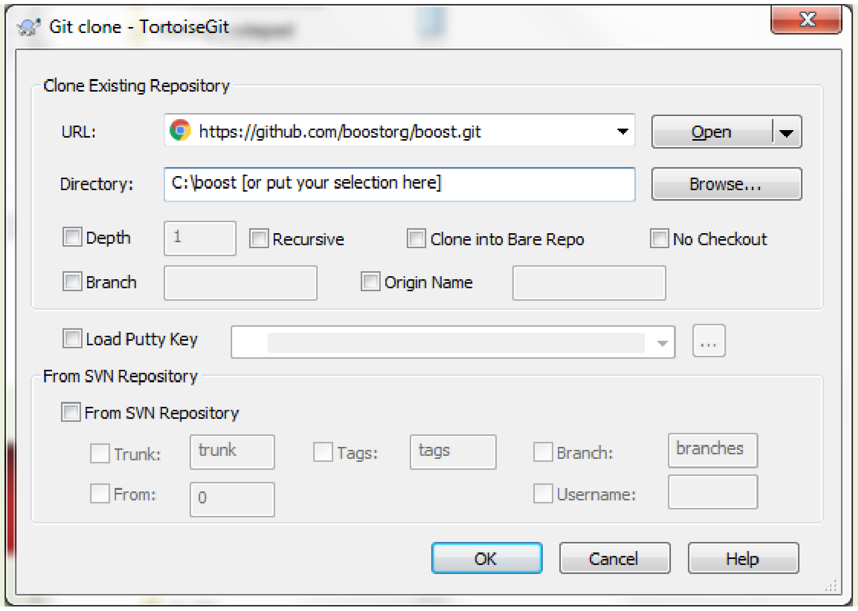
"INtime Boost is a library that utilizes Boost headers but is configured to use INtime Libraries by overriding files and using project preprocessor definitions.

1. Install Git

- You can use your own favorite, but this example shows Tortoise Git.
- Install Tortoise Git from https://tortoisegit.org/

2. Clone Boost from the repository

- Clone boost from https://github.com/boostorg/boost.git choosing your directory.



Git clone - TortoiseGit

Clone Existing Repository

URL: https://github.com/boostorg/boost.git

Directory: C:\boost [or put your selection here]

☐ Depth 1 ☐ Recursive ☐ Clone into Bare Repo ☐ No Checkout

☐ Branch ☐ Origin Name

☐ Load Putty Key

From SVN Repository

☐ From SVN Repository

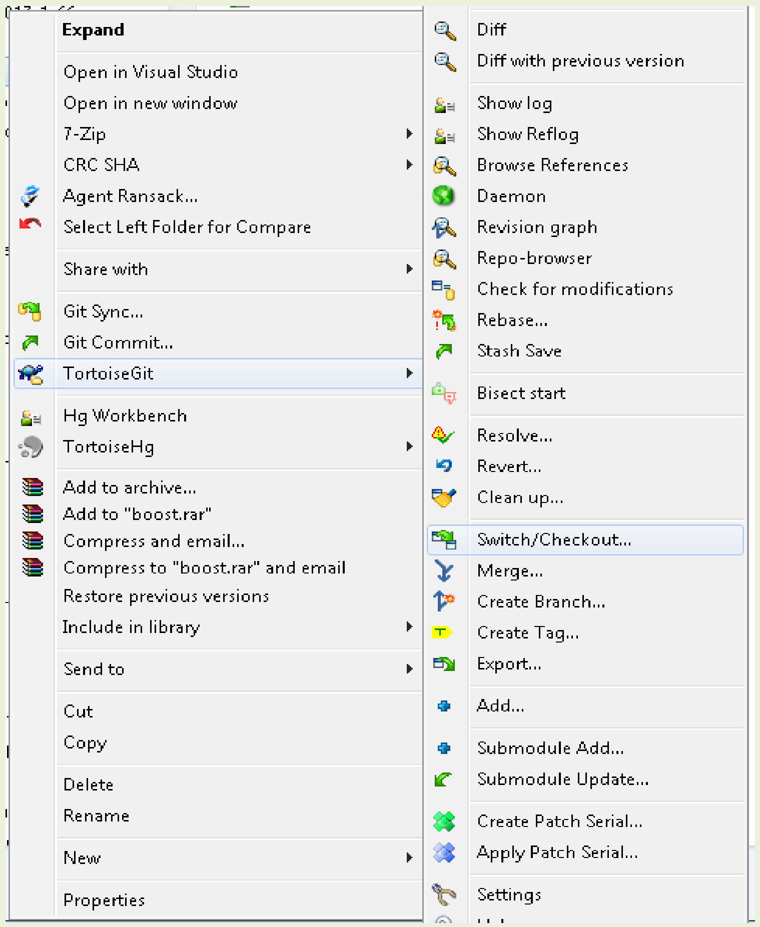
☐ Trunk: trunk ☐ Tags: tags ☐ Branch: branches

☐ From: 0 ☐ Username:

2 Boost Submodules

Cloning the project doesn't get the source from boost, we must update the submodules. Right click on the download directory and select the following.

