

DotNetReport Builder

Quick Start Guide



dotnet Report Builder – an Overview



[dotnet Report Builder](#) allows .Net developers to add Do It Yourself Ad Hoc reporting feature to their software. This means that your end users can quickly and easily build custom reports. While some software may allow users to generate reports, those reports are often confined to a handful of different templates. With .Net Report Builder, end users decide what fields are added to the report? How it's formatted? and much more with a very friendly and intuitive Report Wizard. This is all done from inside your application. There's no need to export data or run another program.

Who Needs Report Builder?

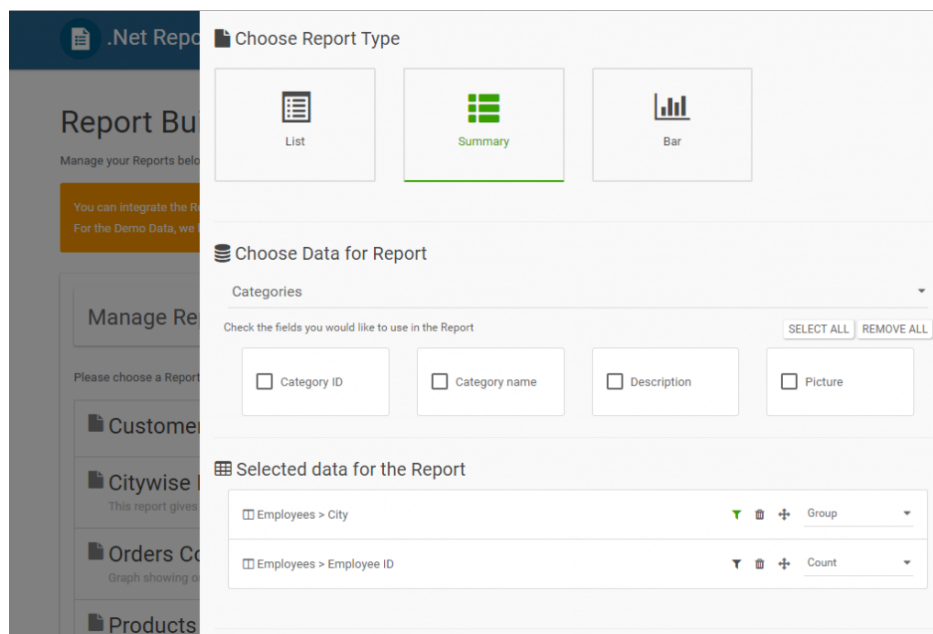
This software is for any Programmer or Software Company using ASP .Net Web Technology. It's especially useful for SaaS providers in accounting, record keeping, CRM, e-Commerce and other database solutions. Specifically where users will often want to create different reports that represent different types of data. For example, in accounting, a user may need to create reports for income, expenses, profit and loss within a specific time frame, income from specific sources, and other custom reports.

What Does it Require?

Incorporating .Net Report Builder into an existing software takes very little time or coding. If the requirement is to build a standalone app for reporting only, that's even simpler. First, the developer needs to set up the database schema in .Net Report Builder, which is easily done using the admin tool. Next, the developer has to setup the front end in an existing Visual Studio Project or a new one by simply adding .Net Report Builder's [nuget package](#) contains the Controller, View and Script files will be added directly to the Visual Studio Solution.

The next step is adding the Api keys to the web.config file. Once that's done, you can open your application and see how Report Builder runs within it. You can easily make style, labels or other cosmetic changes if you need to.

Note that setting up Report Builder does not require our company to have access to your client's database or other information. We understand that most Companies will be especially concerned about the security of their information, but you can rest assure that we never need access to your client's data or your SQL connection.



Why Chose Report Builder over Similar Solutions?

There are two major reasons Report Builder is a great reporting solution. The first, as outlined above, is that it's very small and easy to install. It won't add any considerable bulk to your program, nor will it take up hours of your programmers' time in installing and configuring it.

