# fxReports: Sharing Custom Report Features

*Cathy Pountney*
*White Light Computing*
*Whitelightcomputing.com*
*clpountney@whitelightcomputing.com*
*www.cathypountney.blogspot.com*

With regards to reports, Visual FoxPro 9 SP2 has enormous power and extensibility. It has the ability to create custom features that can be used over and over again on numerous reports. For example, would you like the ability to easily add a watermark to reports? How about the ability to dynamically reduce the font on long text so it fits within a narrow column? The possibilities are endless and no one developer can think of them all. My fxReports reporting framework and corresponding utilities allows each of us to create our own custom features to use in our applications as well as share them with the FoxPro community. And of course, the opposite is true: you can take advantage of custom report features created by other developers. This project is part of the FoxPro Community effort on VFPx (hosted on GitHub). In this session, I'll walk through the entire concept from both sides. You'll learn how to use features developed by other developers, you'll learn how to create your own features, and you'll also learn how to share those features with other developers.

# Table of Contents



## The Concept

The FoxPro Community has always been one of the most giving and sharing development communities around. When I started realizing the potential of what we as a community can do with these custom features on reports, my head started spinning with ideas. I wanted to harness the potential into a concept where we can easily share our custom features with each other.

Enter stage right … VFPx, which is hosted on GitHub (https://VFPx.github.io). This is a place where several FoxPro-related open-source projects reside. These projects have been created by other FoxPro developers who generously donated their time to the community. Many are a collaboration of several people; developers, testers, technical writers, etc. If you've never looked at this site, I encourage you to do it right now. You'll be amazed at all the tools and utilities available to you, all free of charge!

I've created a new project on VFPx called fxReports and it's the basis of this entire session. The concept is simple. I create some custom report features and upload them to VFPx. You download those features and start using them in your reports. Once you get the hang of custom report features, you start creating your own custom features. You then jump on the bandwagon and upload them to VFPx. Now I can download those features and start using them in my reports.

As more and more developers upload custom features, the list of features gets bigger and we all benefit by taking advantage of the work done by others. No more reinventing the wheel every time we need something! This is a perfect example of the FoxPro Community doing what it does best: Sharing!

## The pieces of fxReports

The fxReports concept is implemented through a number of pieces: The core framework code, a set of three utilities, the features, and VFPx. First, I'll give a simple explanation of each piece. Later, I'll give more details about each of the pieces and how to use them.

### The core framework code

The core framework code consists of a handful of classes and programs that are the foundation of the custom report features. Some of the classes are used as the foundation for implementation of a feature on the report. Other classes are used as the foundation to add that same feature to the Report Designer user interface, making it easy to use on any report. The core framework code also consists of some programs that can be tweaked and used in your application to run reports in a way that allows them to take advantage of any custom report feature.

### The utilities

There are three different utilities that work together to help you learn and use this entire concept; fxReports, Unwrap & Merge, and Wrap & Share. The fxReports utility is the main utility to run when first learning about fxReports. It systematically walks you through every

step needed to prepare your application for using custom report features. The Unwrap & Merge utility is used to grab custom features created by other developers and get them working in your reports. The Wrap & Share utility does the opposite and walks you through the process of packaging up your custom report features in a way that allows others to use them. I'll explain more details about each of these utilities later.

## The features

The features are the purpose of this entire concept. A feature is an embellishment to a report added via a Report Listener. For example, I have created a feature to automatically reduce the font size of an object to allow longer text to fit into smaller spaces. I have also created a feature to add a watermark (text or graphic) on a report. And I'm sure the list of features will continue to grow more and more as time goes on. I'll explain more details about creating features later.

## VFPx

VFPx is the public location for everything just mentioned. Using your favorite internet browser, navigate to https://github.com/VFPX/fxReports, which is the main page for the fxReports project on VFPx.

All the files necessary for the core framework, as well as the three utilities, are sitting in a subfolder called CoreFramework.

A folder named FeatureRepository contains additional subfolders, one for each developer who contributes reporting features. You can pick and choose which developer's features you wish to get. For example, my features are located in a folder named FeatureRepository/CathyPountney.