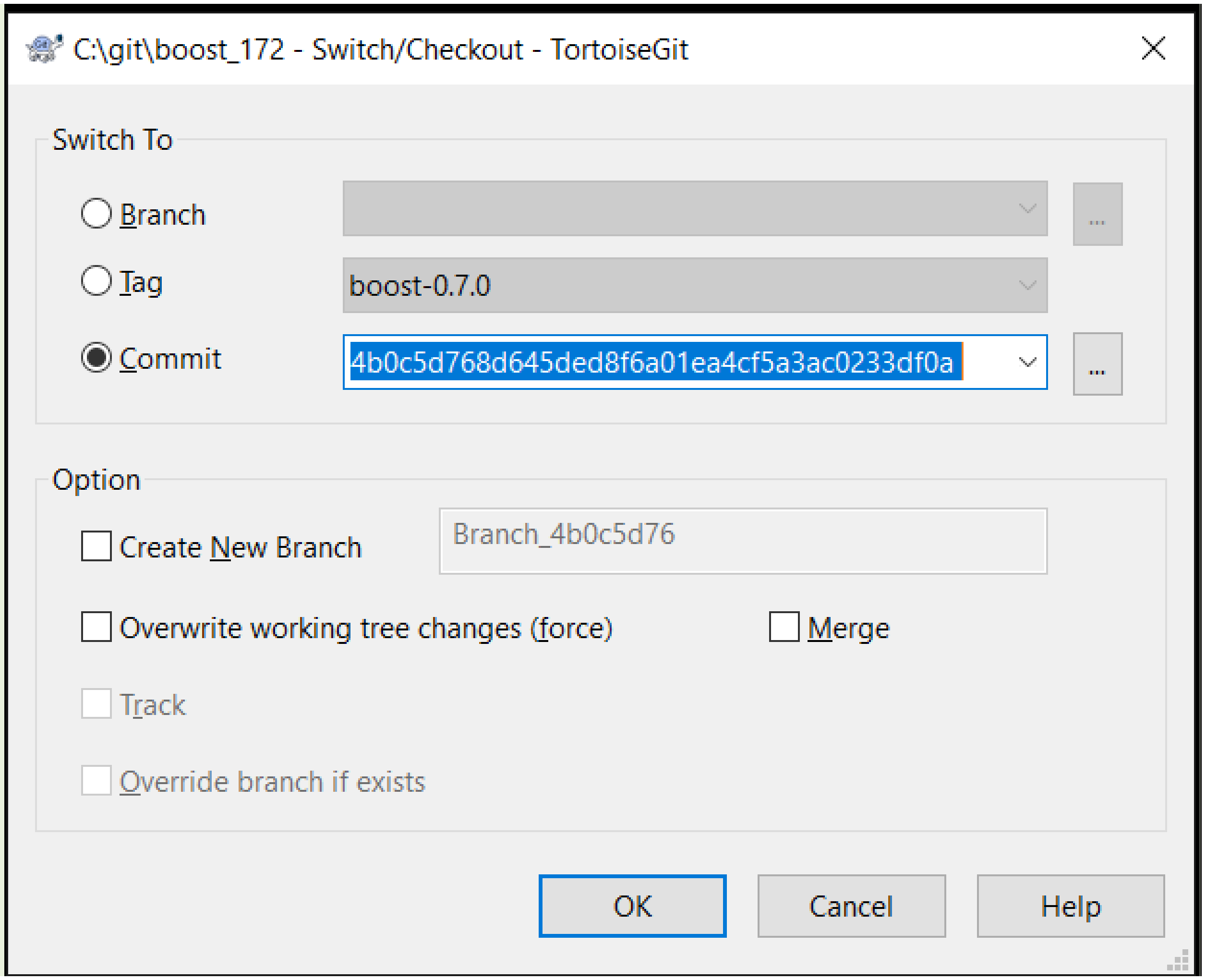
Update Using Git
Select:
TortoiseGit
Switch/Checkout



Please see Boost_release_notes.txt (located in the INtime\help directory) for the boost submodule version to download.

Then add the Commit Version to the Commit Field Then Overwrite Tree Changes.

Then click on
Update Submodules.



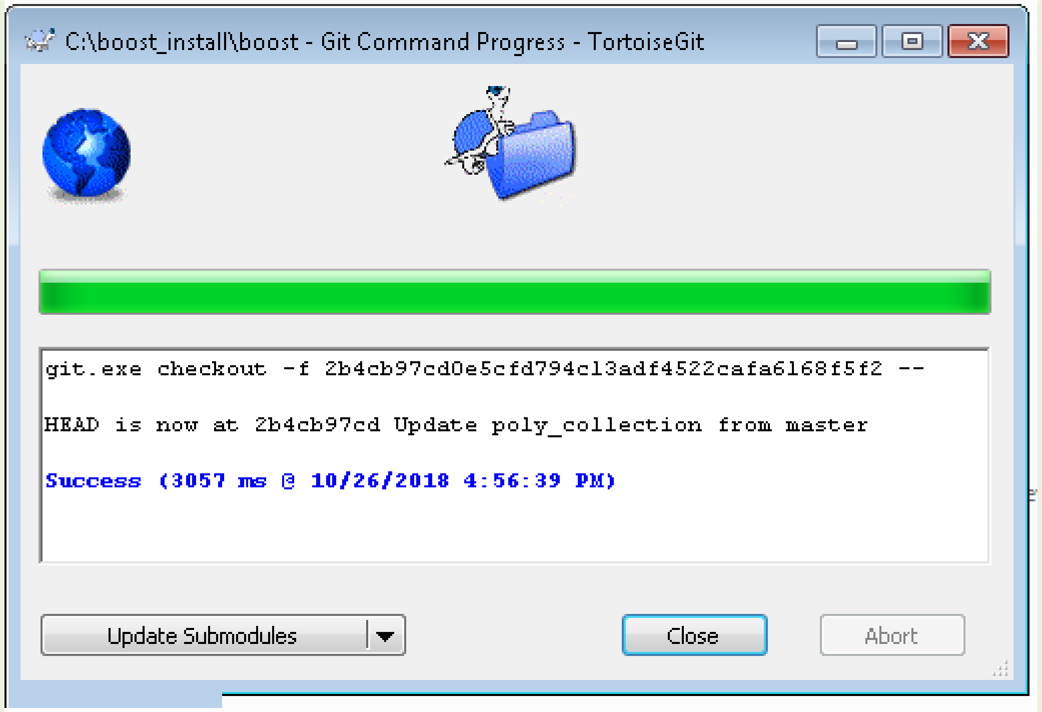
C:\git\boost_172 - Switch/Checkout - TortoiseGit

Switch To

- ( ) Branch
- ( ) Tag: boost-0.7.0
- (●) Commit: 4b0c5d768d645ded8f6a01ea4cf5a3ac0233df0a

