A Configurable Build

Frequently, when you set up a FinalBuilder project to build a product, you actually have several closely related products to build. There are several ways to solve this problem. This tutorial demonstrates one approach, using INI files, prompts and switch statements to make one build process configurable.

First, define a short code for each distinct product. Let's say we have two products, ABC and DEF. However, sometimes ABC needs to be built with an extra sub-product. We'll call that combination ABC2.

INI File

Create an INI file with some parameters for each build:

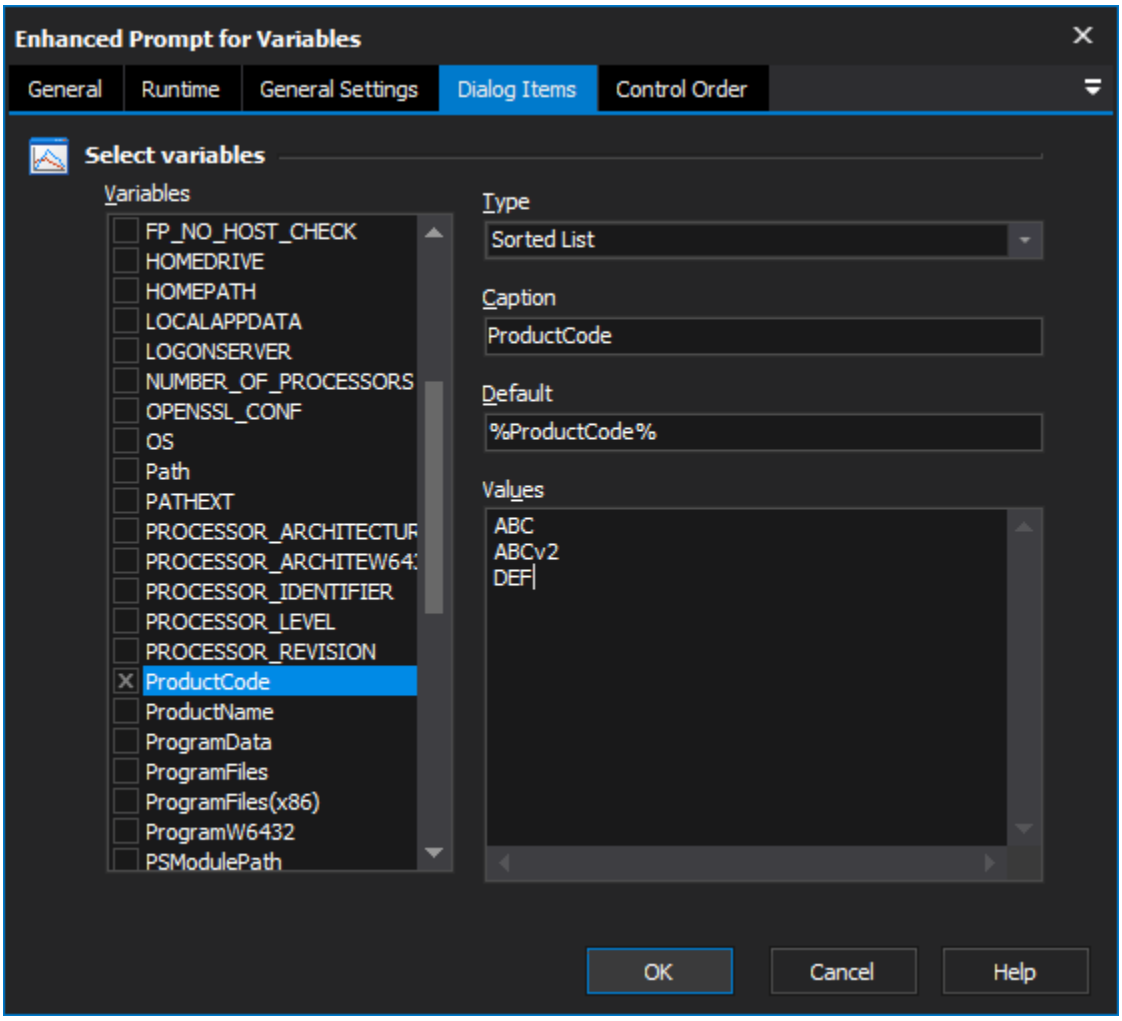
```
[ABC]

SolutionFile=ABC.sln

Description=A brilliant calculator

SVNBranch=Prod/ABC

[DEF]

SolutionFile=DEF.sln

Description=Data encryption filter

SVNBranch=Prod/DEF

[ABC2]

SolutionFile=ABC.sln

Description=A brilliant calculator PLUS

SVNBranch=Prod/ABC

BuildABC2=True
```