VFPx gives you the option to download files to your computer individually or you may clone files for use with Git or SVN. To support this, you'll find subfolders with all the individual files to download. In addition, for each subfolder, you'll find a zip file which contains all the same files that are in the subfolder. For developers who want to download a single file, download the zip file. For those who want to sync with Git or SVN, do so against the subfolder.

I'll explain more details about what to do with the downloaded files later.

> *NOTE: The Southwest Fox conference cd includes all the files for this session. However, I suggest you use VFPx to get the most current files and documentation as they may have been updated since the conference CD was prepared.*
>
> *ALSO NOTE: The "fxReports Documentation.pdf" file on the main fxReports page is basically this whitepaper. Most likely, it has been updated too so be sure to grab that.*

## Visual FoxPro Bug Fixes

Throughout this session, I'm using Visual FoxPro version 9.0 with Service Pack 2. That said, it's important to let you know about various hotfixes provided by Microsoft, as well as a few bug fixes provided by the FoxPro Community. If you don't have all these patches in place, you'll most likely run into some type of issue as you implement this concept in your application.

The VFPX community portal on GitHub is the first place to start (https://github.com):

- **VFP9SP2Hotfix3** (https://github.com/VFPX/VFP9SP2Hotfix3): This download contains the latest official hotfix from Microsoft (KB 968409). It's cumulative, which means it includes all the previous hotfixes, so this is the only one needed to get your Visual FoxPro installation up-to-date. To confirm whether you have this hotfix, issue `? VERSION()` in the command window and if the last 4 digits are 7423, you've got it.
- **GenMenu** (https://github.com/VFPX/GenMenu): This download contains a bug fix to GenMenu.prg. It's an obscure bug that isn't encountered very often. The issue is that it attempts to assign hotkeys to any menu item that doesn't have one defined. However, it blindly uses the first character of the prompt as the hotkey without considering whether that first character is a valid hotkey. For example, if the first character is a quote ("), semicolon (;) or bracket ([), it's not a valid hotkey and the menu ends up crashing when you run it.
- **Reporting Apps** (https://github.com/VFPX/ReportingApps): This download contains multiple bug fixes provided by various members of the Visual FoxPro Community. You can read about each of them in the README.md file on this webpage.

I've created a few other bug fixes that are included in the downloads for this session. *(I intend to get them uploaded to VFPX as soon as I have time):*

- **_GDIPlus**: The MeasureStringA method has a reference to the wrong variable, which causes it to crash when called. This can be fixed by changing the MeasureStringA method of the gpgraphics class in the _GDIPlus class library located in the FFC directory of Visual FoxPro as shown in **Listing 1** below.
- **_ReportListener**: When running reports with a ReportListener, you may encounter an issue where it looks like `SET TALK` is `ON`. This can be resolved with a change to the LoadReport method of the fxListener class in the _ReportListener class library located in the FFC directory of Visual FoxPro. Simply move one line of code as shown in **Listing 2** below.

**Listing 1**. Change the MeasureStringA method in gpgraphics of _GDIPlus.vcx.

```
case pemstatus(m.tvLayoutArea,'gdipRectF',5)  && Rect
    *{ CLP [WLC] - 06/11/2018 - BEG - Bug fix
    *!*lcRect = tvLayoutArea.GdipRectF
    lcRectF = tvLayoutArea.GdipRectF
    *} CLP [WLC] - 06/11/2018 - END - Bug fix
case pemstatus(m.tvLayoutArea,'GdipSizeF',5)  && Size
    *{ CLP [WLC] - 06/11/2018 - BEG - Bug fix
    *!*lcRect = replicate(chr(0),8)+tvLayoutArea.GdipSizeF
    lcRectF = replicate(chr(0),8)+tvLayoutArea.GdipSizeF
    *} CLP [WLC] - 06/11/2018 - END - Bug fix
```

**Listing 2**. Change the LoadReport method in fxListener of _ReportListener.vcx.

```
*-- CLP Bug Fix: Move setFRXDataSessionEnvironment before CreateHelperObjects
*!*THIS.createHelperObjects()
*!*THIS.checkCollectionMembers()
*!*THIS.setFRXDataSessionEnvironment()
THIS.setFRXDataSessionEnvironment()
THIS.createHelperObjects()
THIS.checkCollectionMembers()
*-- CLP Bug Fix: Move setFRXDataSessionEnvironment before CreateHelperObjects
```

