



Product:	CDPUnit
Product version:	v0.2
Document ID:	PM-CDPUnit
Doc revision:	PA1
Written/Aprr.:	KD /
Date:	11. Mar. 2010

Industrial Control Design AS



CDPUnit v0.2

Programmer Manual

The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.

Industrial Control Design AS, www.icd.no, support@icd.no, forum.icd.no

Contents

1. INTRODUCTION.....	3	6. TEST RUNNERS.....	12
1.1. About.....	3	6.1. Overview.....	12
1.2. Terms and Definitions.....	3	6.1.1. Description.....	12
		6.1.2. Class overview.....	12
2. INSTALLATION.....	4	6.2. ConsoleRunner.....	12
2.1. Prerequisites:.....	4	6.2.1. Description.....	12
2.2. Linking with the CDPUnit library.....	4	6.2.2. Methods.....	12
2.3. How to add the CDPUnit library in a new or existing project.....	5	6.2.3. Example.....	12
2.4. Modify the project XML files.....	8	6.2.4. Command-line arguments.....	12
		6.3. CDPConsoleRunner.....	13
3. FUNCTIONAL OVERVIEW.....	9	6.3.1. Description.....	13
3.1. ComponentTestRunner Component.....	9	6.3.2. Methods.....	13
3.2. CDPUnit API.....	9	6.3.3. Example.....	13
3.2.1. DynamicTestInterface.....	9		
3.2.2. HelperMacros.h.....	9	7. CPPUNIT.....	13
3.3. Test Runners.....	9	7.1. C++ unit testing framework.....	13
4. COMPONENTTESTRUNNER COMPONENT.....	10	8. APPENDIX.....	13
4.1. About.....	10	8.1. References.....	13
4.2. Overview.....	10		
4.2.1. Component Architecture.....	10		
4.2.2. Component States.....	10		
4.3. Sending command messages.....	10		
4.3.1. About.....	10		
4.3.2. Text command argument.....	10		
5. CDPUNIT API.....	11		
5.1. DynamicTestInterface.....	11		
5.1.1. Description.....	11		
5.1.2. Methods.....	11		
5.1.3. Example.....	11		
5.2. HelperMacros.h.....	11		
5.2.1. Description.....	11		
5.2.2. Macros.....	11		
5.2.3. Example.....	11		

1. Introduction

1.1. About

This document describes how the CDPUnit CDP component works, and how to set it up and use it with the CDP system. The CDPUnit CDP component has the following features:

- Facilitates testing of CDP components at the low and intermediate level.
- Allows for white-box and black-box testing of CDP components.
- Integrates with and uses the widely popular CppUnit testing framework.

1.2. Terms and Definitions

Black-Box Testing

Testing that only relies on the public interface of the unit under testing, and thusly is loosely coupled with the code under test.

CDP

Control Design Platform.

CDP Controller

Computer (Usually an industrial computer) running CDP application, usually on a true real- time operating system.

Component

Object with strict interface specification.

Component Test

Test case that requires an active CDP component running in a CDP application. Frequently a test component that inherits the component under testing to allow instrumentation and inspection of the component during tests (white-box testing).

Unit Test

Stand-alone test case that does not need an active CDP component nor complete running CDP system. Can often also be run entirely outside of CDP, with dependancies only to CppUnit.

White-Box Testing

Testing that accesses or modifies internal state of the unit under testing, and thusly is tightly coupled with the implementation.

2. Installation

The CDPUnit can be delivered both as source code and as a separate library which is linked into the application. If delivered as separate library, the CDPUnitLib Setup will by default install all files in sub-folders of “CDP Developer”.

2.1. Prerequisites:

- A valid CDP license
- Familiar with CDP
- Familiar with unit testing
- CDP version 2.3.1.10 (or newer)
- CppUnit version 1.12.1 (or newer)

2.2. Linking with the CDPUnit library

Copy the example ComponentTestRunner.xml component file that came with the distribution into your project's Components folder, and the ComponentTestRunner.xml model XML file into the Models folder.

