

Getting Started with CppDepend

How to Get Set Up and Running with CppDepend

Whether you have purchased or downloaded the trial of CppDepend, we thank you for your involvement and interest in our product. Here we have compiled a quick "Getting Started" guide to help you get up and running with CppDepend as quickly as possible.

While we have tried to collect the basics in this document, we urge you to experiment and explore CppDepend on your own code bases. There are a ton of features and possibilities, many of which are out of the scope of this document.

CppDepend has been in development for over 9 years, and with every version we take into account user feedback and requests. For any additional help, reach out to us at support@cppdepend.com, go to our User Voice to leave a suggestion.

1. How to install CppDepend

- Download the CppDepend zip file,
- Extract the **CppDepend .zip** file that starts the Installation Wizard,

NOTE: DON'T UNZIP THE FILES IN: '%PROGRAMFILES%/CPPDEPEND' AS THIS WILL CREATE PROBLEMS BECAUSE OF WINDOWS PROTECTION

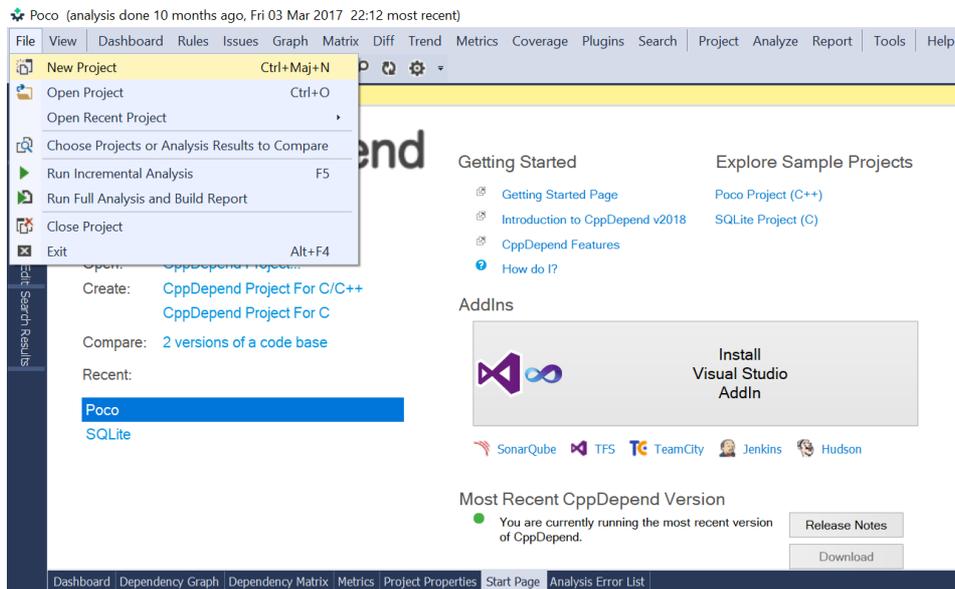
- Follow all steps suggested by the wizard.

2. How do I analyze my C\C++ applications using CppDepend?

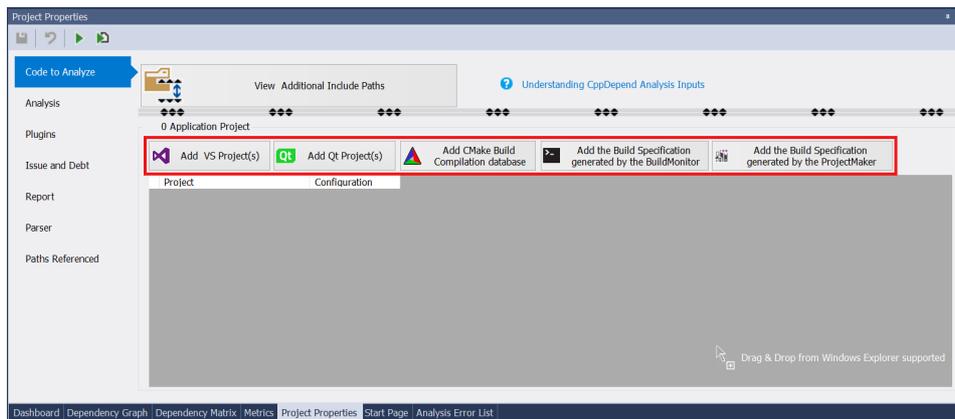
- Run **VisualCppDepend.exe** from windows OS or **launchVisualCppDepend.sh** from Linux.