Option

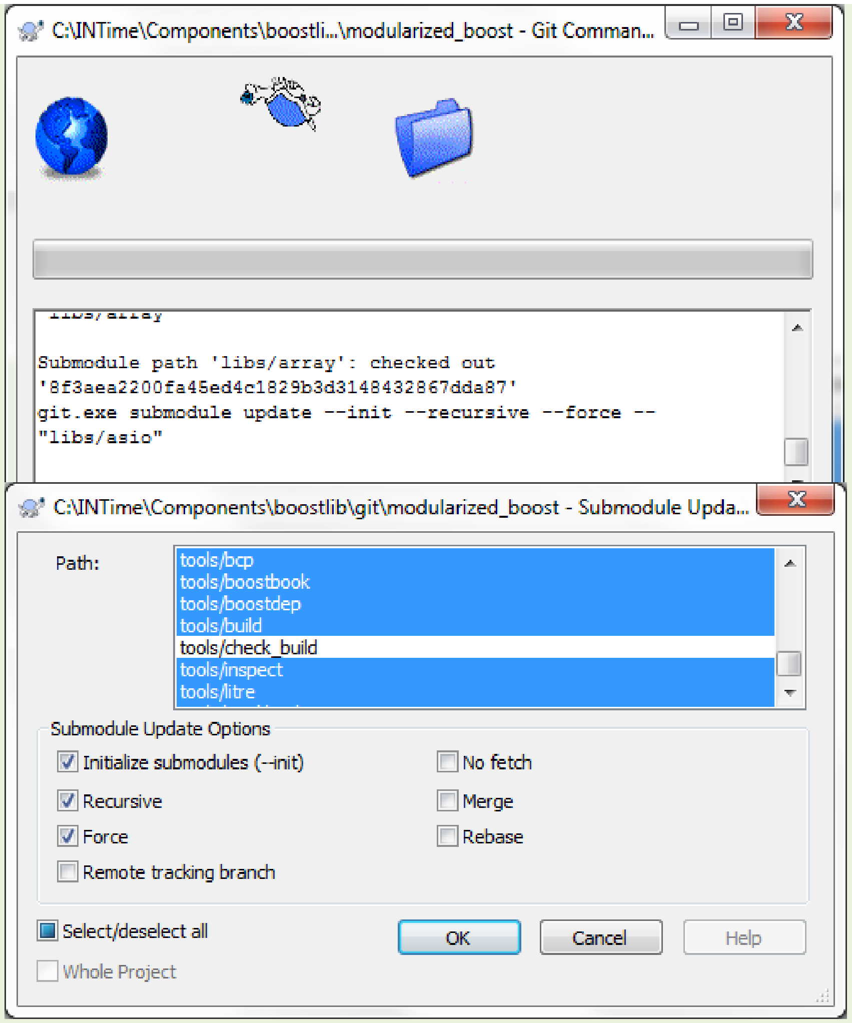
- ☐ Create New Branch: Branch_4b0c5d76
- ☐ Overwrite working tree changes (force)
- ☐ Merge
- ☐ Track
- ☐ Override branch if exists



```
git.exe checkout -f 2b4cb97cd0e5cfd794c13adf4522cafa6168f5f2 --

HEAD is now at 2b4cb97cd Update poly_collection from master

Success (3057 ms @ 10/26/2018 4:56:39 PM)
```


C:\INTime\Components\boostli...\modularized_boost - Git Comman...

```
libs/array

Submodule path 'libs/array': checked out
'8f3aea2200fa45ed4c1829b3d3148432867dda87'
git.exe submodule update --init --recursive --force --
"libs/asio"
```

C:\INTime\Components\boostlib\git\modularized_boost - Submodule Upda...

Path:
- tools/bcp
- tools/boostbook
- tools/boostdep
- tools/build
- tools/check_build
- tools/inspect
- tools/litre

Submodule Update Options

- [✓] Initialize submodules (--init)
- [✓] Recursive
- [✓] Force
- [ ] Remote tracking branch
- [ ] No fetch
- [ ] Merge
- [ ] Rebase

