



How to Get Set Up and Running with NDepend

Whether you have purchased or downloaded the trial of NDepend, 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 NDepend as quickly as possible.

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

NDepend has been in development for over 8 years, and with every version we take into account user feedback and requests. For any additional help, reach out to us at support@ndepend.com, go to our [User Voice](#) to leave a suggestion, or reach out to us on Twitter [@ndepend](#).

Table of Contents

- Installing the NDepend Extension for Visual Studio2
- Analyzing a Visual Studio Solution.....3
- Analyzing .NET applications using Visual NDepend5
- Using CQLinq with NDepend6
- Using NDepend for Reports.....7
- NDepend's Dependency Matrix9
- What does the NDepend report tell me about my code? 10



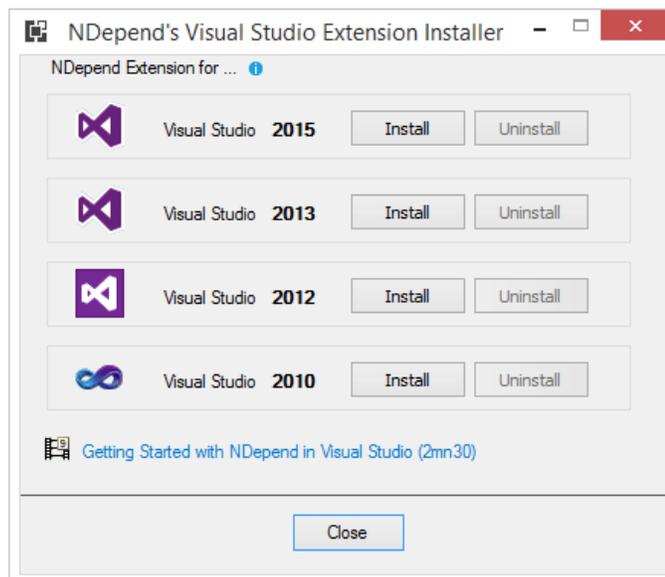
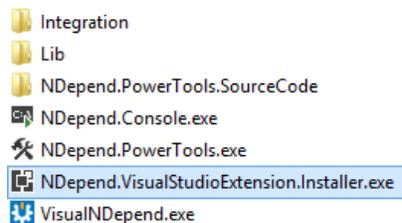
Getting Started with NDepend

Installing the NDepend Extension for Visual Studio

1. Download the NDepend installation .zip file
2. To install NDepend, just unzip the files in a private folder on your machine

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

3. If you are running the NDepend Professional Version, copy your license file into the same folder where you unzipped the NDepend files
4. Start **NDepend.VisualStudioExtension.Installer.exe**