Copy the CDPUnitLib folder either into your project, to a location for standard libraries, or copy the CDPUnitLib_<config>.lib and header-files to a location where your linker can find it.

Include the CDPUnitLib.h header file in your application's Libraries.h file:

```
#include <CDPUnitLib.h>
```

If necessary, update the include paths by adding the path to where to find CDPUnitLib header-files (and source code if available):

- CDP_Application -> Properties
 - C/C++ -> General -> Additional Include Directories

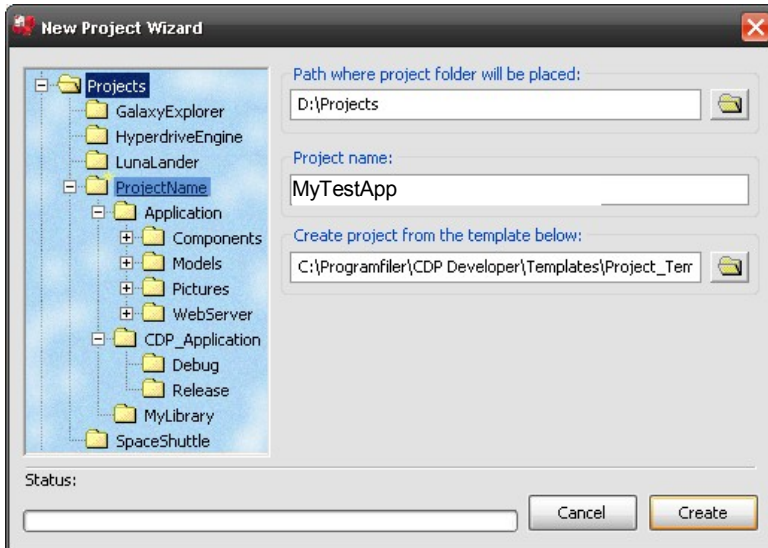
Link the CDPUnit library into your application by putting CDPUnit_<config>.lib in a folder included in your linker path. For the CDP_Application project:

- CDP_Application->Properties
 - Linker -> Input -> Additional dependencies

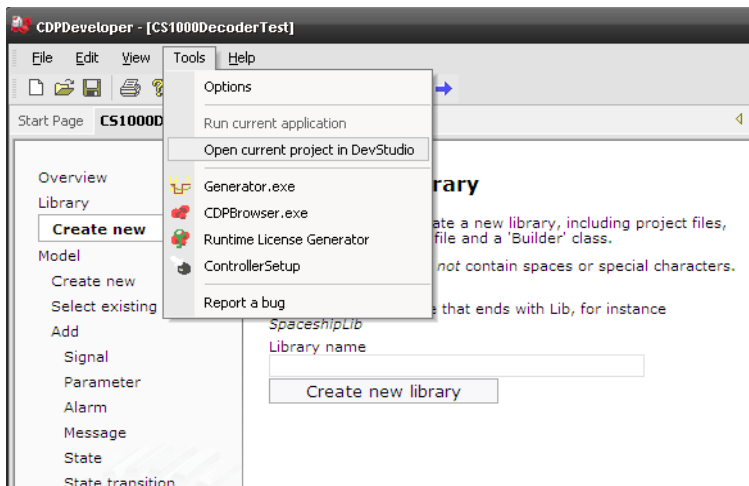
Finally, link to the CppUnit library. Same procedure as for linking with the CDPUnit library above.

2.3. How to add the CDPUnit library in a new or existing project

- Start CDPDeveloper located at Start Menu > Programs > CDP > CDPDeveloper
- Make a new project by Selecting File->New, type in project name and click Create (or skip four steps forward and open an already existing project in Visual Studio)

