

Tools for Forms and Classes

Andrew MacNeill, AKSEL, 2006

Pssst. Let me tell you a secret. It's not a huge secret – in fact, many, if not most, FoxPro developers are aware of it already. So the secret is – almost everything in FoxPro's really a DBF. The PJX project file? It's a table (the memo field is the PJT file). The FRX report writer? Yes, that one too. Screens, menus, class designers, yes, even the database container – they are all DBF files. So what does that mean? As with most secrets, it's useless unless you do something with it. If you're involved in a long drawn out project, your focus isn't on building extra tools – so how can you use this little gem of information? You can make changes in your application without using the designers. You can search through forms and libraries just as you do with your application data. What's more, though, is that it allows tools to be written that make it easier to understand how these project pieces work. WhiteLight Computer's HackCX is one of these tools.

Let's take a form as an example. You build a form using FoxPro base classes. That may be a no-no but you were just building it to test something out. But now, it's gotten to be an actually useful form. You can use it in all of your projects but there's one thing that holds you back. All of your applications use a common class library – nothing is based on the FoxPro base classes. So what do you do? Do you rebuild your form using the common classes? While redoing something that's already done always sounds like a good way to spend one's time, what you really want to do is just change the classes of the buttons and the text box and of the entire form to use the common library. The SCX is simply a table with a structure as shown in table 1. Why not make the change directly in there?

Field Name	Type
Platform	Character – typically says WINDOWS or COMMENT
UniqueID	Character
Timestamp	Numeric
Class	Memo
Classloc	Memo
Baseclass	Memo
Objectname	Memo
Parent	Memo
Properties	Memo
Protected	Memo
Methods	Memo
Objcode	Memo
Ole	Memo
Ole2	Memo
Reserved1-7	Memo
User	Memo

Table 1 – The structure of the SCX file. I love useful field names like Reserved6 and Reserved7 – don't you?

Platform	Uniqueid	Timestamp	Class	Classloc	Baseclass	Objname	Parent	Properties	Protection
COMMENT	Screen		memo	memo	memo	memo	memo	memo	memo
WINDOWS	_1NTOFBQRQ	862861714	Memo	memo	Memo	Memo	memo	Memo	memo
WINDOWS	_1NTOFBQRR	862862839	Memo	memo	Memo	Memo	memo	Memo	memo
WINDOWS	_1NTOFBQRS	862862839	Memo	memo	Memo	Memo	memo	Memo	memo
WINDOWS	_1NTOFBQRT	862861714	Memo	memo	Memo	Memo	memo	Memo	memo
WINDOWS	_1NTOFBQRU	862861714	Memo	memo	Memo	Memo	memo	Memo	memo
WINDOWS	_1NTOFBQRV	862861714	Memo	memo	Memo	Memo	memo	Memo	memo
WINDOWS	_1NTOFBQSB	862861714	Memo	memo	Memo	Memo	memo	Memo	memo
WINDOWS	_1NTOFBQSC	862861714	Memo	memo	Memo	Memo	memo	Memo	memo
COMMENT	RESERVED		memo	memo	memo	memo	memo	Memo	memo

Figure 1 – Browsing an SCX file.

Type USE myform.scx and browse the table (see figure 1). Change the Class and ClassLoc fields to match that of your class library. Change the ObjectName field of some of the buttons from Command1 to a more usable name like cmdSave. Close the Browse window. Type USE to close the form and then re-open it. Your form now uses the same class library your other applications use.

If you had made a typing mistake when naming the class or class location, FoxPro would have either prompted you for the proper location or returned an error, making the form unusable until you corrected it. If your form used containers and you changed the name of one of the larger containers, FoxPro might have lost its way and not known what objects they should have belonged to, instead providing you with a nasty error message: Error loading file.



Figure 2 – Error Loading File.

This same scenario can apply to refactoring projects, the process in which you rework code to make it more understandable or maintainable. Changing the name of a commonly used class can wreak havoc on larger projects where those changes have to be made everywhere else, often by hacking the class or forms. As a result, many developers choose not to refactor, creating a legacy of unfriendly or misleading names in their projects.

As with many things, being able to change things "under the hood" can also lead to some major problems. Consider what might happen if you did a REPLACE ALL instead of just a single REPLACE. Or if you accidentally typed ZAP and wiped out the entire file. When you make changes in a form or class library, there is no backup or buffer

– so those changes are instantly committed, regardless of your intention. That's why it's great that there is a tool like HackCX from White Light Computing that lets us go under the hood to make these types of changes, automatically creates backups and presents a much easier view of the two most commonly used pieces of a FoxPro project, the screen and the class library.

What is HackCX?

HackCX opens a class or form file as a table and provides a usable display for each record, making sense of the various memo fields we previously dealt with in the BROWSE. Call HackCX4 and it prompts for a class or form file. It then displays it. You can also call DO HACKCX4 WITH "XXX.SCX" and it will open it directly.