5. Click the "Install button" for your version of Visual Studio
6. Start Visual Studio.



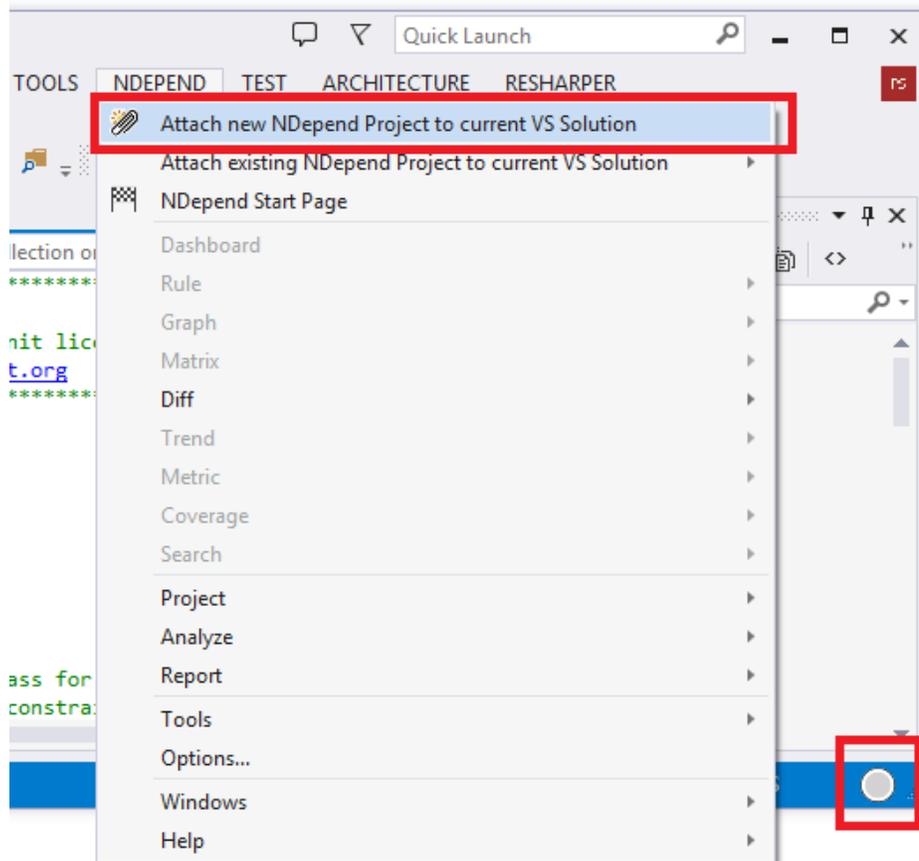
Analyzing a Visual Studio Solution

1. Start Visual Studio after you have installed the NDepend add-on
2. Open the Visual Studio solution containing the code to analyze
3. Click the grayed-out NDepend circle status icon in the bottom right corner of the Visual Studio window

OR

In the menu, click *NDepend > Attach new NDepend project to current Visual Studio solution*

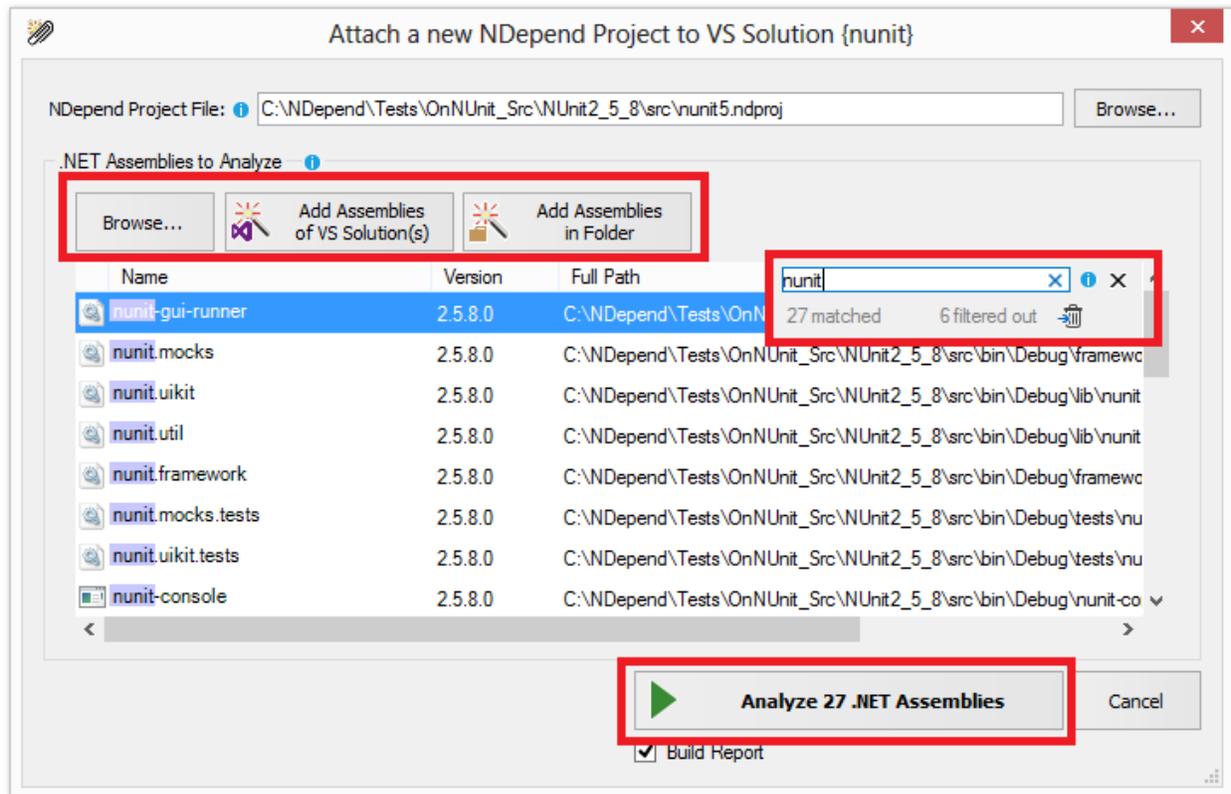
Note: An NDepend project can be attached to multiple Visual Studio solutions. This is useful if you wish to analyze several .NET assemblies compiled with several Visual Studio solutions. In addition, it allows one to navigate across several Visual Studio solutions opened in several instances.