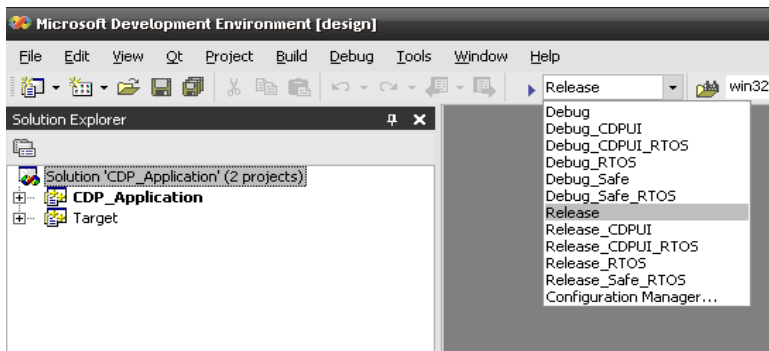


- Choose 'Open current project in DevStudio' from the Tools menu:

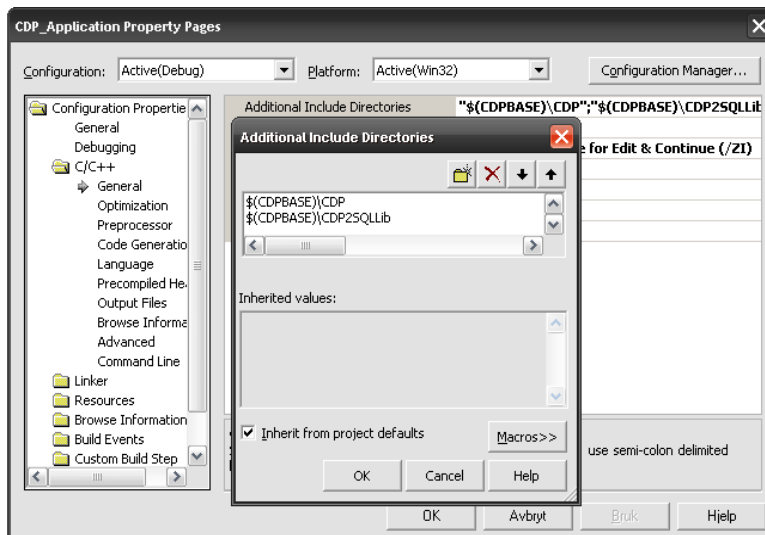


- If Visual Studio asks to convert the project, accept this by selecting 'Next'/'Finish'/'Close' until done. Close the conversion report.

- In Visual Studio, select the configuration that is right for your target platform.



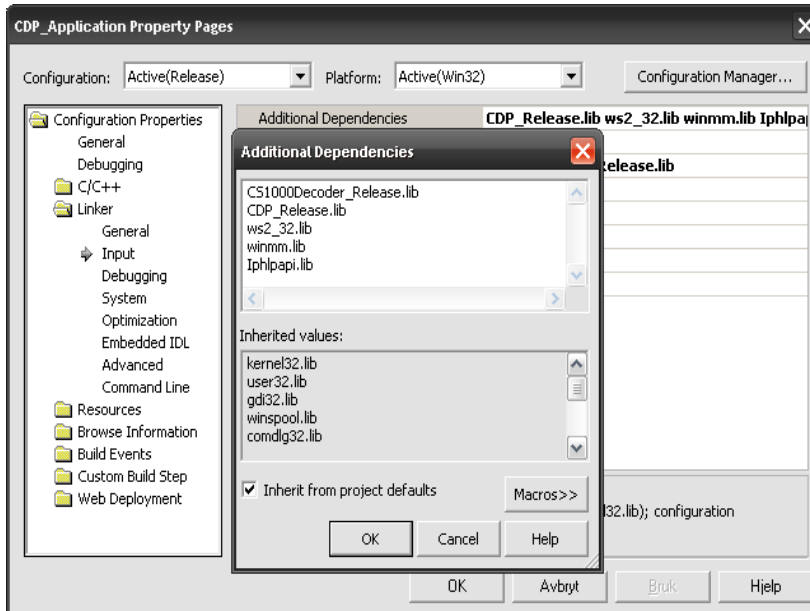
- Select 'CDP_Application' from the 'Solution Explorer', right-click and select Properties.
- In C++/Additional include Directories, make sure it says: “\$(CDBASE)CDPUnitLib”; “\$(CDBASE)CDP”. When compiling for On Time RTOS, the following include is also required: “;”\$(RTTarget)Include”.