Variables

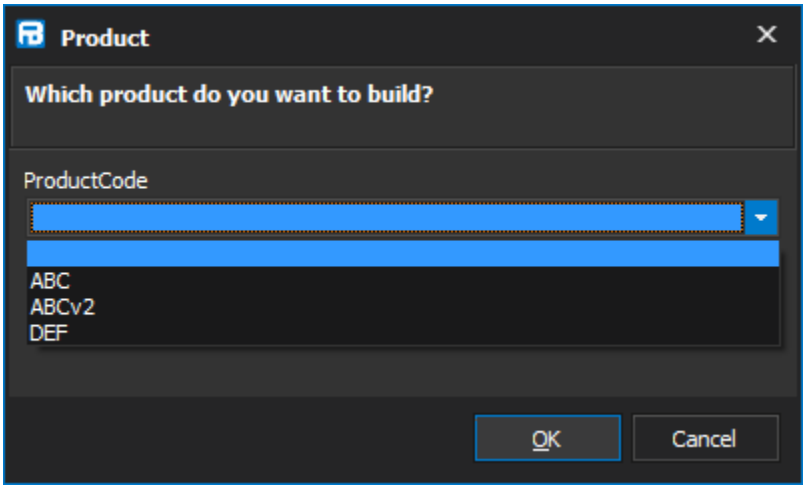
Create a "productcode" variable. Make it persistent, so that each build can default to the same type as the previous one.

Prompt

Create a "Enhanced prompt for variables" action. Here you will give the user the choice of which build to create. By using the "unsorted list" type with the current value as the default, a drop down list is shown with the current value already selected.