Getting Started with NDepend

The open dialog below shows the .NET assemblies of the Visual Studio solution. Here you can add more .NET assemblies compiled from other solutions.



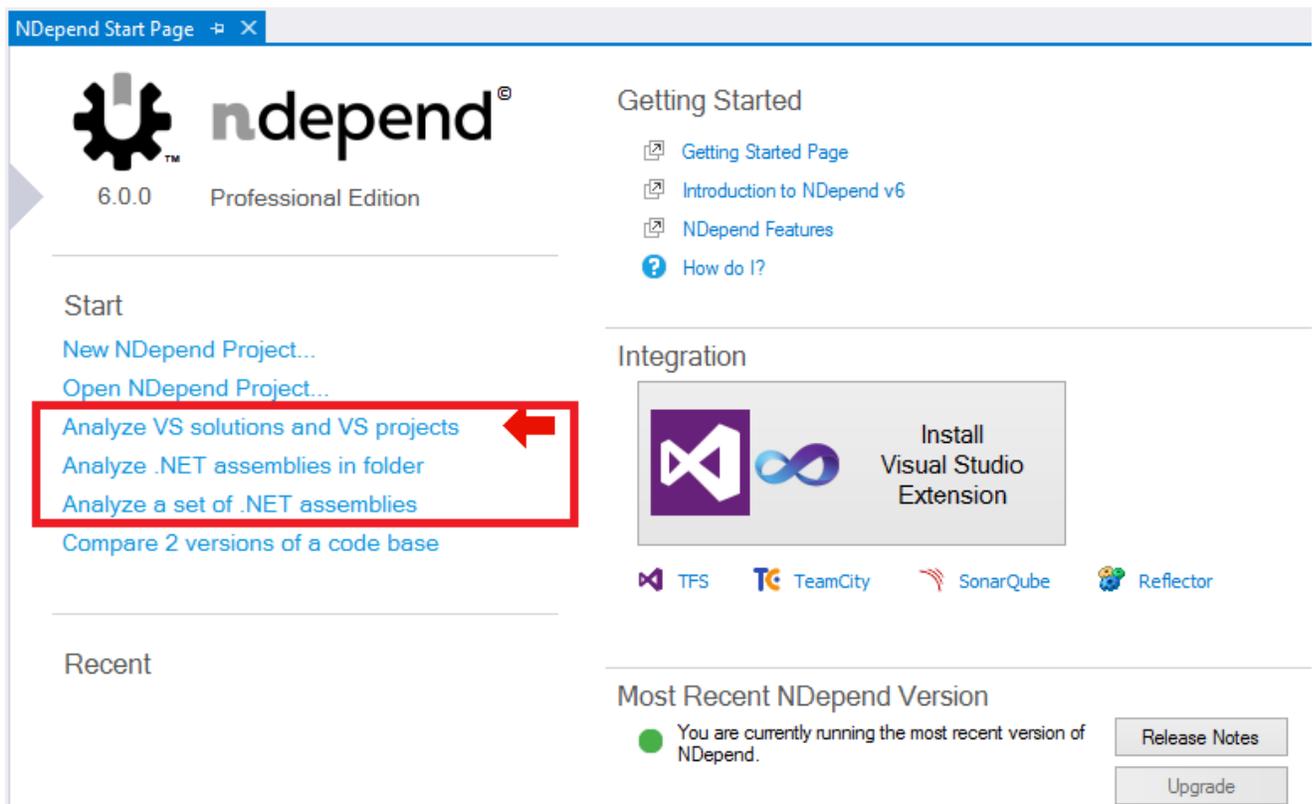
Click "Ok" and the NDepend analysis will start. It will just take a few seconds to analyze your code base for the first time.