Select/deselect all

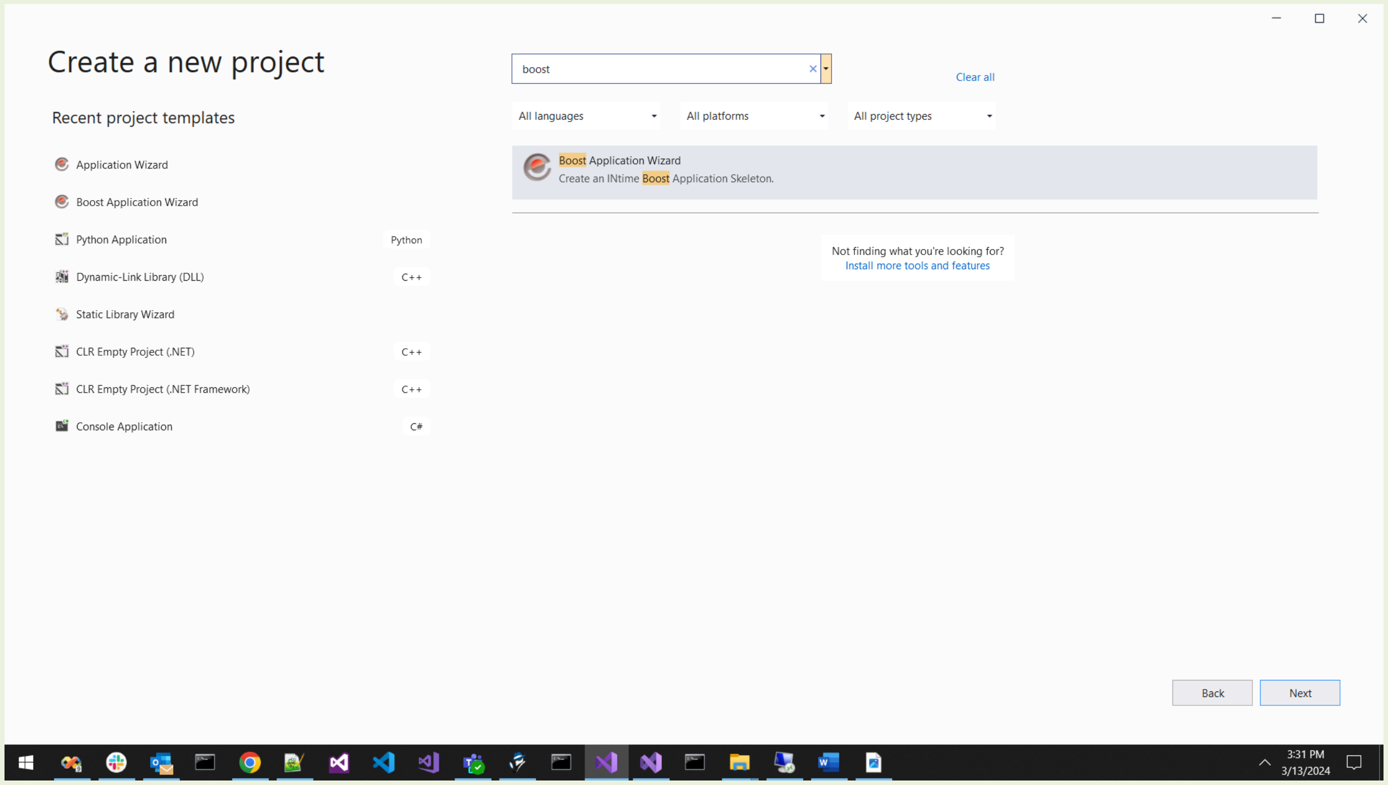
Whole Project

3 Install Intime's Boost SDK

As of intime7.0, 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.
Boost167sdk is phased out as of Intime 7.

a. **Work around for missing application wizard after installing boost.**

If the boost application wizard doesn't show up as shown in the following picture. Please 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.



Create a new project

Recent project templates

Application Wizard	
Boost Application Wizard	
Python Application	Python
Dynamic-Link Library (DLL)	C++
Static Library Wizard	
CLR Empty Project (.NET)	C++
CLR Empty Project (.NET Framework)	C++
Console Application	C#

boost

Clear all

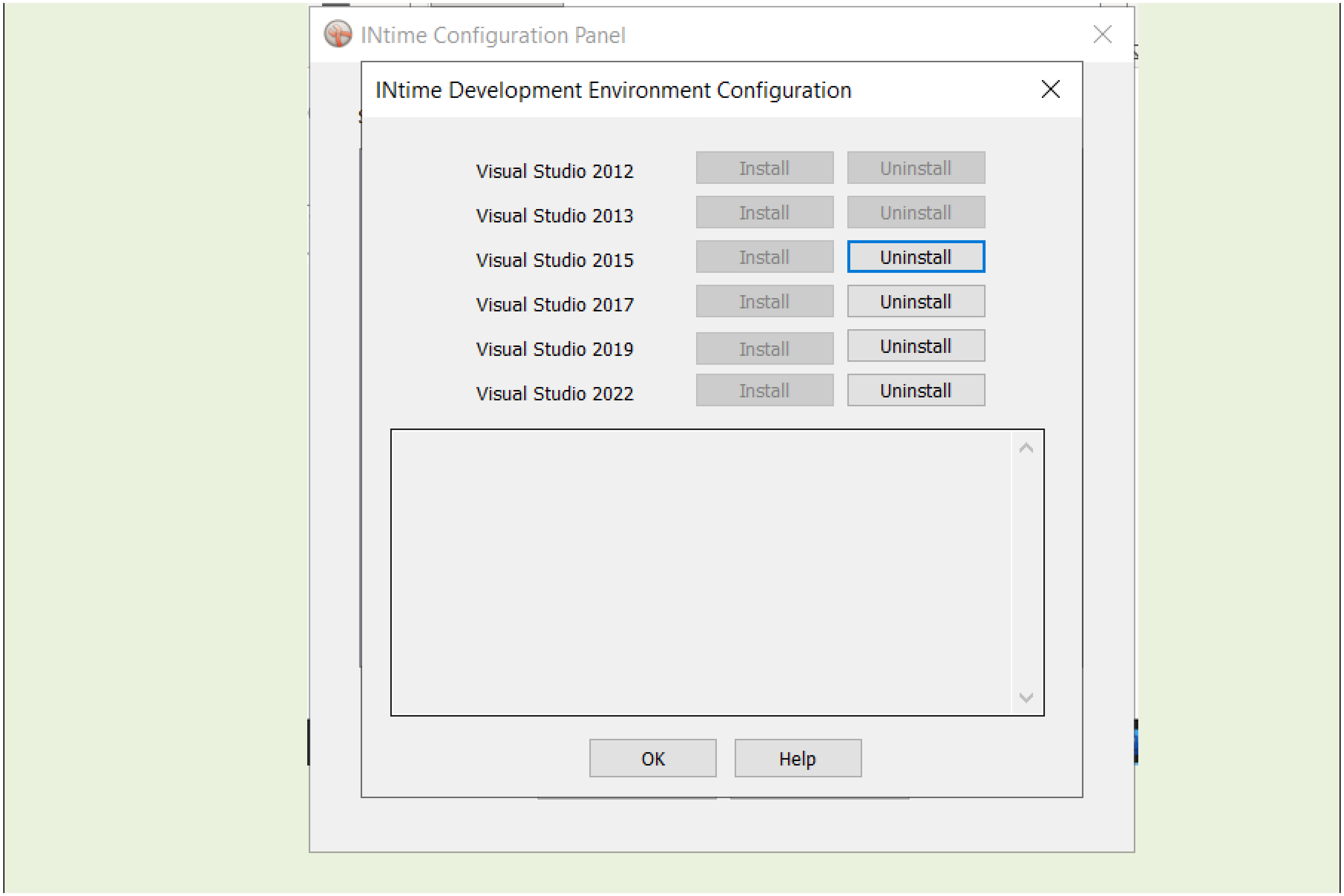
All languages | All platforms | All project types

Boost Application Wizard
Create an INtime Boost Application Skeleton.