Getting Started with CppDepend

- Create New CppDepend Project



- Add your C/C++ projects.

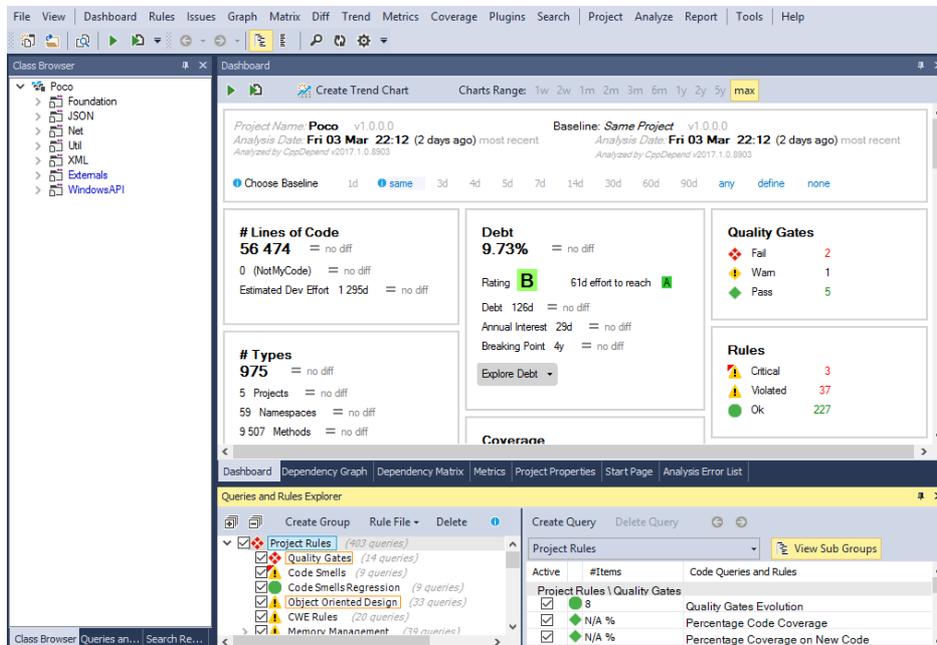


The following kind of projects are managed by CppDepend:

- **.vcproj, .vcxproj, .sln**: Visual Studio projects.
- **Qt projects**: CppDepend treats out of the box the Qt projects, you have just to specify the qmake tool path.
- **.build files**: The recommended way to analyse your C\C++ projects other than the VS ones, is to use the **BuildMonitor** tool to intercept all compilation and link commands.
- **Set of source files**: You can use the **ProjectMaker** tool to describe the specification of your project. With ProjectMaker you can specify your source files, the include paths and the defines.

Getting Started with CppDepend

- Click F5 to analyze your projects. The analysis will take a few seconds to a few minutes depending on the size of the projects.
- Once the analysis done, the result is displayed in the VisualCppDepend.exe UI.