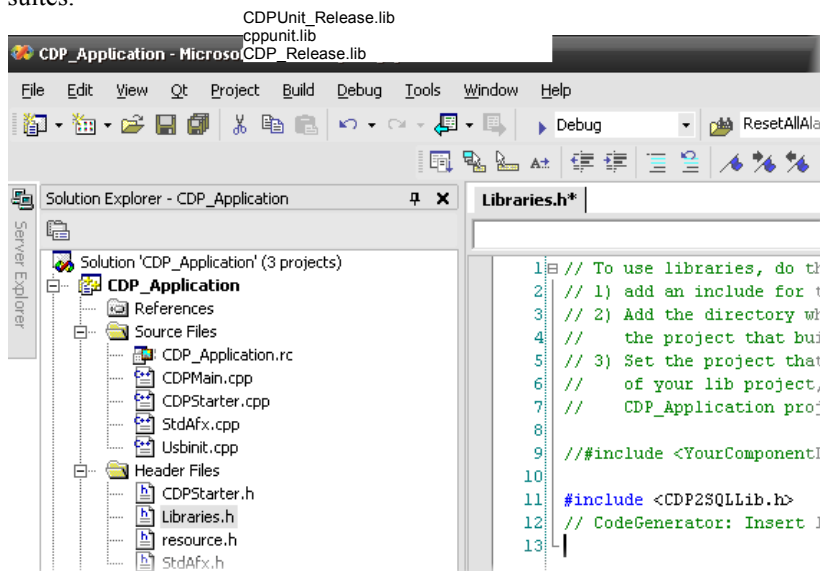
The compiler will look in the directories specified in 'Additional Include Directories' for files that you #include in your .cpp and .h files. If you get an error 'Can not open include file ...' then it is most likely caused by a missing include directory, or that the file you #include does not exist.

- Once you also start linking in libraries containing unit-tests or component tests, you may find that you also need to include the CppUnit headers in the include paths: “\$(CPPUNIT)\include”

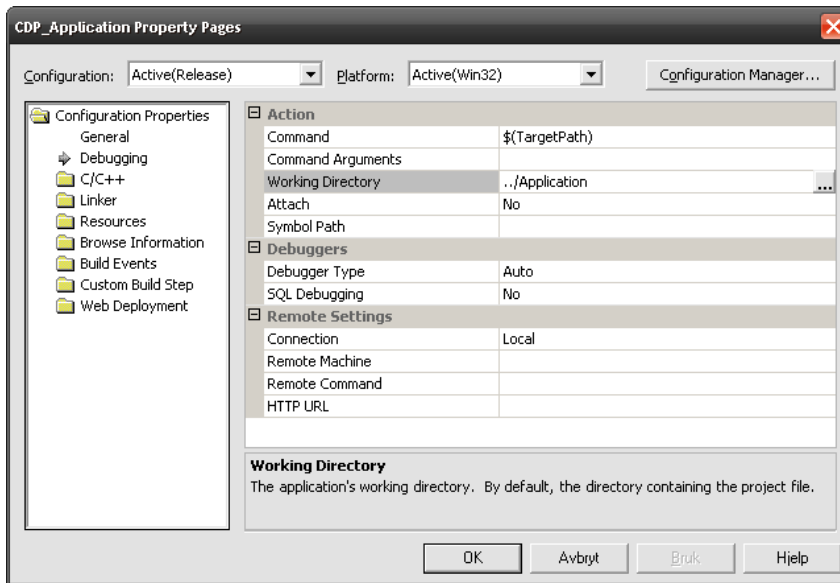
- In Linker->Input, make sure you list CDPUnitLib_Release.lib, cppunit.lib and CDP_Release.lib in addition to all the win32 libraries you need. This will ensure that the linker finds all the functions that is referenced in the code. Note that when compiling for other Operating Systems you will need to replace these libraries with the ones for the target OS. If you have unit tests or component tests in separate libraries, they also need to be linked in.



- In the 'Solution Explorer', inside the 'CDP_Application' project, locate the Header file Libraries.h, and add the line `#include <CDPUnitLib.h>`. If you have separate libraries with unit tests or component tests, you will also need to include the appropriate headers for the test components and/or unit test suites.