Not finding what you're looking for?
Install more tools and features

Back | Next

3:31 PM
3/13/2024



INtime Configuration Panel

INtime Development Environment Configuration

Visual Studio 2012	Install	Uninstall
Visual Studio 2013	Install	Uninstall
Visual Studio 2015	Install	Uninstall
Visual Studio 2017	Install	Uninstall
Visual Studio 2019	Install	Uninstall
Visual Studio 2022	Install	Uninstall

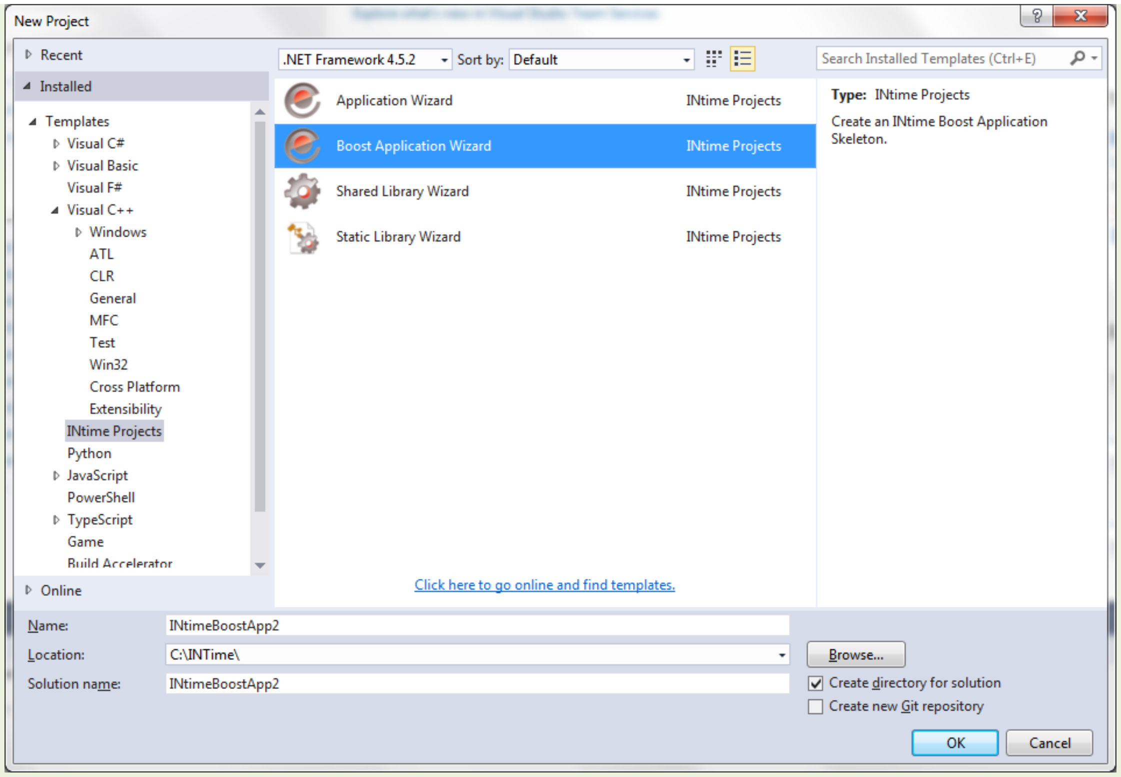
4 Set Up Project

Since the boost project is vast, it is recommended that INtime boost projects use the Property Manager in Visual Studio with project property sheets. These property sheets provide project property definitions (PRPD) for include directories, linker library directories, and linker input dependencies.

b. Decide how you want to configure your project.

In Visual Studio, and INtime, you can create a boost application. The following shows for an empty project.

c. **Create A New Project**



.NET Framework 4.5.2 — Sort by: Default

Application Wizard	INtime Projects
Boost Application Wizard	INtime Projects
Shared Library Wizard	INtime Projects
Static Library Wizard	INtime Projects

Type: INtime Projects

Create an INtime Boost Application Skeleton.

Name: INtimeBoostApp2

Location: C:\INTime\

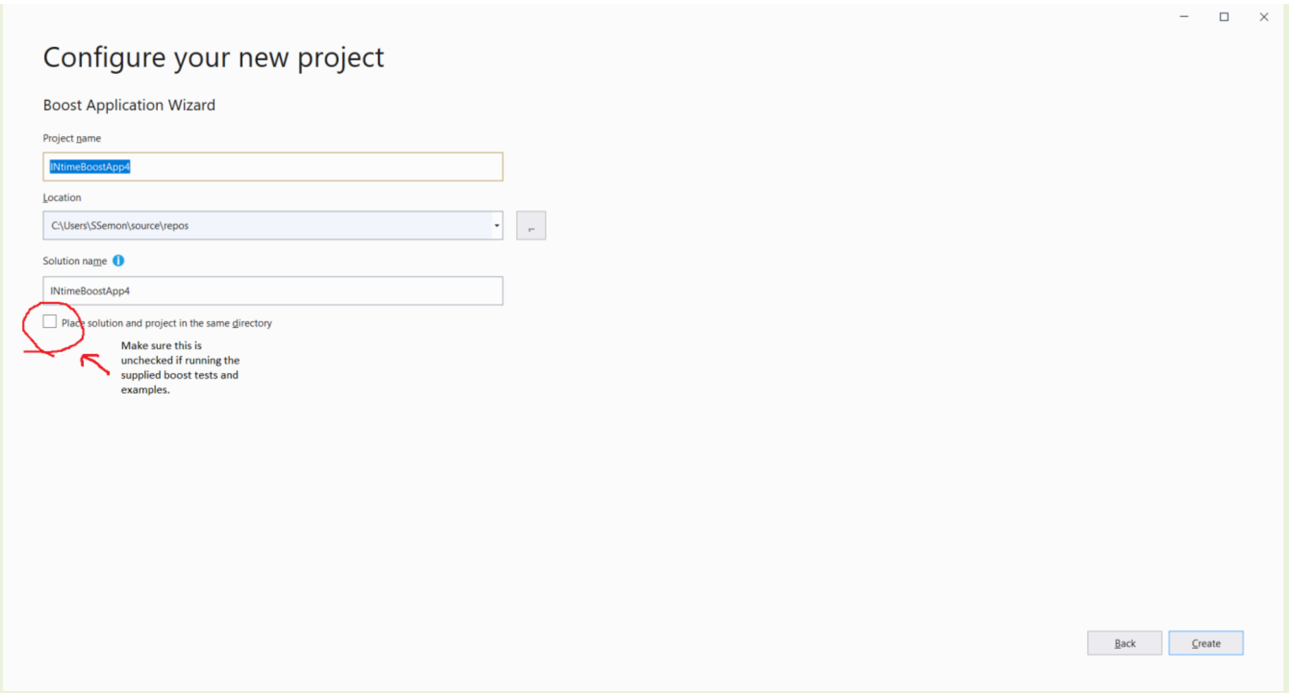
Solution name: INtimeBoostApp2

[✓] Create directory for solution

[ ] Create new Git repository

d. **If developing projects for testing please uncheck place solution and project in same directory.**

The tests require files which are read from dynamically, Installation_Paths.txt is located in the support_folder located in the solution directory. The checkbox needs to be unchecked for this file to be found during runtime.