NOTE: Please read the full details in my blog as it describes some special steps to take to ensure you don't encounter issues with Window's Virtual Store. The blog is specifically written about the _ReportListener issue, but the same issue applies to _GDIPlus. http://cathypountney.blogspot.com/2009/04/set-talk-appears-to-be-on-when-running.html

## How do I use fxReports?

Here's where we get into the meat of this session. First, I'll tell you how to download the files. Then I'll explain how to use the utilities.

### Download the core framework files

Go to the fxReports page in VFPX, as shown below in **Figure 1**. Either download the CoreFramework.zip file or sync to all the individual files in the CoreFramework subfolder.

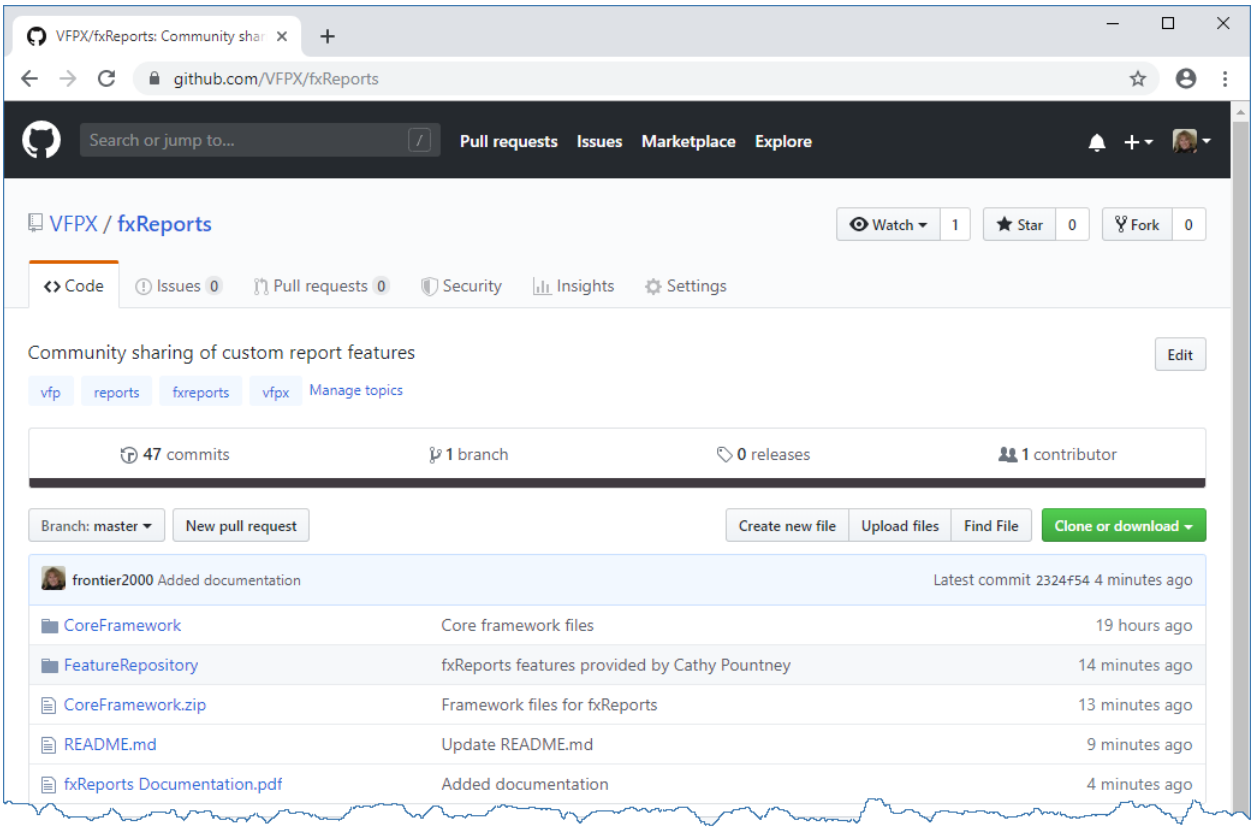



**Figure 1**. Download files from fxReports on VFPx.

Store these files in a place you can access from Visual FoxPro. For example, if you have a folder you use for all your development utilities, this would be the best spot. If you downloaded the zip file, be sure to unzip the file.