The second reason Report Builder outclasses similar programs is that it's very customizable. You can incorporate Report Builder so deeply into your

application that it's virtually unidentifiable as a separate module. You can make it perfectly match everything you've done so that it doesn't stand out at all. So it goes beyond white labelling.

Advantages of .Net Report Builder

Report Builder offers many advantages over designing your own reports and templates. By allowing end users to create their own ad hoc reports, you don't have to spend development time to create new report templates for them whenever they need one. With that task off your developer's plate, you'll be able to focus on other clients and building new features or applications.

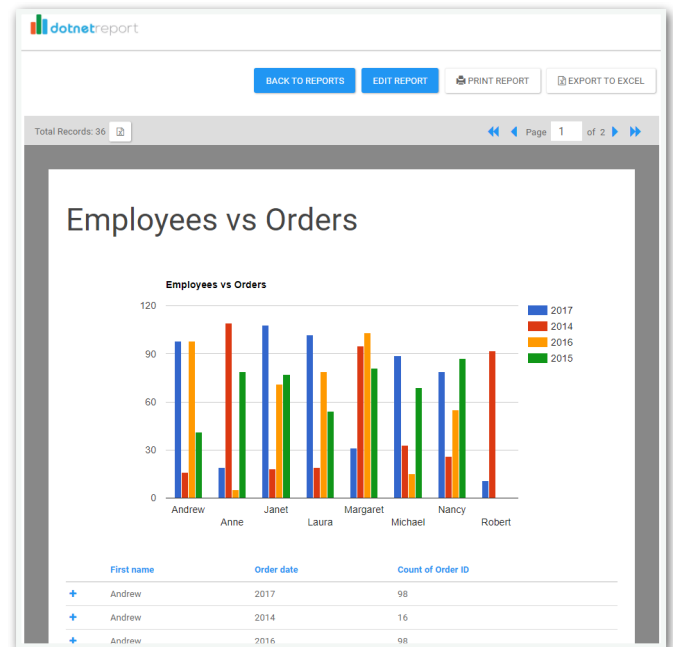
It's very user-friendly and intuitive. Even simple end users won't face much of a learning curve thanks to the [easy to use interface](#). The Report Wizard makes selecting fields and applying filters very easy. Once created, a custom report can be saved and updated or ran at any time in the future.

Getting Started with dotnet Report

Getting started with dotnet Report builder is very easy and quick. You can actually be up and running reports in less than 10 minutes!

This video explains all the steps you need to get going.

[Click here to watch the tutorial video >>](#)