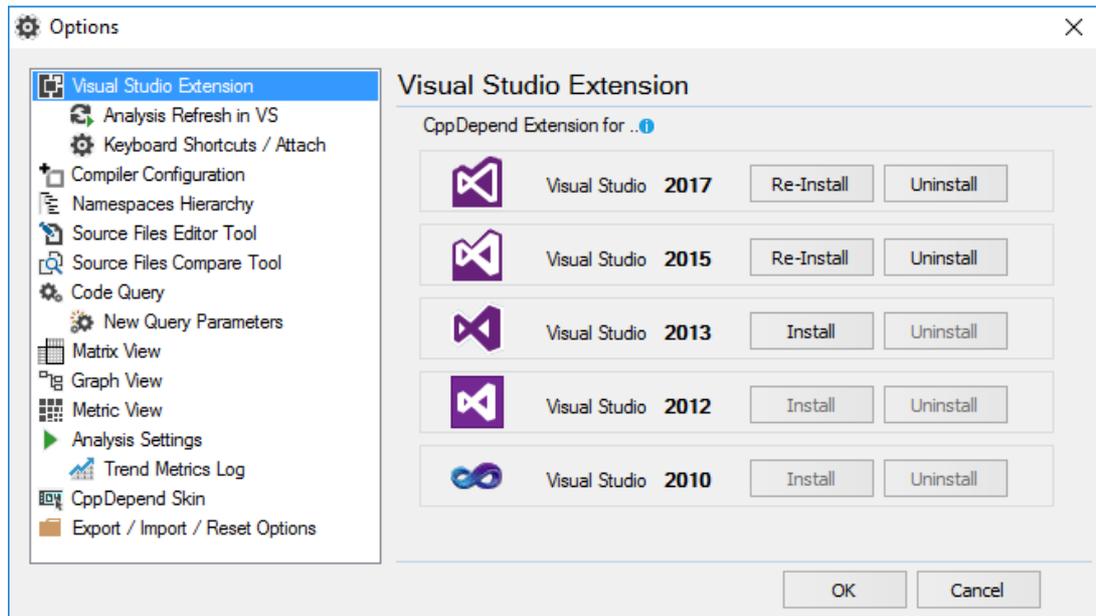


3. Installing the CppDepend addin for Visual Studio 2017, 2015, 2013, 2012, 2010

- Download the CppDepend installation zip file
- To install CppDepend, just install the files in a private application folder on your machine.
Don't install files in '%ProgramFiles%\CppDepend'. This will provoke problems because of Windows protection.
- Start *CppDepend.VisualStudioAddin.Installer.exe* or Start VisualCppDepend and install the VS addin from the start page.

Getting Started with CppDepend

- Click the button Install for the right Visual Studio version



4. Analyzing a Visual Studio solution

- Start Visual Studio after having installed the CppDepend addin.
- Open the Visual Studio solution containing the code to analyze.
- Click the greyed CppDepend circle status (bottom right of the Visual Studio window)

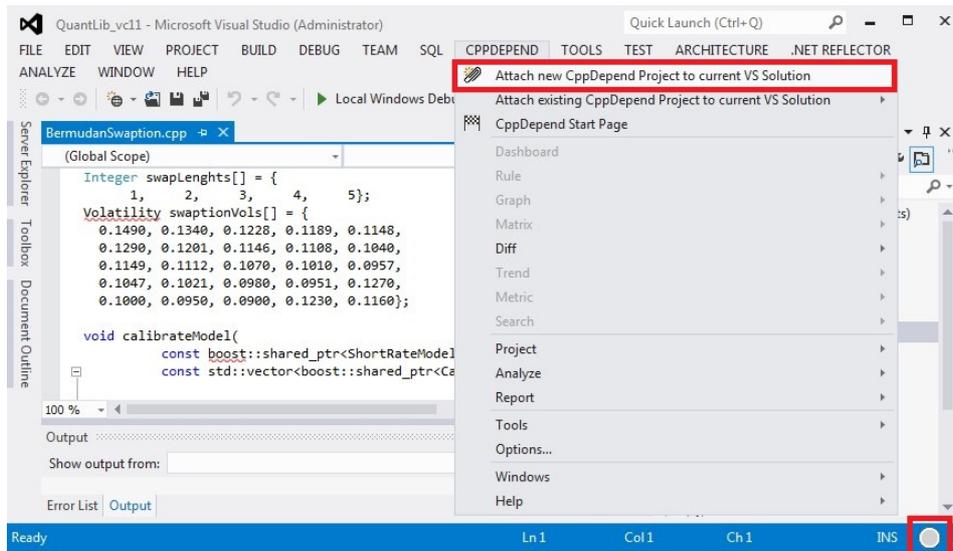
OR

click the menu CppDepend > Attach new CppDepend project to current Visual Studio solution

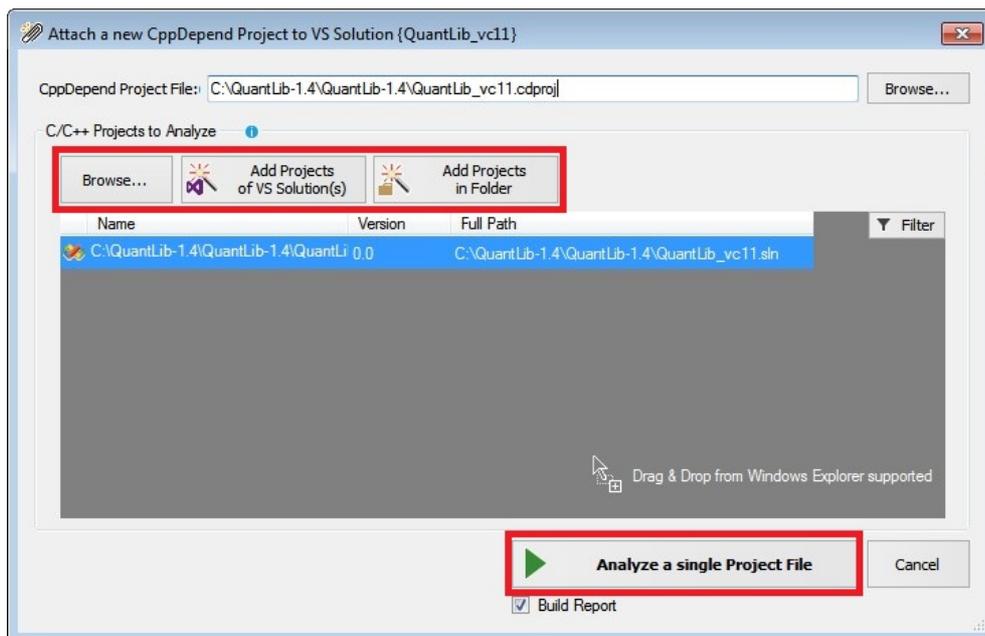
Note:

- A CppDepend project can be attached to several Visual Studio solutions.
- This is useful if you wish to analyze several C/C++ projects compiled with several Visual Studio solutions.
- A benefit will be the possibility to navigate across several Visual Studio solutions opened in several Visual Studio instances.

Getting Started with CppDepend



- The dialog opened has been populated with the Visual Studio solution. Here you can add more projects compiled from another solution



- Launch the CppDepend analysis start. It will just take a few minutes to analyze your code base for the first time.

5. Using CQLinq with CppDepend

CQLinq is a query language based off of the C# LINQ syntax and is a critical part of CppDepend which uses CQLinq queries to assess a code base and, furthermore, it allows users to create their own code queries. This allows architects and developers to custom make code rules to test for custom metrics. Using CQLinq, there is virtually no metric you can't test.

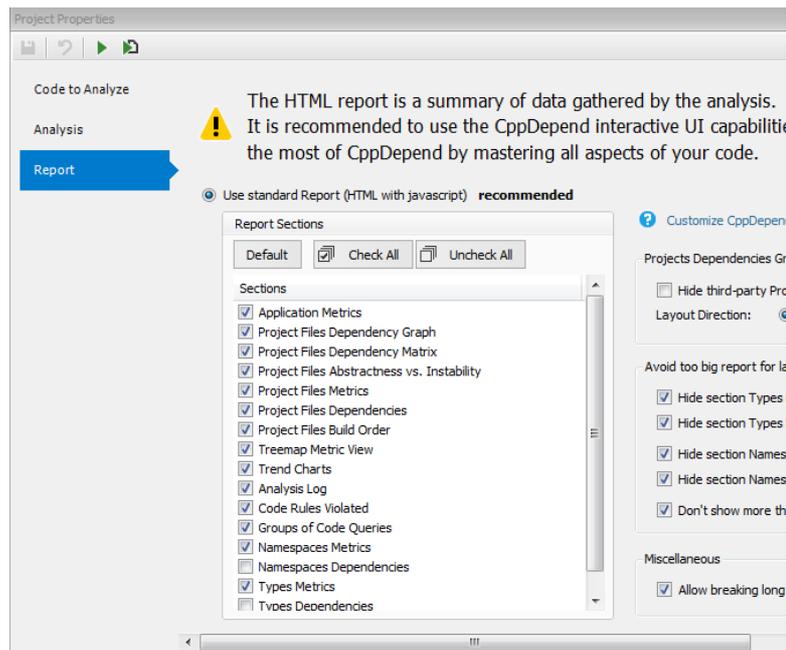
Getting Started with CppDepend

CppDepend itself comes with 200 ready-made code queries that you can immediately use on your code base.

When you run the analysis on your code, you will see some color feedback on the quality. Yellow and red highlighted lines of code show you where rules are being broken, with red being more severe breaches.

Further information on CQLinq can be found on the [CppDepend website](#).

6. Using CppDepend for Reports



Reports can be customized through the VisualCppDepend or the VS addin > Project Properties > Report sub-panel. You can choose to activate many pre-defined sections like Application and Assemblies metrics, diagrams, CQLinq rules violations, and more. The generated reports are in .html format and are found in the folder CppDependReportFiles. Any reports should automatically open in your browser.