- Finally, make sure that the working directory of the project is set to ../Application. This will ensure that the executable is started where the configuration files are located.



2.4. Modify the project XML files

Add the following to your project's Application.xml:

Inside the <Components> element, add an instance of a ComponentTestRunner component, for instance:

```
<Component Name="ComponentTestRunner" Type="ComponentTestRunner"
src="Components/ComponentTestRunner.xml"></Component>
```

Or, inside the <Subcomponents> element, add:

```
<Subcomponent Name="ComponentTestRunner" Type="ComponentTestRunner"
src="Components/ComponentTestRunner.xml"> </Subcomponent>
```

This will tell CDP to initialize a component named “ComponentTestRunner” from a component file located at “Components/ComponentTestRunner.xml”. Make sure that your Models folder contains a ComponentTestRunner.xml model file, or the component will not be initialized correctly.

Configuration is done by modifying the component xml file. It should not be necessary to modify the model XML file. For details, refer to the CDPUnit User Manual.

3. Functional Overview

3.1. ComponentTestRunner Component

The ComponentTestRunner CDP component is a ready-to-use component for running unit tests and component tests. The component is configured in XML. The ComponentTestRunner component can also be sub-classed for extending the functionality.

3.2. CDPUnit API

The CDPUnit API provides a thin wrapper on top of CppUnit.

3.2.1. DynamicTestInterface

DynamicTestInterface inherits and implements the abstract TestFactory from CppUnit to allow integration with other CppUnit-based runners. The DynamicTestInterface is implemented by Component Tests for creating component tests.

3.2.2. HelperMacros.h

The CDP_UNIT_TEST_*_DYNAMIC macros in HelperMacros.h

Used for registering Component Tests dynamically at run-time.

3.3. Test Runners

A few simple command-line unit test runner applications. Does not require a running CDP system.

Can be used when there are no or few CDP dependencies in the unit tests.

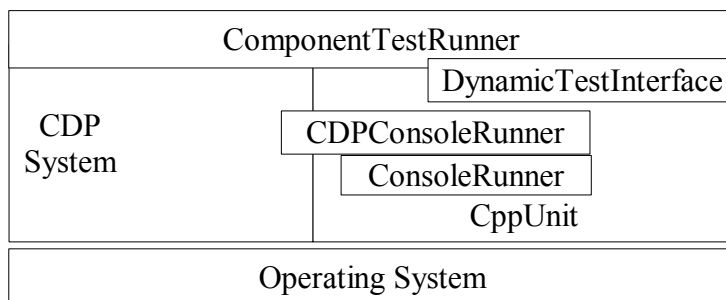
4. ComponentTestRunner Component

4.1. About

The ComponentTestRunner CDP component is a ready-to-use component. The component is configured in XML as described in the CDPUnit User Manual.

4.2. Overview

4.2.1. Component Architecture



4.2.2. Component States

The component includes multiple states. See the CDPUnit User Manual for more information.

4.3. Sending command messages

4.3.1. About

The RegisterTests and RunTests text command messages are used to prepare and run the test cases.

4.3.2. Text command argument

If a text value is sent along with the RunTest command, it is used as a test selection name.

5. CDPUnit API

The CDPUnit API is introduced as a thin layer between CDP and CppUnit to help create testcases for, and running them in, CDP applications.

5.1. DynamicTestInterface

5.1.1. Description

The DynamicTestInterface must be implemented by any CDP component that wants to serve as a test fixture. The ComponentTestRunner uses this interface to identify and fetch testcases from CDP test components.

5.1.2. Methods

- `makeTest` – creates a collection of test cases and returns to them.

5.1.3. Example

Example on how to define a test component that implements DynamicTestInterface.

```
#include <CDPUnit/CDPUnit.h>
class MyTestComponent
  : public MyComponent, public CDPUnit::DynamicTestInterface, public CppUnit::TestFixture
{
  // ...
```

5.2. HelperMacros.h

5.2.1. Description

The helper macros are modifications and extensions based on the CppUnit helper macros. They are used to aid in registering test fixtures and test cases, so that you do not have to write code that explicitly collects all the individual tests.