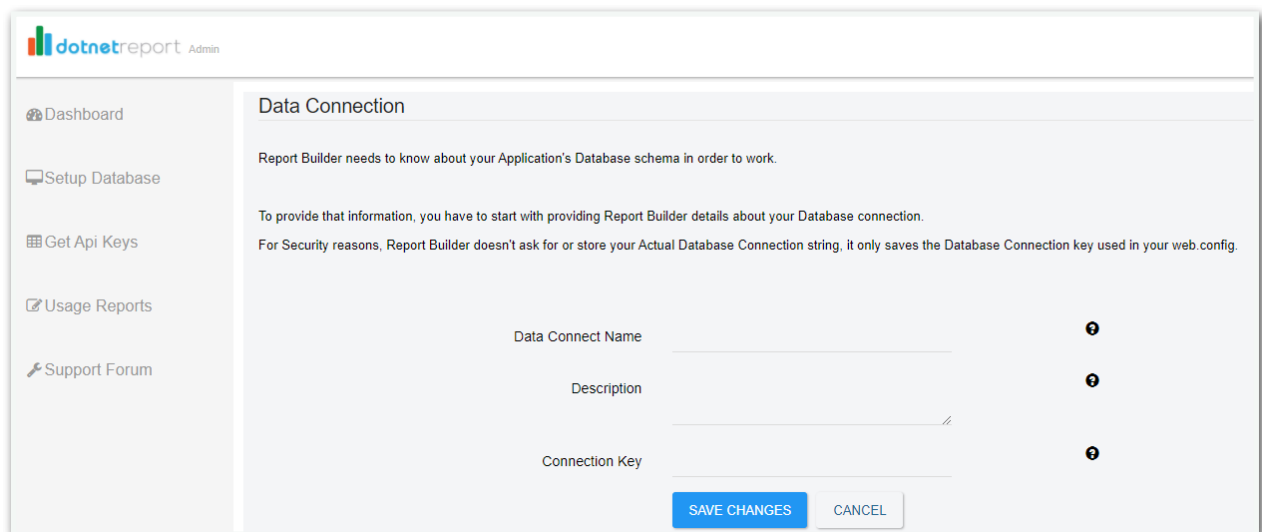


Here are the steps described in detail:

Step 1: Create an online account and Data Connection

Even though dotnet Report Builder runs all data queries and connections locally in your environment, the API Service it uses is hosted on the cloud, and you need to start with an online account. This can easily be transferred to run on-prem on your local environment later if you decide to upgrade to the Enterprise version. Creating the online account is **absolutely free** and you don't need to provide a lot of information or a credit card.

However, you do need the API keys provided in the online account to connect to the service, so head over to the [signup page](#), and create an account and log in. You would also need to add a "Data Connection" for each database you would like to use to build reports, so add one by clicking on "Setup Database".



The screenshot shows the 'dotnetreport Admin' interface. On the left is a navigation menu with items: Dashboard, Setup Database, Get Api Keys, Usage Reports, and Support Forum. The main content area is titled 'Data Connection' and contains the following text: 'Report Builder needs to know about your Application's Database schema in order to work. To provide that information, you have to start with providing Report Builder details about your Database connection. For Security reasons, Report Builder doesn't ask for or store your Actual Database Connection string, it only saves the Database Connection key used in your web.config.' Below this text are three input fields: 'Data Connect Name', 'Description', and 'Connection Key', each with an information icon to its right. At the bottom right of the form are two buttons: 'SAVE CHANGES' and 'CANCEL'.

Step 2: Install the nuget package in your Visual Studio Project

The next thing to do is open up Visual Studio and either start up a new project or open an existing one, and install the [dotnet Report Nuget Package](#). You should note that dotnet Report adds all the Controller, Views

and javascript files directly in to your application. If you are using Web Forms, install the Web Forms nuget package.

Step 3: Enter your API Keys from the online portal in your web.config.

The next thing after the nuget package is installed is to setup your API keys from the online portal in your application. To do that, you have to copy some settings in to your web.config file. The nuget package installation will add the following to your appSettings:

```
<add key="dotNetReport.accountApiToken" value="Your Public Account Api Token" />  
<add key="dotNetReport.dataconnectApiToken" value="Your Data Connect Api Token" />  
<add key="dotNetReport.privateApiToken" value="Your Private Account Api Token" />
```