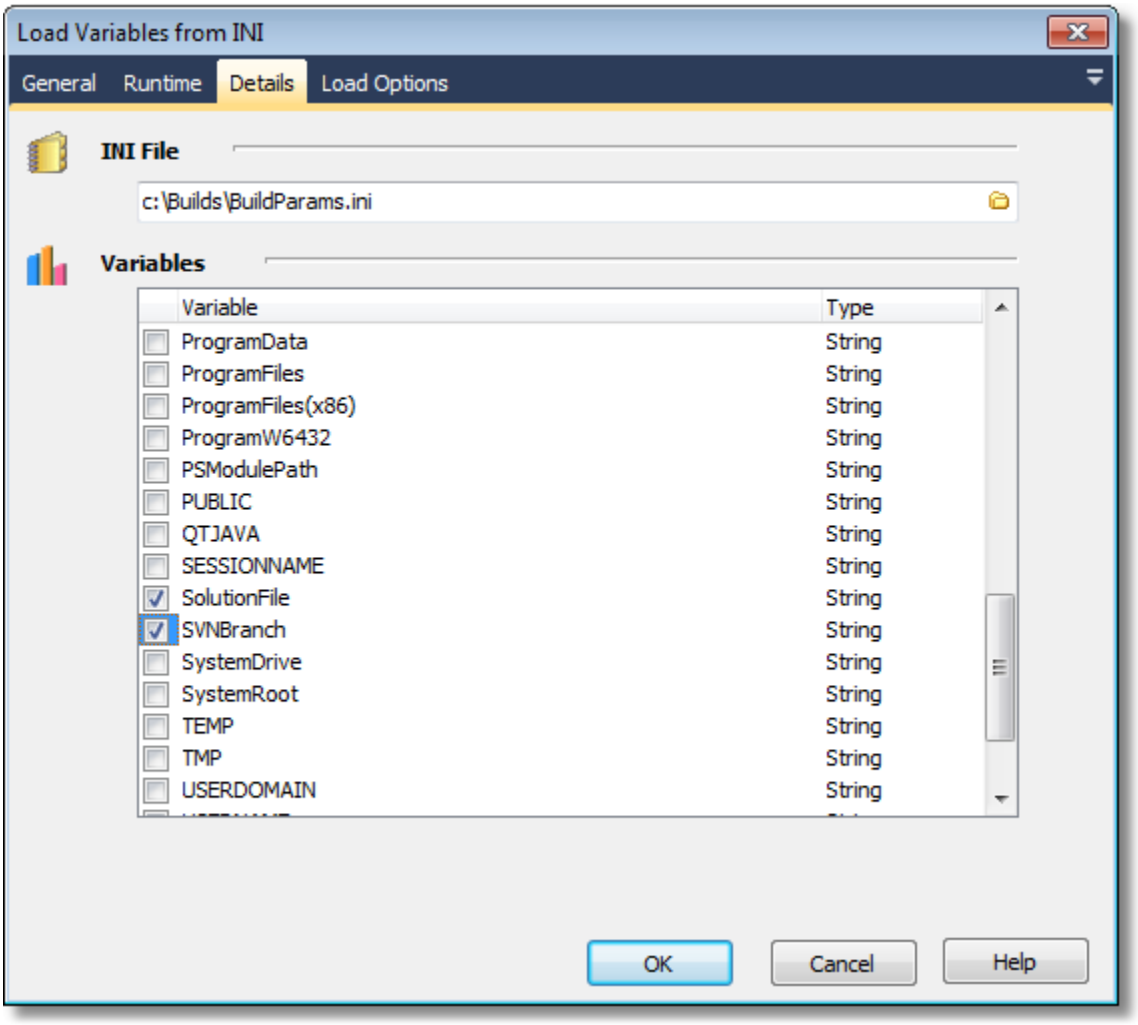
At runtime, this will look as follows:



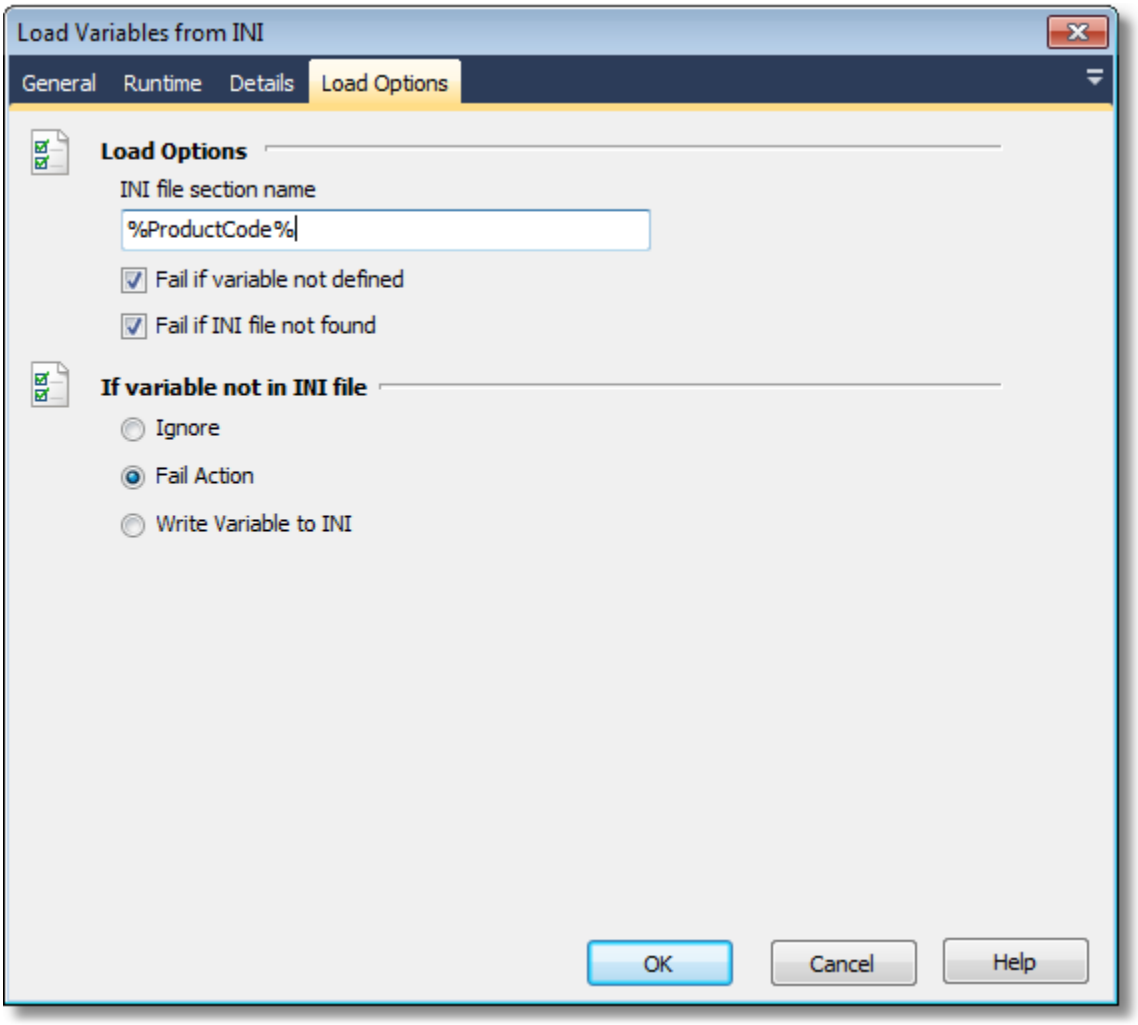
Load INI file

Next, we need to load the settings for the chosen build. We first load the mandatory settings, then the optional ones.

We create one "Load Variables from INI" action with these settings:

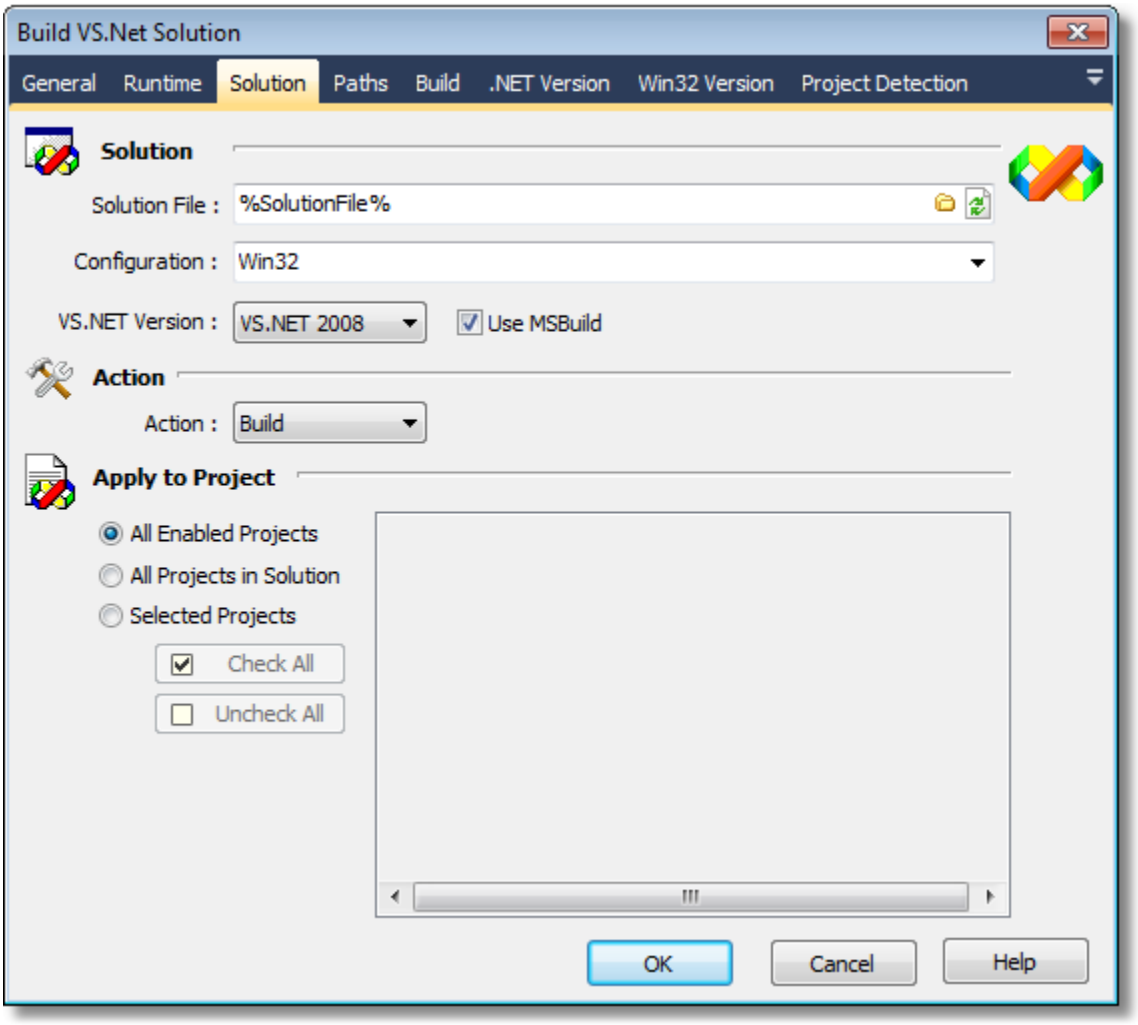


Next, we create one for the optional settings:



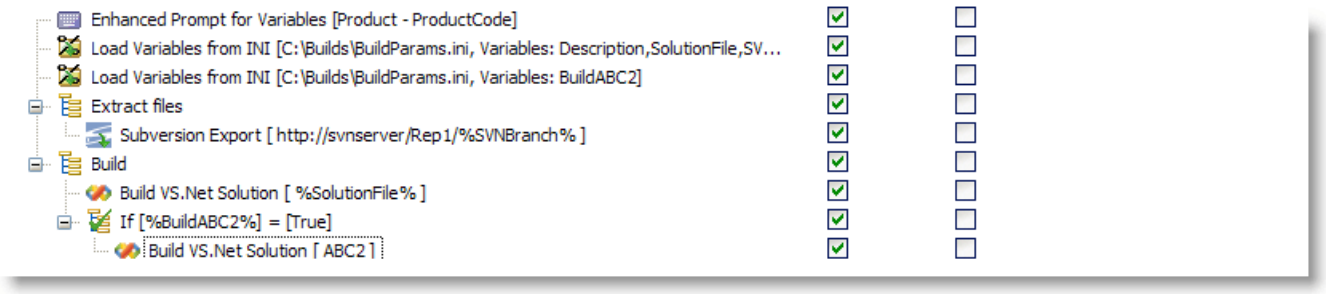
Use the variables

Now construct your build process, using these variables everywhere:



Conclusion

The overall result looks like this:



You now have a single build which is capable of building different products. All the same core logic - loading variables, extracting files from your version control system, building - is stored in only one place. This is much better than having a separate FinalBuilder project for each product. In that situation, if you found a problem in one build, you would have to fix it in every project individually, making your build process more error prone and labour intensive.