## Running the fxReports utility

Now that you have the core framework downloaded, let's get starting by running the main utility. From the Visual FoxPro Command Window, enter the following command (assuming the fxReports files are in your path):

```
DO FORM fxReports
```

You now have the main fxReports form running (see **Figure 2**.) This form has a pageframe with four pages; Getting Started, Using Custom Features, Creating Custom Features, and Sharing Custom Features.

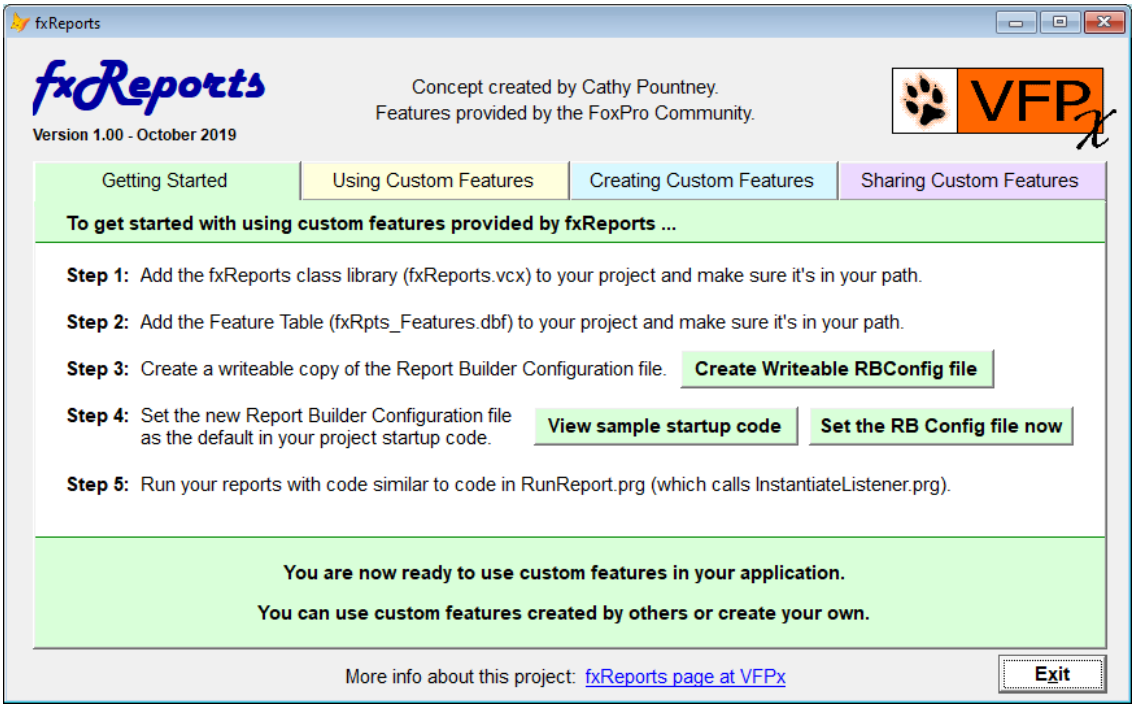



**Figure 2.** Run the main fxReports form to guide you through each step.

Each of the pages walks you through a particular set of tasks related to custom report features.

## Getting started

At this point you should have all the core framework files on your machine. To start taking advantage of using custom report features you need to get the core framework code merged into your application. You also need to setup up a Report Builder Configuration file. The first page of the pageframe (see **Figure 2**) walks you through the steps necessary to do just that.

### Step 1: Add fxReports.vcx to your application

The class library called fxReports.vcx contains two classes that constitute the "framework" of custom report features. This class library needs to be available to your application when it's running. You can either copy it into your application's directory structure, or change your startup program to add the class library's directory into the PATH statement. Either way, once that's done you should add fxReports.vcx to your application's project file.