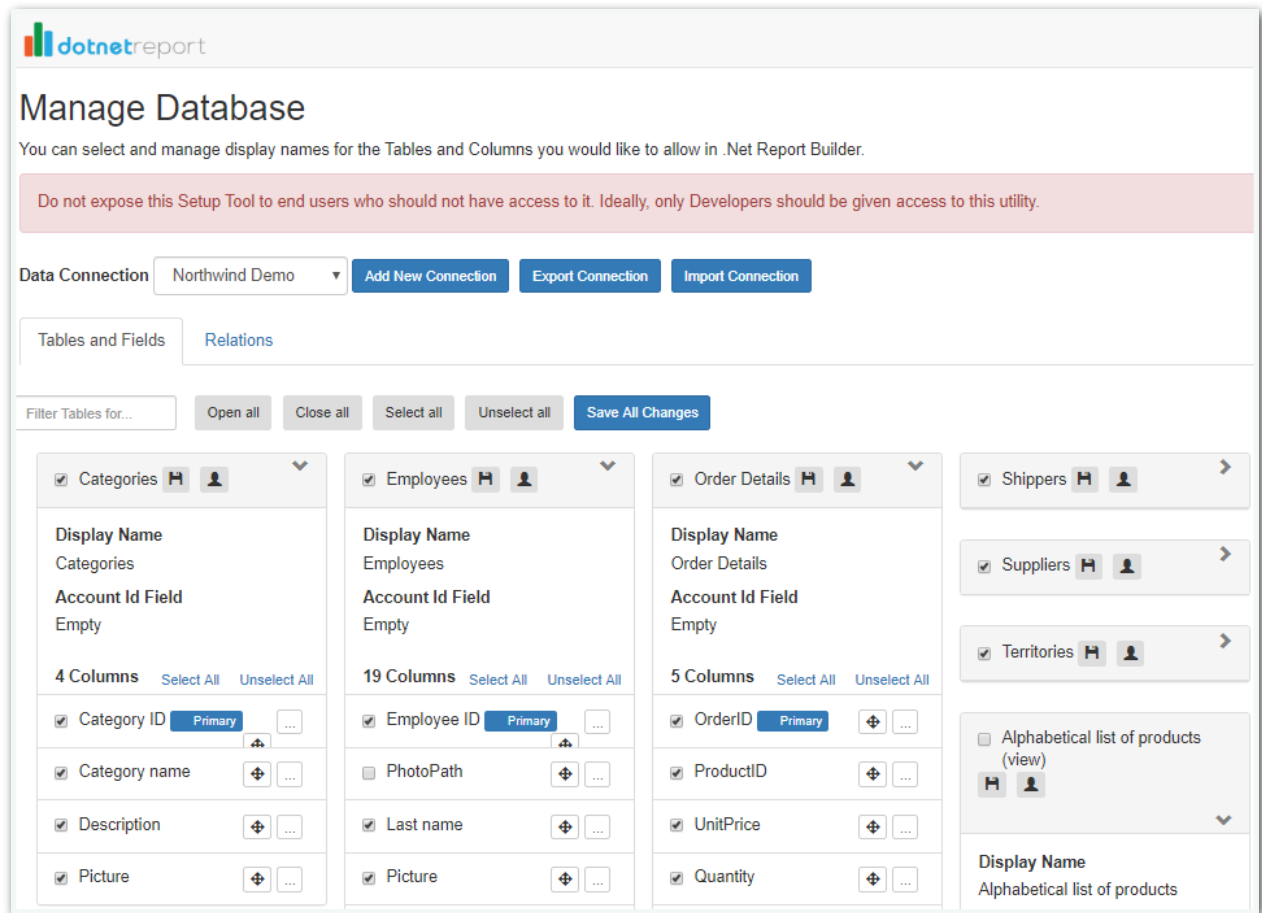
Login to your account in <https://dotnetreport.com> and click on "Get Api Keys". Locate your keys and copy and paste them in to your web.config file.

Note: You also need to add your connection string corresponding to the Data Connection you had setup.

Step 4: Setup your Database Tables and Views to be used by dotnet Report

Next step is to build and run the project, and navigate to the / **dotnetsetup** route. This will bring up the administrator/developer screen that should connect to the database you provided in web.config and list all the Tables and Views in it. Your task is to select the tables, views and the columns you want dotnet Report Builder to use to let you or your end users create reports. You would also have to setup the database relations. This is a pretty easy and intuitive interface, and is meant only for developers as an initial configuration. You can always come back to it and add more tables or columns.

So check the tables you want to use, and click Save.



And that's really pretty much it to get you started and going.

At this point, you should be create, save and Run Reports! You have to run the project and then navigate to the **/dotnetreport** route to navigate to the Report Builder.

There are many features to learn about and try out, so next you can read about them here in the [knowledge base](#).