Getting Started with CppDepend

Rules summary

41 88 0

This section lists all Rules violated, and Rules or Queries with Error

- Number of Rules or Queries with Error (syntax error, exception thrown, time-out): 0
- Number of Rules violated: 88

Summary of Rules violated

 Rules can be checked live at development-time, from within Visual Studio. [Online documentation.](#)

 NDepend rules report too many flaws on existing code base? Use the option **Recent Violations Only!**

 Some Critical Rules are violated. Critical Rules can be used to break the build process if violated. [Online documentation.](#)

Display 25 records

Q

Name	# Matches	Elements	Group
 Types too big - critical	4	types	Code Quality
 Quick summary of methods to refactor	341	methods	Code Quality
 Methods too big	74	methods	Code Quality
 Methods too complex	20	methods	Code Quality
 Methods potentially poorly commented	96	methods	Code Quality
 From now, all methods added or refactored should respect basic quality principles	36	methods	Code Quality Regression
 From now, all types added or refactored should respect basic quality principles	32	types	Code Quality Regression
 From now, all types added or refactored should be 100% covered by tests	97	types	Code Quality Regression
 Avoid decreasing code coverage by tests of types	1	types	Code Quality Regression
 Avoid making complex methods even more complex (Source CC)	18	methods	Code Quality Regression
 Avoid making complex methods even more complex (IL CC)	13	methods	Code Quality Regression

In addition, the reports will give a summary of all the main rules violations. To customize which violations you want included in the report, you can check or uncheck the box next to each one as well as choose how the violation will be displayed in the report. You can also set which rules are to be considered critical.

Note: It is recommended to uncheck the Type Metrics and Type Dependencies when generating reports on code bases containing more than a thousand types. This will cause the generated report to be very large. It is recommended that you use the Interactive UI to browse metrics and dependencies instead.

Getting Started with CppDepend

Here is what the code rule violation part of the report looks like:

The screenshot shows the CppDepend interface. On the left is a navigation menu with various code quality rules, each with a colored box containing a number of violations. The 'Methods too complex' rule is highlighted in yellow. The main window displays the rule's definition in XML and SQL, followed by a table of 20 methods that violate the rule.

```
<Name>Methods too complex</Name>
warnif count > 0 from m in JustMyCode.Methods where
  m.CyclomaticComplexity > 20 ||
  m.ILCyclomaticComplexity > 40 ||
  m.ILNestingDepth > 5
orderby m.CyclomaticComplexity descending,
  m.ILCyclomaticComplexity descending,
  m.ILNestingDepth descending
select new { m, m.CyclomaticComplexity,
  m.ILCyclomaticComplexity,
  m.ILNestingDepth }
```

methods	Cyclomatic Complexity (CC)	IL Cyclomatic Complexity (ILCC)	IL Nesting Depth	Full Name
IsPlatformSupported(String)	40	40	5	NUnit
ObjectsEqual(Object, Object)	32	45	3	NUnit
GetDisplayString(Object)	31	38	2	NUnit
Execute(ConsoleOptions, ResultSummarizer&)	28	43	4	NUnit
BestCommonType(Type, Type)	28	30	2	NUnit
Load()	26	29	7	NUnit
CheckTestMethodSignature(TestMethod, ParameterSet)	23	31	5	NUnit
EscapeControlChar(Char)	18	18	6	NUnit
GetDataFor(ParameterInfo)	17	24	6	NUnit

You can clearly see the rules on the left, along with how many violations were found (in yellow and red). Clicking on a code rule on the left, gives you additional information about the violations in the information window on the right.