5.2.2. Macros

- `CDP_UNIT_TEST_SUITE_DYNAMIC` – start registering a test suite.
- `CDP_UNIT_TEST_DYNAMIC` – register a specific test into the test suite.
- `CDP_UNIT_TEST_SUITE_END_DYNAMIC` – end registration of a test suite.
- `CDP_UNIT_TEST_SUITE_REGISTER_DYNAMIC` – add code to return the above tests dynamically.

5.2.3. Example

Example on how to register tests in a component.

```
class MyTestComponent
  : public MyComponent, public CDPUnit::DynamicTestInterface, public CppUnit::TestFixture
{
  // Registering the tests.
  CDP_UNIT_TEST_SUITE_DYNAMIC( MyTestComponent );
  CDP_UNIT_TEST_DYNAMIC( testSomething );
  CDP_UNIT_TEST_DYNAMIC( testSomethingElse );
  CDP_UNIT_TEST_SUITE_END_DYNAMIC();
  CDP_UNIT_TEST_SUITE_REGISTER_DYNAMIC();    ///< Implements makeTest() for this class.
  // ...
```

6. Test Runners

6.1. Overview

6.1.1. Description

The CDPUnit Test Runners are found in the CDPUnit::UI namespace and provides a way of running unit tests as a stand-alone executable without CDP.

Note: Depending on how coupled a test case is with CDP, it may not be possible to run it without a functioning CDP system.

6.1.2. Class overview

The add-on provides the following test runners:

- ConsoleRunner – run unit tests with no CDP dependency.
- CDPConsoleRunner – run unit tests with some CDP dependency.

6.2. ConsoleRunner

6.2.1. Description

The ConsoleRunner is a simple test runner class for creating command-line test runner applications without CDP. It uses the TestFactoryRegistry and will only find unit tests that have been registered with it. Component tests are not registered nor run.

6.2.2. Methods

- ParseCommandLineArgs – extract options from the command line arguments.
- RunConsole – run the tests in the console.

6.2.3. Example

Short example of creating the runner.

```
#include "Libraries.h" //< This must also include the test definitions.
#include "CDPUnit/UI/ConsoleRunner.h"
int main(int argc, char *argv[])
{
    CDPUnit::UI::ConsoleRunner myTestApp;
    myTestApp.ParseCommandLineArgs( argc, argv );
    return myTestApp.RunConsole();
}
```

6.2.4. Command-line arguments

The following command-line arguments can be given to the console runner.

- | | |
|---------------|--|
| --verbose | Gives a more detailed report of the test run. |
| --test [name] | Run a specific test suite or test case only. |
| postbuild | Used for running from Visual Studio post-build step. |

6.3. CDPConsoleRunner

6.3.1. Description

Extension to ConsoleRunner that adds some helper functionality to run tests with some minor CDP dependencies.

6.3.2. Methods

- RunMain – reimplemented from ConsoleRunner, starting the tests in a separate thread with increased priority to allow tests to do priority changes. (Useful if you are testing thread-dependant code.)

6.3.3. Example

Simple example

```
#include "CDPUnit/UI/CDPConsoleRunner.h"
int main(int argc, char *argv[])
{
    CDPUnit::UI::CDPConsoleRunner myTestApp;
    myTestApp.ParseCommandLineArgs( argc, argv );
    return myTestApp.RunConsole();
}
```

7. CppUnit

At the bottom of the CDPUnit application and library, lies the CppUnit library.

7.1. C++ unit testing framework

CppUnit is a C++ unit testing framework that started as a port of JUnit to C++ by Michael Feathers.

The design of the framework is similar to others in the “xUnit” family of test frameworks, so people familiar with other frameworks (JUnit, NUnit, etc) will quickly find their way.

It is Open Source and available under the GNU Lesser General Public License (LGPL).

8. Appendix

8.1. References

CppUnit - C++ unit testing framework
<http://cppunit.sourceforge.net/>