Case Study: Mobilitie



Enabling Executives to create
custom reports in an Intuitive Self
Serve interface, freeing up
Developers time to pursue high value objectives

Infrastructure giants seeking flexibility with self serviced Reports find help

The Challenge

Mobilitie is an infrastructure company and works with venues and wireless carriers to ensure their customers are better connected. For the executives, the IT Team used to provide reports about all the assets managed by the company varying in terms and conditions. With the increase in business and volume of data, multiple different types of reports were needed, and the report format was customized as per user. A dedicated resource of the IT Team was required just to manually create these reports every day and thereby was not able to utilize their development skills. Also, it was nuisance for business teams to give requirements for the reports and then wait for them to be created manually. Unfortunately, IT Team did not have enough time to develop custom report in-house and at that time search for available reporting tools in market started.

The Solution

We came across dotnet Report and tested with other applications. It was wonderful. It was very easy to integrate into our existing application. We contacted Ahmed and the team at dotnet Report, and they were very helpful and gave us a detailed understanding of their complete reporting tool. It provides detailed level of reporting and all the customization features which we were looking for.

The Results

dotnet Report's self service reporting solution enables our users to create their own reports, saving 30% of our developers' time. Not only users are happy but IT team is also more productive as they don't have to create manual reports anymore. End users have more flexibility with the range of filters and different type of reports they can build. dotnet Report development team is also very co-operative and pay attention to any specific user demands. Their turnaround time for any new development is also very fast. We are pleased to work with dotnet Report.



Case Study: ProxiGroup

RaaS leaders transitioned from non-reporting platforms to full on demand Custom Reports

The Challenge

ProxiTrak Platform delivers RFID as a Service (RaaS), a model highly desired by the industry in many vertical markets. Real time tags and sensors tracking, virtual 3D environment and unique cloud architecture make our platform a real disruptive technology accelerating the Industry 4.0 revolution. However, with our complex and advanced software, it was missing adequate reporting for our customers.