Getting Started with NDepend

Analyzing .NET applications using Visual NDepend

1. Start *VisualNDepend.exe*
2. Select one of the options:
 - Analyze VS solutions and VS projects
 - Analyze .NET assemblies in folder
 - Analyze a set of .NET assemblies



Click "OK". It will trigger the analysis of your assemblies and source code by *NDepend.Console.exe*. This analysis will take a few seconds to a few minutes depending on the size of the assemblies. Once finished, the result is displayed in the *VisualNDepend.exe* UI.



Using CQLinq with NDepend

CQLinq is a query language based off of the C# LINQ syntax and is a critical part of NDepend 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. NDepend 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 [NDepend website](#).

Additional CQLinq resources can be found online here:

[Scott Hanselman created a NDepend Metric Placemat](#)

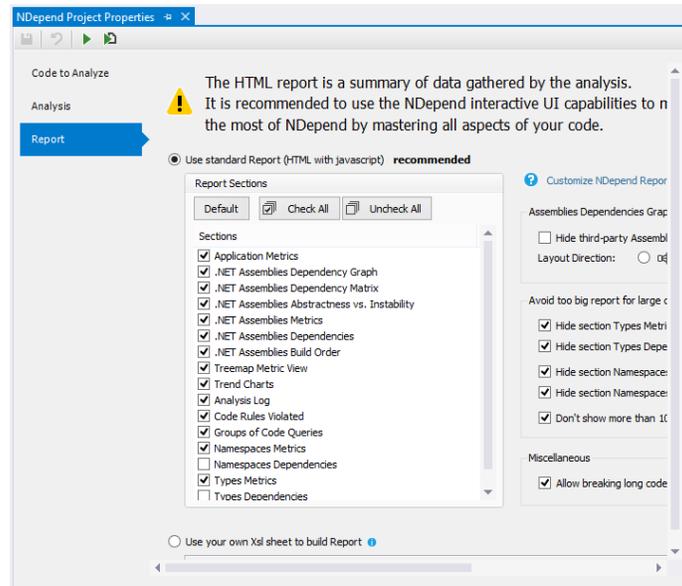
[Erik Dietrich has started a blog series on making a metric on the NDepend blog](#)

[How to Measure Module Coupling and Instability by Zorvan Horvat](#)

or check out the discussions on StackOverflow about [CQLinq](#)



Using NDepend for Reports



Reports can be customized through the *VisualNDepend* 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 *NdependReportFiles*. Any reports should automatically open in your browser.

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

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



Getting Started with NDepend

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.

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

The screenshot shows the NDepend interface with a sidebar on the left containing various code quality rules. The 'Code Quality' rule is highlighted, showing 4 violations. The main window displays a warning for 'Methods too complex' with a yellow warning icon. Below the warning, there is a code snippet in SQL-like syntax defining the rule and listing 20 methods that match the criteria. A table below the code lists these methods with their respective Cyclomatic Complexity (CC), IL Cyclomatic Complexity (ILCC), and IL Nesting Depth.

methods	Cyclomatic Complexity (CC)	IL Cyclomatic Complexity (ILCC)	IL Nesting Depth	Full N
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