The first class in this class library, fxRpts_fxAbstract, is subclassed from the fxAbstract class of the _ReportListener class library which resides in the FFC directory. If the relative path between your FFC directory and your fxReports.vcx class library don't match mine, you may have to fix the class parentage in the VCX. You can use your favorite hacking tool, the class browser, or the PEM Editor to do so. I'm assuming that as a FoxPro developer, you know how to do this. If not, you can certainly Google it or post the question on one of the many FoxPro developer forums.

### Step 2: Add fxRpts_Features.dbf to your application

The next step in preparing your application is to add the data-driven feature table (fxRpts_Features.dbf) to your application. When first starting out, this data file is empty. As you add custom report features to your application, this data file will contain one record for each feature. It's used by the fxReports framework as a way to be self-aware of any and all custom report features that may be used on your reports. It's also the mechanism for which the framework knows how to tie a particular custom report feature used in a report to the appropriate class necessary to implement the feature. In other words, the custom report features are data-driven and this is the data that does the driving.

The version of fxRpts_Features.dbf included in the fxReports download contains an empty version. Just as with the fxReports.vcx class library, you can copy this file into the directory structure of your application or change your startup program to ensure this data file is accessible within the path. Once you've done this, add the data file to your application's project file.

### Step 3: Create a writeable copy of the Report Builder Configuration file

Most FoxPro developers have no idea what a "Report Builder Configuration file" is. If you do, and you already have one, you can skip this step and move on to the next step. For the rest of you, it's okay that you don't know what it is. It's not necessary that you know what it is at this time. What is necessary is that you have to create a writeable copy of this file.

The Getting Started page on the fxReports main form contains a button labeled, "Create Writeable RBConfig file". Simply press this button and the work is done for you. First, you'll be prompted for the directory you want to store this file in. Next, you'll be prompted for the name of this file. The default is "RBConfig_Custom.dbf". Feel free to change this name to reflect your company or your application. For example, if I had an application for tracking widgets, I might choose "RBConfig_Widgets.dbf" as the filename. After entering the filename, the utility creates the new file for you and notifies you of its completion.

### Step 4: Use the Report Builder Configuration file

Once you've created a writeable copy of the Report Builder Configuration file, you need to tell FoxPro to use this file instead of its native internal file. You can do this in either the config.fpw file or with a line of code in your startup program. The Getting Started page on the main fxReports form contains a button labeled, "View sample startup code". Click this button and a message box appears showing you the necessary code for both techniques. Once you close the message box, the sample code is saved in your clipboard so you can easily paste it into your config.fpw file or your startup program.

If you don't already have a custom Report Builder Configuration file active, select the "Set the RB Config file now" button. You'll be prompted to select a file which is then set as the Report Builder Configuration file for this Visual FoxPro session.

### Step 5: Run your reports using special code

The last step in preparing your application to use custom report features is to run your reports through a special program. In other words, don't issue REPORT FORM xyz. Instead, run a program, passing a few parameters such as the report name, and the program sets the stage for using custom report features.

The core framework code you previously downloaded contains two programs, RunReports.prg and InstantiateListener.prg. The two programs work together. Reports are run by calling RunReports and inside of that program, it calls InstantiateListener. Here's an example of how to run reports:

```
DO RunReport WITH .f., 'My_Special_Report', '', 1, .f.
```

You can copy these two programs into your application and add them to your project file. However, don't feel like you have to use them exactly as written. In fact, if you already have a program you run reports through, feel free to incorporate the main pieces of these programs into your existing program. The same is true for Report Listeners. If you already have your own Report Listener in your application, modify InstantiateListener to invoke yours. You do, however, have to ensure your Report Listeners are based from the fxListener class (or one of its derivatives) in the _ReportListener class library in the FFC directory.

◆ ◆ ◆

Now that you have completed the five steps listed above, your application is ready to use custom report features. Running your application at this point shouldn't yield any different results than what you're used to seeing. The difference is that the foundation is now in place to start using custom report features created by you or anyone else. In the next section, I describe how to use custom report features created by someone else.