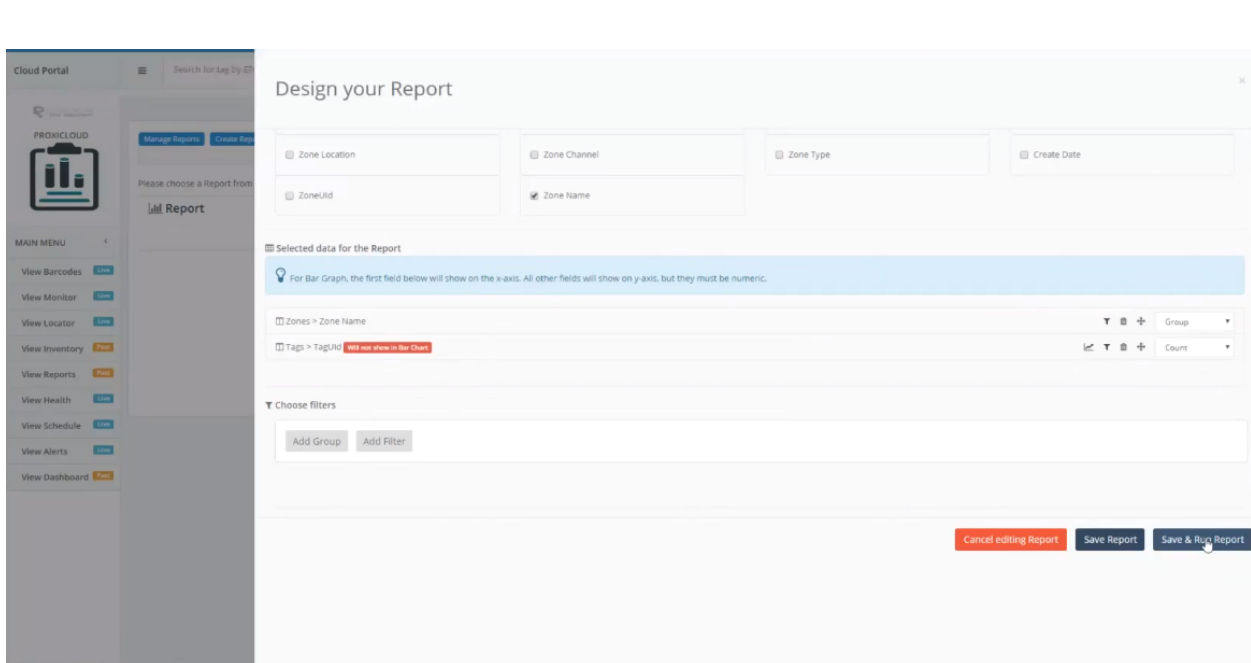
The Solution

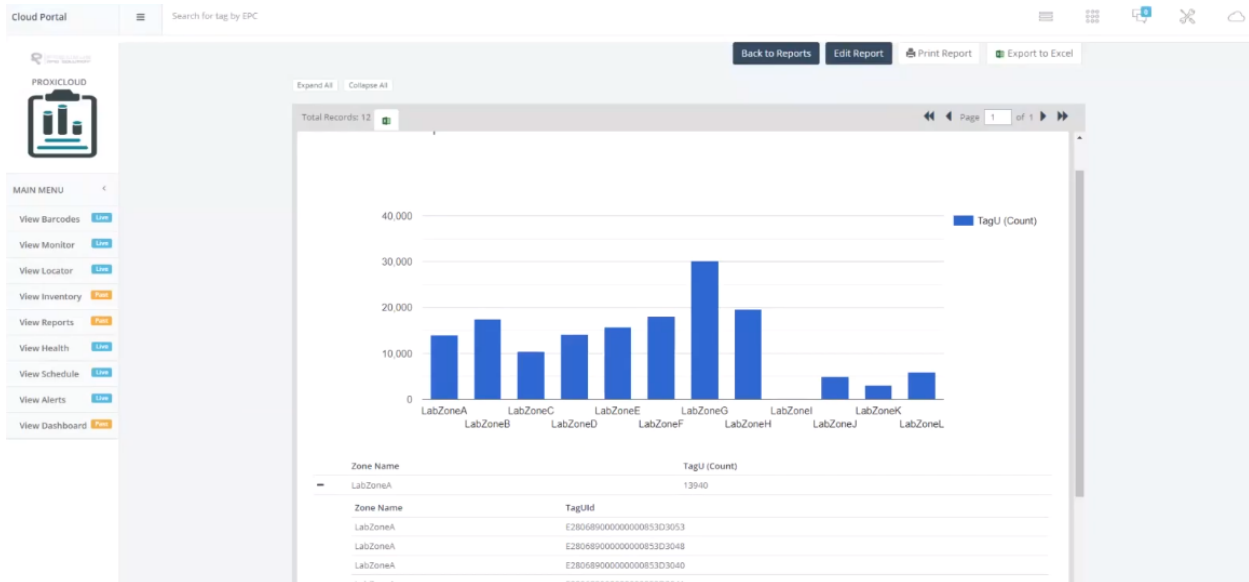
dotnet Report was able to transition us from non-reporting platforms to process improvement offers in the RFID arena with ability to negotiate our clients reporting needs very adeptly. Extremely professional, patient and a true genuine desire to help as I missed several meetings with them and it

was all ways met with benevolence. Their commitment to service level support is unbelievably superb. We will stay with this company as our long term solution for RFID/IoT software Reporting provisions.

The Results

dotnet Report improved the productivity of of our clients in the arena of IoT/ RFID Reporting Improved knowledge discovery process time and cost effective. Allowed better deterministic decisions quickly without loss of service. Eliminated inaccurate forecasting with our clients regarding RFID/ IoT asset tracking reporting.





Sign up for a Free Trial

We've been there, building Reports from scratch is frustrating for Developers. That's why we started dotnet Report, to make Reporting simple and help other Software Developers! Dotnet Report will save you precious coding hours and your users will love the ability to manage their own reports.

[Click here to try it out for free! >>](#)