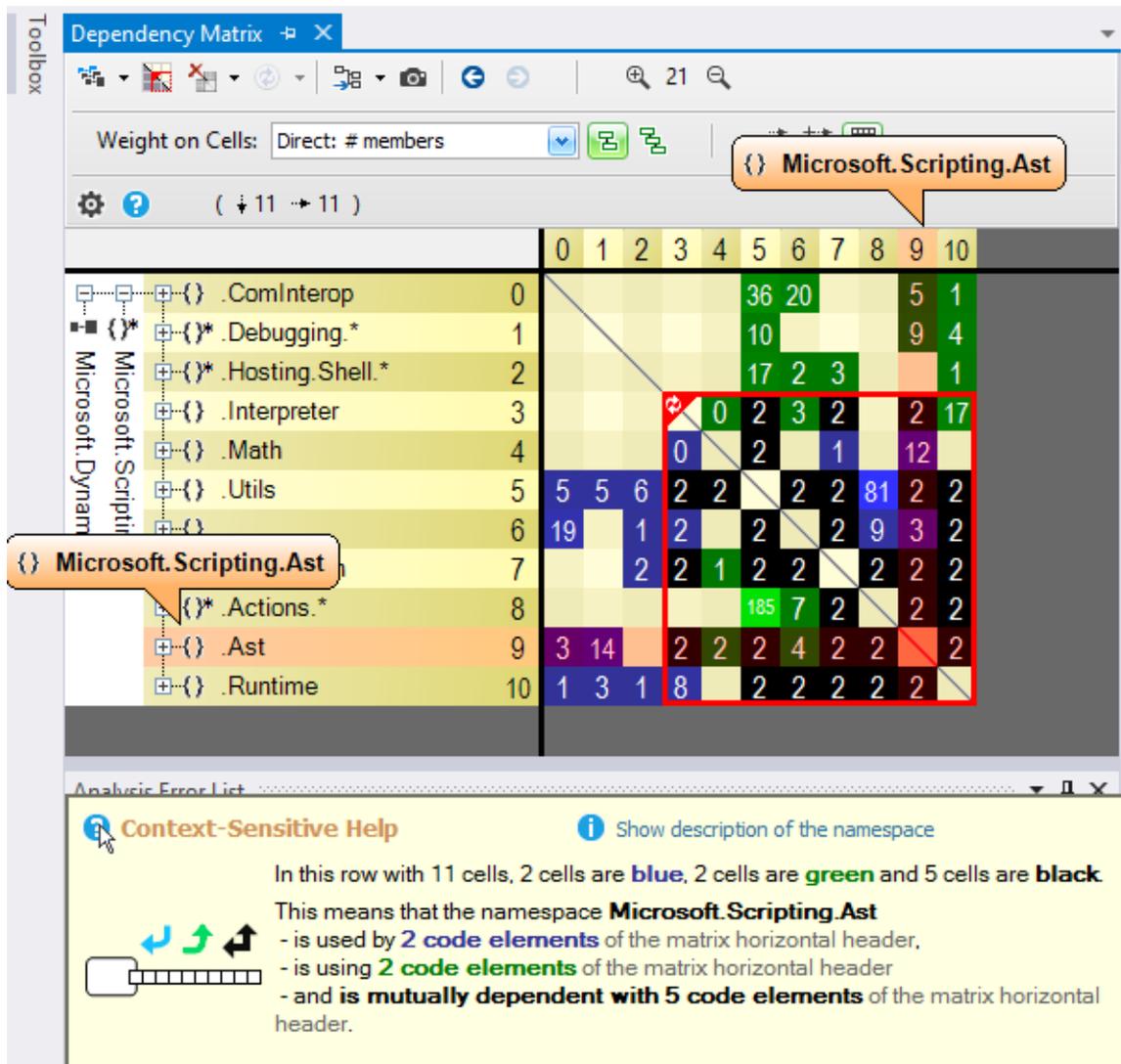
You can clearly see the rules on the left, along with how many violations were found (in yellow and red). Clicking on a code rule on the left, gives you additional information about the violations in the information window on the right.



Getting Started with NDepend

NDepend'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.



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 an cell, you will see a dependency graph of that object.



What does the NDepend report tell me about my code?

- **Application Metric:** This section gives you an idea of the topology of your application.
- **Assembly Metric:** This section gives you an idea of the size of each assembly within your application in terms of #IL instructions and other metrics.
- **VisualNDepend View:** This section illustrates the size of your assemblies, namespaces, types and methods in terms of #IL instructions. You can browse this view dynamically by clicking the *Launch Visual NDepend on this Project* icon in the NDepend Project UI.
- **Assembly Abstractness vs. Instability:** This section illustrates the Abstractness/Instability principle explained in the [assemblies metrics](#) section.
- **Assembly Dependencies:** This section shows all the dependencies between the assemblies of your application in a table.
- **Assembly Dependency Diagram:** This section shows all the dependencies between the assemblies of your application in a diagram.
- **Assembly Build Order:** This section gives you one of the possible build orders for your assemblies. If a cycle exists in your assemblies' dependencies graph, this section will report it.
- **NDepend Information and Warnings:** This sections gives you feedback on your code:
 - It warns you when an assembly depends on a less stable assembly 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 Rules:** This section reports default CQLinq rules which have been violated. More about CQLinq [here](#).
- **Type Metrics:** This section recaps the type metrics in a table. A link to the documentation is provided for each metric.

Further Information Can be Found On Our Website

<http://www.ndepend.com>