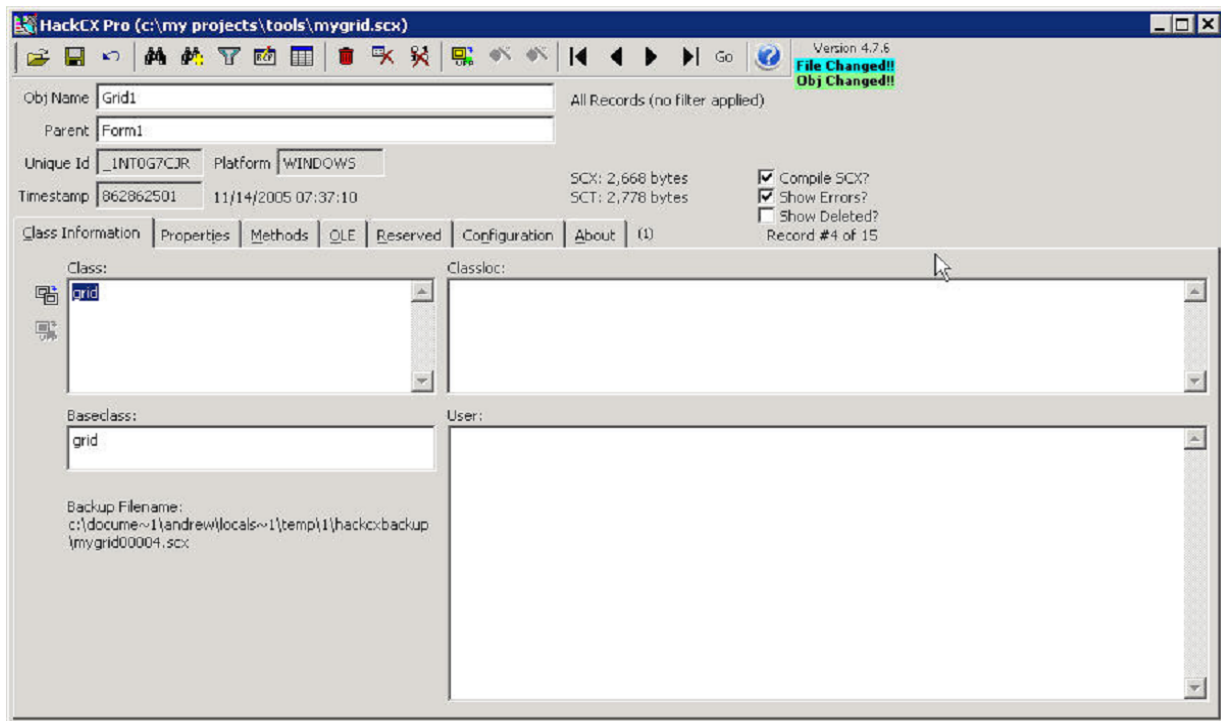


Figure 3 – The HackCX display. Note how changes are noted in the upper right hand corner.

Using our previous example of wanting to change the class of an object on the form, click the arrows to move back and forth through the table or click the Find button and specify the name of the object you are looking for. As soon as you make a change on the record, HackCX does two things: 1) it notes that there are changes and allows you to undo them and 2) it creates a backup in case you do make any mistakes. Each change is buffered. Clicking the Undo button undoes all of the changes. Right-click on the Undo button and it only undoes the most recent change.

Changing a class is easy – type in the name of the desired class in the Class Field and the name of the class library in the ClassLoc field. If typing isn't your style, click the Redefine button right beside the class box and you can specify it using the standard class specifier dialog.

This is where HackCX shines – since it was written by developers for developers, it notes the little things that sometimes can easily break a form. For example, I chose to re-define a text button and accidentally chose a command button class. Instead of immediately accepting my change, it noted that I wasn't just redefining the class, I was changing its entire type.

HackCX may have been written to deal with the most common type of "hack" that developers need to make when browsing a form or a class – changing of the name – but that's not all it does. Click on the Properties tab and you can review or change the settings for each object. Click the Methods tab and review the actual code. Perhaps most importantly, HackCX also brings sense to all of those reserved fields. Click the Reserved tab and each field is labeled with its actual purpose. If you view a class library with HackCX, you'll see that Reserved7 field is where the description of the class is stored and Reserved3 contains a list of all the custom properties and methods.