## Using custom features

The second page on the fxReports main form (see **Figure 3**) is designed to walk you through the process of merging custom report features created by someone else into your existing application.

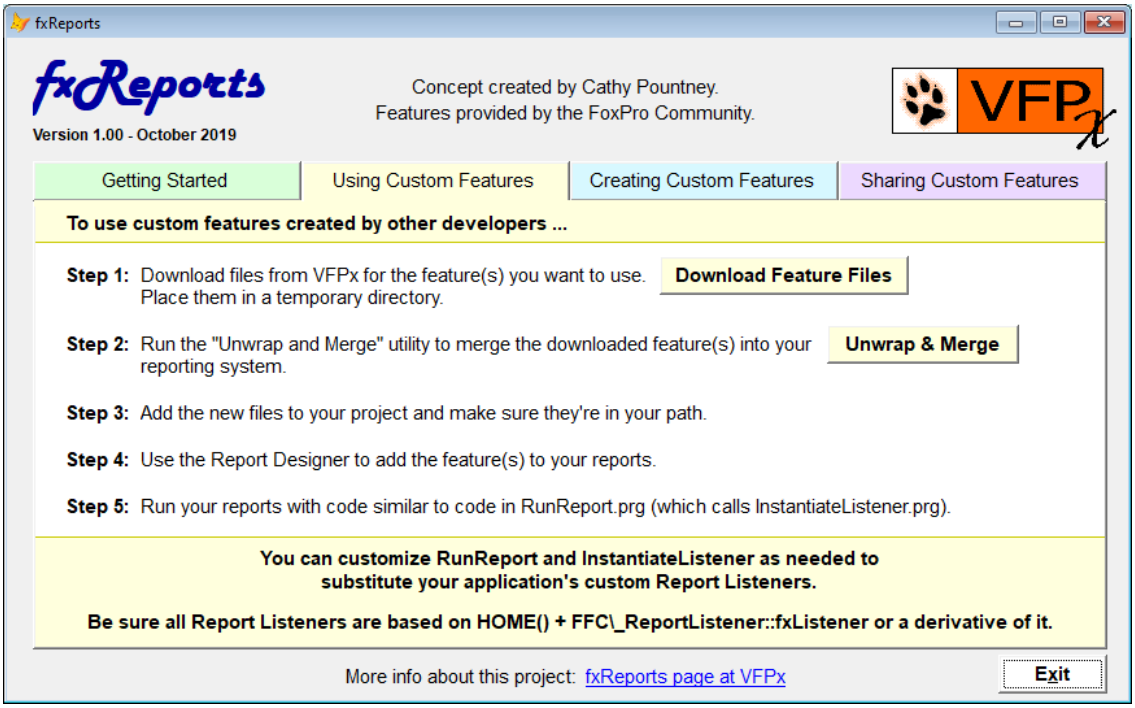



**Figure 3.** Use the "Using Custom Features" page to guide you through the process of using custom report features designed by others.

### Step 1: Download a custom report feature

Custom report features provided by the Visual FoxPro community are stored in the FeatureRepository subfolder of the fxReports project on VFPx. Click the "Download Feature Files" button to navigate your default browser to that page.

Within the FeatureRepository subfolder, you'll see one subfolder and one zip file for each developer who has contributed features to fxReports. Selectively download the zip files (or sync to all the individual files) for the feature sets you which to incorporate into your application.

> *NOTE: Review the TOC.md file for a list of each feature provided by the contributors.*

I'd like to point out that you aren't limited to custom report features available from the fxReports project on VFPx. This same reporting framework can be used to create custom report features that are shared amongst developers of a team or organization. Or maybe custom report features are shared between a developer and his or her clients. All you need to do is obtain the packaged set of feature files from the developer and you'll be able to continue with the process of using their custom report features in your application.

### Step 2: Run the "Unwrap & Merge" utility

The files downloaded from the FeatureRepository subfolder of the fxReports project on VFPx cannot simply be added to your application. There's a little bit of work that has to be done to merge records from temporary data files contained in the download into your fxRpts_Features table and your Report Builder Configuration file.