Configure your new project

Boost Application Wizard

Project name: INtimeBoostApp4

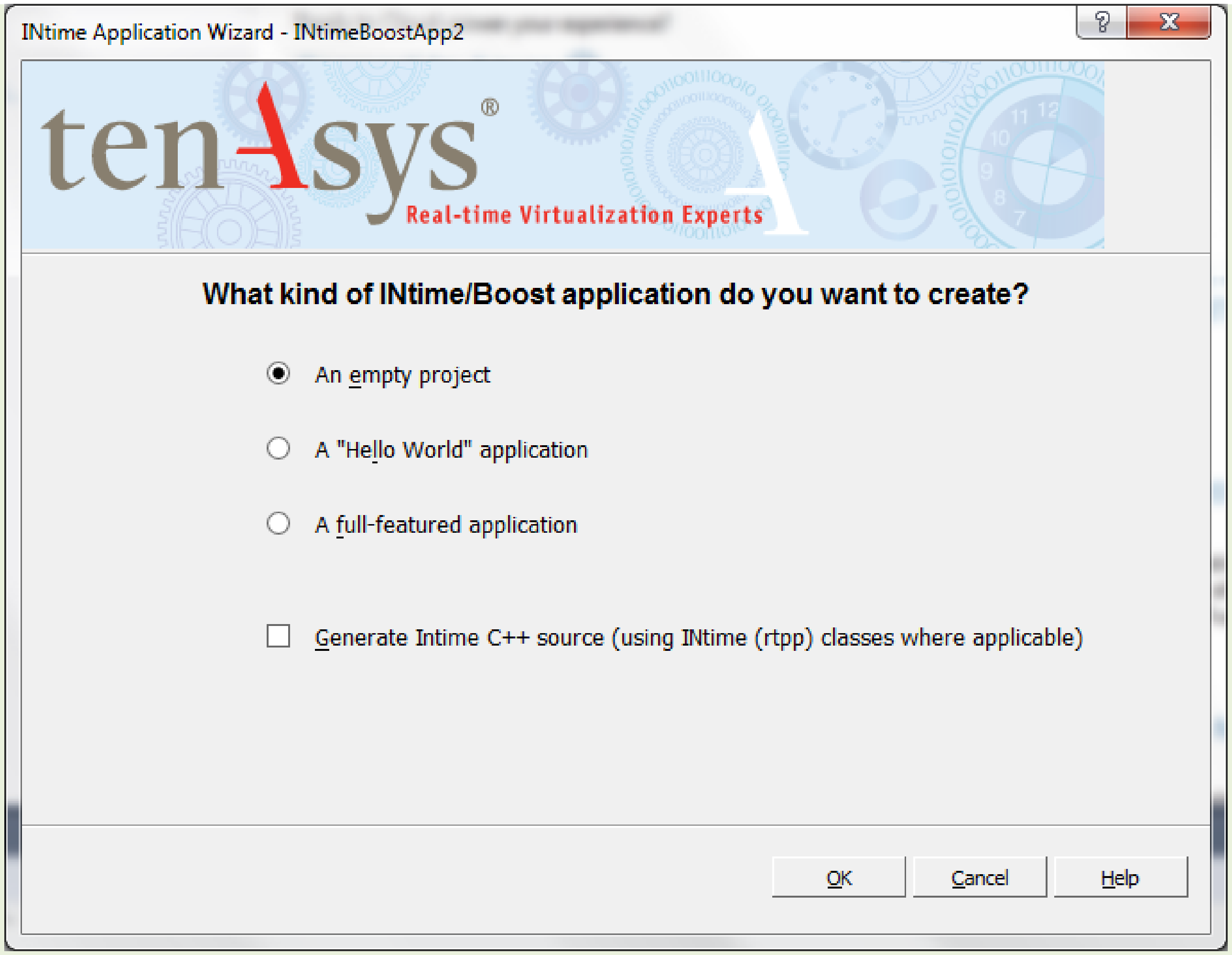
Location: C:\Users\SSemon\source\repos

Solution name: INtimeBoostApp4

☐ Place solution and project in the same directory

Make sure this is unchecked if running the supplied boost tests and examples.

e. Empty Project


INtime Application Wizard - INtimeBoostApp2

What kind of INtime/Boost application do you want to create?