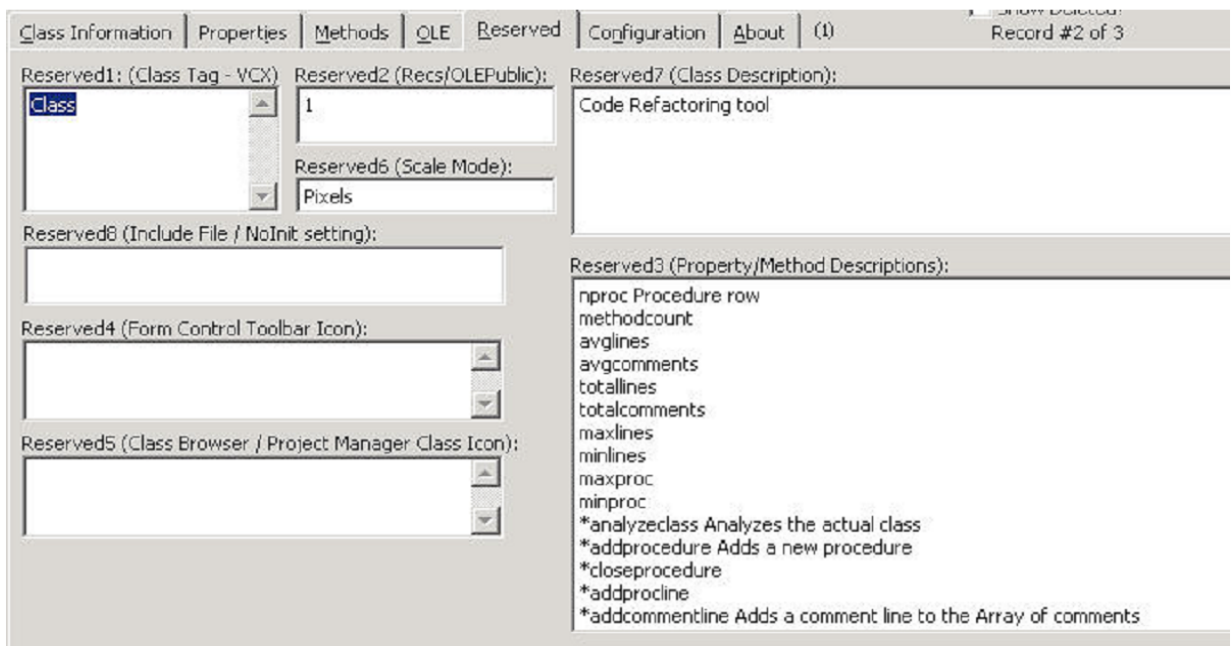


Figure 4 – Making sense of reserved fields. HackCX sheds some light as to the mysteries of the Reserved fields.

If you're really not sure of what a particular field can be used for, hit F1. The Help in HackCX is first-rate. Not only is each function explained, but every field is documented as well as a great section aptly named "How to Hack Techniques".

Why Changing Methods Doesn't Change the Code

Whenever an object has code behind it, the Methods field is populated. If you hack the file with a BROWSE statement, you can open the Methods memo field and make changes directly to the code. However, if you attempt to run your form immediately after making those changes, you won't see them. This is where the Objcode memo field comes in. When you use the Form or Class designer and issue a save, FoxPro invisibly compiles the code and places it into the Objcode memo field. If you manually make the change, you need to type `COMPILE FORM myform.scx` or `COMPILE FORM classlib.vcx` to update the Objcode field.

On the main HackCX page, there are three checkboxes: Compile VCX, Show Errors and Show Deleted. HackCX will automatically compile changes in the methods when you save it if the Compile VCX box is checked. When you compile code, FoxPro also generates an ERR file if the code has any errors. These errors can be instantly displayed with the Show Errors checkbox.

The Show Deleted option lets you further understand how FoxPro works with classes and forms. Whenever you make changes to a class in the class designer, and then save it, FoxPro deletes the old record and then adds a new one to the class library. As a result, if you are browsing a VCX file, you may see lots of deleted records, all containing older pieces of code. When you explicitly issue the COMPILER FORM command, it packs the tables, removing any deleted records. By default, HackCX only shows you the most current work contained in the Form or Class. Check Show Deleted and you can review older changes made in the code as well.

Valuable Insights and Tools for Your Code

HackCX has a variety of other useful tools. For example, with one click, you can change every class back to its native FoxPro base class. You can filter and browse the records in the library or form— especially useful when dealing with a large class library. It also lets you add new properties and methods to the form or class. When adding a new property, you can specify the name, description, default values, visibility - everything all at once. When you add new methods, it automatically puts in a ToDo reminder.

HackCX shows the file size of both the main CX file and the related memo field. This can be valuable as class libraries can tend to get extremely large as they are modified. It also demystifies the timestamp field, showing the value and then the actual date and time that the last changes were made.

The Configuration tab lets you set some standard settings such as backup folders and startup directories but it also provides a valuable team development feature. It is important to note that HackCX doesn't protect you from completely screwing up your form or class. Hacking a screen or class library isn't something that you may want everyone doing. If you work on a shared machine, you can provide a list of allowed developer logins who are allowed to use HackCX or not.

How valuable a tool like HackCX is really depends on the type of development you do. It's not expensive (\$50 per developer license) but if you build applications using frameworks and rarely spend time cleaning up or refactoring code, then it may not be something you need. But the moment you need to re-define a class or make changes under the hood, you'll wonder how you never lived without it.