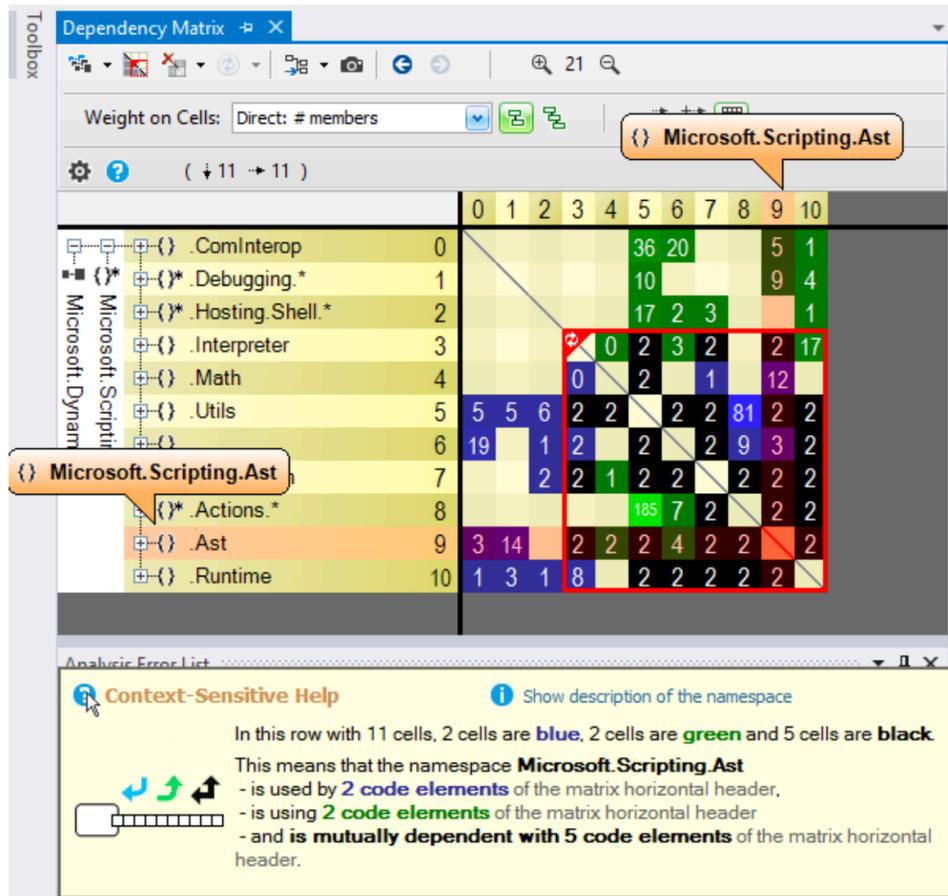
The DSM is a grid of colored cells with numbers inside of them. Each color, blue, green, and black, mean a different thing and are defined via a context sensitive help system when hovered over with your mouse cursor. If the cell has a number, this is the number of couplings between the objects in the same row and column.

By clicking on a cell, you will see a dependency graph of that object.

7. CppDepend's Dependency Matrix

The DSM (Dependency Structure Matrix) allows users to quickly get a broad overview of the dependencies and couplings that exists within their code base and displays structural patterns at a glance. It is different from a graph since we found that a simple dependency graph does not scale well with larger assemblies of code. Graphs are very good at displaying information about smaller code bases but get confusing the bigger the base is.

Getting Started with CppDepend



8. What does the CppDepend report tell me about my code?

- **Application Metric:** This section gives you an idea of the topology of your application.
- **Project Metric:** This section gives you an idea of the size of each project within your application in terms of code line and others metrics.
- **VisualCppDepend View:** This section illustrates the size of your projects, namespaces, types and methods in terms of code line. You can browse this view dynamically by clicking the *Launch Visual CppDepend on this Project* icon in the CppDepend.Project UI.
- **Project Abstracness vs. Instability:** This section illustrate the Abstracness/Instability principle explained in the [projects metrics](#) section.
- **Project Dependencies:** This section shows all dependencies between project of your application in a table.
- **Project Dependency Diagram:** This section shows all dependencies between projects of your application in a diagram
- **Project build order:** This section gives you one of the possible build order for your projects. If a cycle exists in your projects dependencies graph, this section will report it.

Getting Started with CppDepend

- **CppDepend Information and Warnings:** This section gives you feedback on your code:
 - It warns you when a project depends on a less stable project than itself.
 - It warns you when the visibility of a type or of a member is not optimal (in the context of the analyzed application).
 - It warns you when a type or a member is not used (in the context of the analyzed application).
- **CQLinq Queries and Constraints:** This section reports default CQLinq constraints which have been violated. More about the CQLinq language and CQL constraints [here](#). You can build your own CQLinq constraints with **VisualCppDepend**.
- **Type Metrics:** This section recaps type metrics in a table. A link to the documentation is provided for each metric.

9. I want to go further to have a better control over my code

- Understand [metrics](#) yielded by CppDepend and read recommendations.
- Get a new understanding of your code by playing with the CppDepend panels. Modify the view with the Level and Metric combo boxes.
- Tune your report thanks to the Output panel in the CppDepend.Project UI.
- Learn [CQLinq](#), adapt existing CQLinq code rules and create your own ones.
- Integrate the CppDepend report in your daily [build process](#).

Further Information Can be Found On Our Website

<http://www.cppdepend.com>