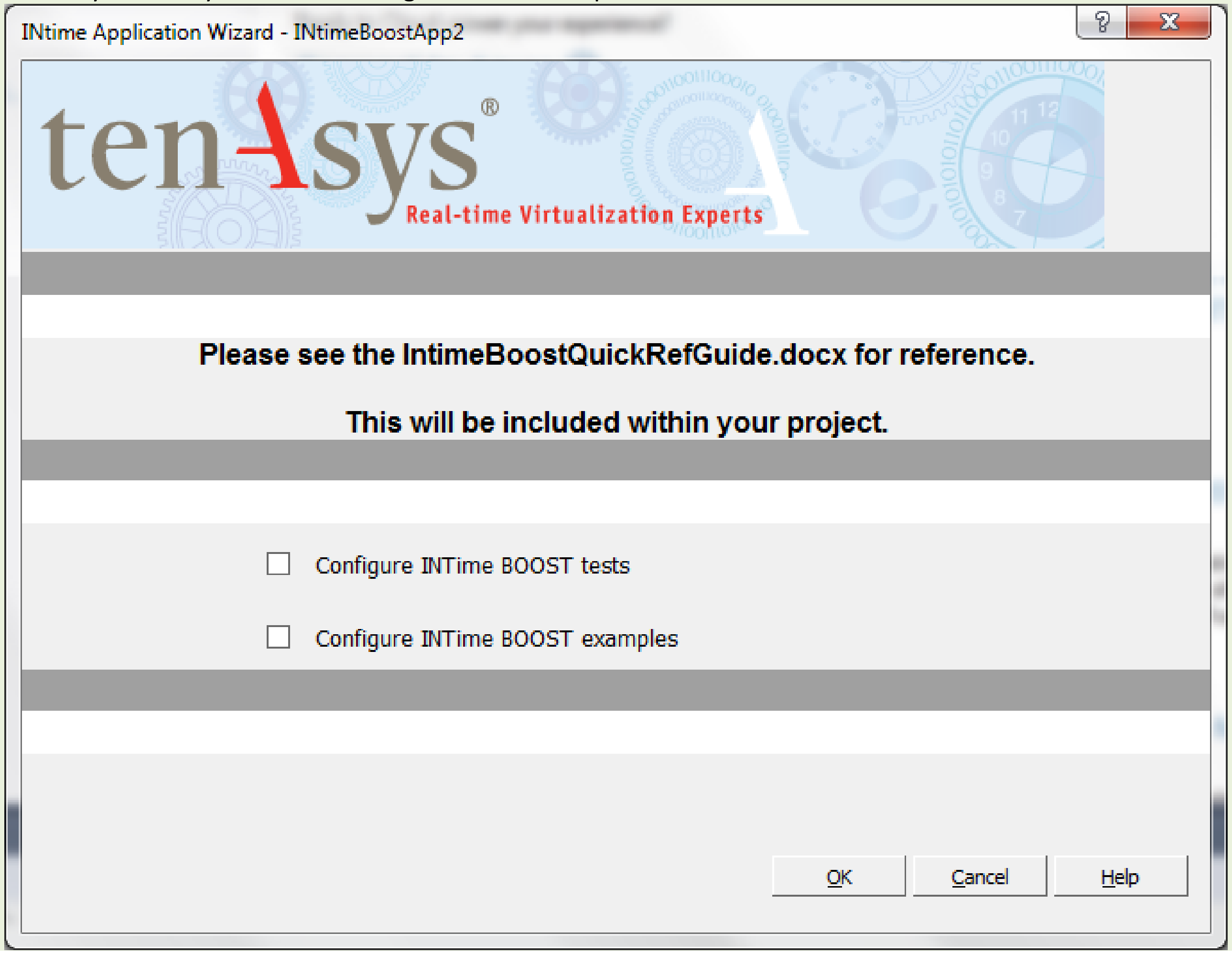
(●) An empty project

( ) A "Hello World" application

( ) A full-featured application

[ ] Generate Intime C++ source (using INtime (rtpp) classes where applicable)

f. Boost Wizard The wizard sets the property manager, and project property definitions (PRPD) for you. You may select if you want to configure boost examples or boost tests.


INtime Application Wizard - INtimeBoostApp2
tenAsys®
Real-time Virtualization Experts

Please see the IntimeBoostQuickRefGuide.docx for reference.

This will be included within your project.

☐ Configure INTime BOOST tests

☐ Configure INTime BOOST examples

5 Property Sheets

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

a. Property Sheets

The boost wizard adds the following property sheets to the project. (PRPD) project **property** *definitions.*
boostlib.props *– (PRPD) used to build the cpp11 - vs2015 project.*
boost17.props *– (PRPD) used to build the cpp17 - vs2017 project.*
Boost20.props *– (PRPD) used to build the cpp20 - vs2019 project or vs2022 project.*
boostlibtest.props *– This sheet sets the (PRPD) for the test directories of INtime boost.*
boostlibExample.props *– This sheet sets the (PRPD) for the example directories of Intime.*
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.

b. How to manually update the Property Manager Config

Within Visual Studio, click the **View** tab. In the **Other Windows** group, click **Property Manager**. In the property manager window right click on your project and click on **Add Existing Property Sheet** button. Select the boost[…].props file from **INtimeInstallDirectory**\INtime**vstudioVersion**\wizards\boostprojects.

c. How to change 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 include 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. Generic Boost: $(boost_lib_include_paths);$(other);%(AdditionalIncludeDirectories) For Boost Tests: $(boost_lib_include_paths);$(other); (boost_test_include_directories); %(AdditionalIncludeDirectories) For Boost Examples: $(boost_lib_include_paths);$(other); $(boost_example_include_directories); %(AdditionalIncludeDirectories)

d. How to change preprocessor definitions and macros.

In **C++ General** group modify the **Preprocessor Definitions**, you will at least want to have:

For Cpp11/Vs2015:

```
__INTIME_BOOST__;__INTIME_CPP11;__INTIME_USE_IOCP__;
__INTIME_PTHREAD__;_WIN32;_WIN32_WINNT=0x0501;_HAS_NAMESPACE;
_USRDLL;_USE_64BIT_TIME_T;__INTIME_TEST_TOOLS;
_BOOST_GIL_NO_TIFF_LIB;BOOST_GIL_NO_PNG_LIB;
BOOST_THREAD_USES_CHRONO;
BOOST_THREAD_THROW_IF_PRECONDITION_NOT_SATISFIED;
BOOST_USE_OWN_EXCEPTIONS;$(wolfssl_macros);%(PreprocessorDefinitions)
```

For Cpp17/Vs2017:

```
__INTIME_BOOST__;__INTIME_CPP17;__INTIME_USE_IOCP__;
__INTIME_PTHREAD__;_WIN32;_WIN32_WINNT=0x0501;_HAS_NAMESPACE;
_USRDLL;_USE_64BIT_TIME_T;__INTIME_TEST_TOOLS;
BOOST_GIL_NO_TIFF_LIB;BOOST_GIL_NO_PNG_LIB; BOOST_THREAD_USES_CHRONO;
BOOST_THREAD_THROW_IF_PRECONDITION_NOT_SATISFIED;
BOOST_USE_OWN_EXCEPTIONS;
SILENCE_ALL_CXX17_DEPRECATION_WARNINGS;
_SILENCE_BOGUS_WARNINGS;__BYPASS_ITERATOR_DEBUG_LEVEL_2__;
$(wolfssl_macros);%(PreprocessorDefinitions)
```

For Cpp20/Vs2020, or Cpp20/Vs2022:

```
__INTIME_BOOST__;__INTIME_CPP20;__INTIME_USE_IOCP__;
__INTIME_PTHREAD__;WIN32;_WIN32_WINNT=0x0501;_HAS_NAMESPACE;
_USRDLL;_USE_64BIT_TIME_T;__INTIME_TEST_TOOLS;
BOOST_GIL_NO_TIFF_LIB;BOOST_GIL_NO_PNG_LIB; BOOST_THREAD_USES_CHRONO;
BOOST_THREAD_THROW_IF_PRECONDITION_NOT_SATISFIED;
BOOST_USE_OWN_EXCEPTIONS;
SILENCE_ALL_CXX17_DEPRECATION_WARNINGS;
_SILENCE_BOGUS_WARNINGS;__BYPASS_ITERATOR_DEBUG_LEVEL_2__;
$(wolfssl_macros);%(PreprocessorDefinitions)
```

e. 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

f. **How to change linker properties**

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

Generic Boost: C:\INtime\rt\lib; or $(boost_lib_directories)

In Linker Input group modify the Additional Dependencies you will at least want to have.

Generic Boost: $(boost_lib_dependencies);

For Boost Tests: $(boost_test_dependencies);

For Boost Examples: $(boost_example_dependencies);

g. **Set Project Property Definitions (PRPD) within the property sheets.**

After the property manager is set up, you may want to update the default settings.

boostlib.props / Boost17.props / Boost20.props

```
 Modify your submodule install location within this placeholder.
<!--USER UPDATES HERE-->
<!-- Boost's Submodules Download Location -->
<boost_submodules_location>
       Put_boost_submodules_location_here
<boost_submodules_location>
```

This is the location where you **downloaded boost** including the **\lib,** so add the whole path there including the **\lib.** For example, I used: c:\git\boost\libs.

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

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

boostlib.props / cpp11/ vs2015

If necessary, modify the INtime header locations here.

```
<INtime_include>
   $(INtime)\rt\include\cpp11;
   $(wolfssl_location);
   $(zlib_location);
   $(pthreads_root);
   $(INtime)\rt\include\network7;
   $(INtime)\rt\include\network7\sys;
   $(INtime)\rt\include;
</INtime_include>
<!-- LibJpeg Include Path -->
<jpeg_include_path>$(INtime)\rt\include\jpeglib</jpeg_include_path>
```

	Boost17.props / cpp17/ vs2017
	If necessary, modify the INtime header locations here.
	`<INtime_include>`
	`   $(INtime)\rt\include\cpp17;`
	`   $(wolfssl_location);`
	`   $(zlib_location);`
	`   $(pthreads_root);`
	`   $(INtime)\rt\include\network7;`
	`   $(INtime)\rt\include\network7\sys;`
	`   $(INtime)\rt\include;`
	` </INtime_include>`
	` <!-- LibJpeg Include Path -->`
	
	`<jpeg_include_path>$(INtime)\rt\include\jpeglib</jpeg_include_path>`
	
	Boost20.props / cpp20/ vs2019/vs2022
	If necessary, modify the INtime header locations here.
	`<INtime_include>`
	`   $(INtime)\rt\include\cpp20;`
	`   $(wolfssl_location);`
	`   $(zlib_location);`
	`   $(pthreads_root);`
	`   $(INtime)\rt\include\network7;`
	`   $(INtime)\rt\include\network7\sys;`
	`   $(INtime)\rt\include;`
	` </INtime_include>`
	` <!-- LibJpeg Include Path -->`
	
	`<jpeg_include_path>$(INtime)\rt\include\jpeglib</jpeg_include_path>`

h. Setup to use Wolfssl

If the developer would like to use WolfSSl in their project, they first must contact wolf ssl and purchase a license.
Thus after doing so they may update the following properties in the Boost[…].props file.

```
<wolfssl_location>
    Locations where the wolfssl headers are.
</wolfssl_location>
<wolfssl_macros>
          BOOST_ASIO_USE_WOLFSSL;
           WOLFSSL_ASIO;
           INTIME_RTOS;
           OPENSSL_EXTRA;
           OPENSSL_ALL;
           WOLFSSL_ALLOW_SSLV3;
</wolfssl_macros>
<wolfssl_lib
     //the library name.
      libwolfssl551.lib
</wolfssl_lib>
```

i. Setup to use 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 the following properties in the Boost[…].props file.

```
<zlib_location>
    Location where the zlib git headers are.
</zlib_location>
<zlib_lib>
     zlibstaticd.lib;
</zlib_lib>
```

6 Setup Examples

The wizard attempts to setup 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.

1. **Config**

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.

```
boostlibExample.props:   Modify your example location with this placeholder.
<Choose>;
Otherwise>
<!--USER UPDATES HERE-->
<PropertyGroup Label="UserMacros">
<!-- Location where examples were installed -->
<root_example_location>'Put where you installed the examples here'</root_example_location>
```

7 Setup Tests

The wizard attempts to setup most of 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.

1. **Config**

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.

boostlibtest.props: `Modify your test location with this placeholder.`

```
<Choose>;
Otherwise>
<!--USER UPDATES HERE-->
<PropertyGroup Label="UserMacros">
<!-- Location where examples were installed -->
><root_test_location>'Put where you installed the tests here'</root_test_location>
```

8 Test 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.

1. **Empty Project**	Recommend using an empty project. Add a blank .cpp file to the project.
2. **Running Examples**	Copy one of the examples into the .cpp file. Build the project and run it.
3. **Running Tests**	Copy one of the tests into the .cpp file. Build the project and run it.

9 Setup Automation

The wizard Install an automation script to run tests automatically. However, this need to be configured.

1. **Edit Installation_Paths.txt**

This file is used to find the path of runtime files used by the tests. In order 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.
```

2. **Config**

Edit the following files: (Note: boost17 files have a 17.cmd, boost20 have 20.cmd)
TestOne.cmd, TestOneWNodeReset.cmd, testdir.cmd, testall.cmd
TestOne.cmd/TestOneWNodeReset.cmd:
set solutionName='The solution you chose with the wizard.
set projectName=the project name you chose with the wizard.
set cppFile=the file name that you added to the empty project.

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

3. **Commands**

Open a visual studio command prompt. i.e. Developer Command Prompt for VS2015
Set current directory to your project location. for example, I used:
cd C:\Users\ssemon\Documents\Visual Studio 2015\Projects\INtimeApp1>
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.
And C:\INtime\Tests\boostlibtest\all_test_results.txtis an output log file.
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.