Don't fret. I've done the work for you. Click the "Unwrap & Merge" button to launch the Unwrap & Merge utility (see **Figure 4**) which takes care of processing the files and merging the data as needed.

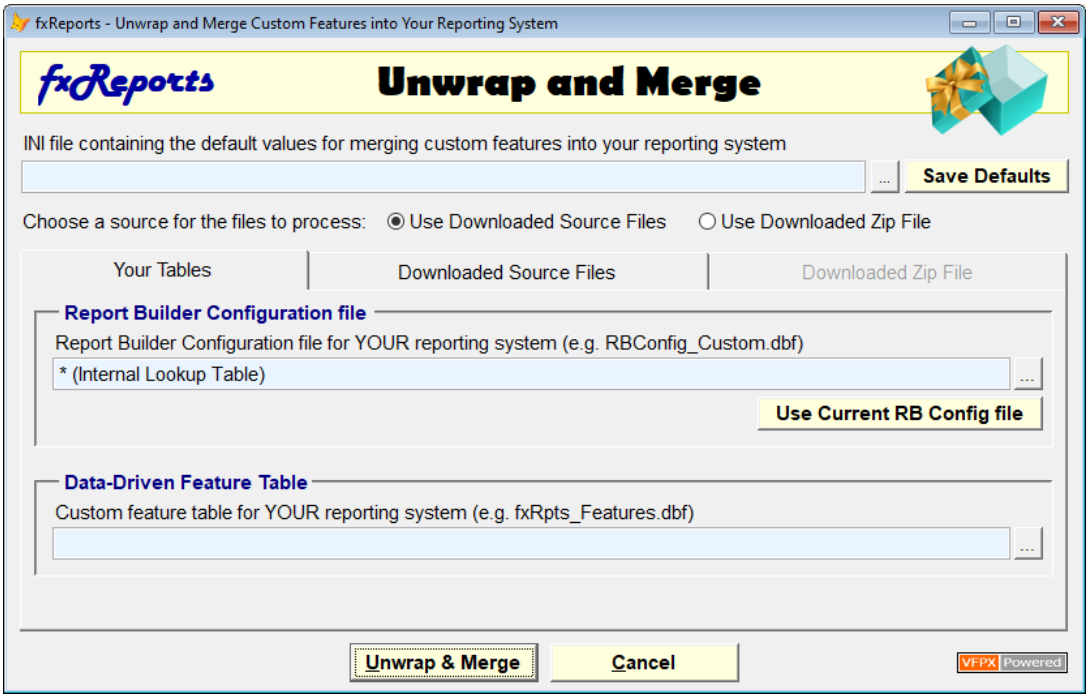



**Figure 4.** Use the Unwrap & Merge utility to process a downloaded custom report feature and merge it into your application.

The first object at the top of the form is an ellipsis button [...] for choosing an INI file. Ignore this the first time you run this utility. I'll discuss this more later.

The second object at the top of the form is an option group that lets you choose between individually downloaded source files or a zip file. If you downloaded individual files from the fxReports\FeatureRepository\[DeveloperName] page on VFPx, select the first option. If you downloaded just the .ZIP file, select the second option. Depending on which of those options you chose, the applicable page is enabled on the pageframe. I'll describe each of these options in a moment.

#### *Your Tables*

The next object on the form is a pageframe with three pages. The first page is labeled "Your Tables" and the first object is the location of your Report Builder Configuration table. Previously, in Step 3 of the Getting Started page, you created a writeable copy of the Report Builder Configuration table. Use the ellipsis button [...] to locate that table. If you're

currently running with that Report Builder Configuration table, you can use the "Use Current RB Config file" button to automatically fill in the name and location of the current file.

> *NOTE: If you select the "Current RB Config file" button and "* (Internal Lookup Table)" is displayed in the textbox, it means you are running the native Report Builder Configuration table within Visual FoxPro, not a writeable copy of that table.*

The next object on the "Your Tables" page is the location of your data-driven feature table. This was created previously in Step 2 of the Getting Started page. Use the ellipsis button [...] to locate that table.

### *Downloaded Source Files*

The second tab on the pageframe (see **Figure 5**) is where you indicate the location you stored the downloaded source files from VFPx.

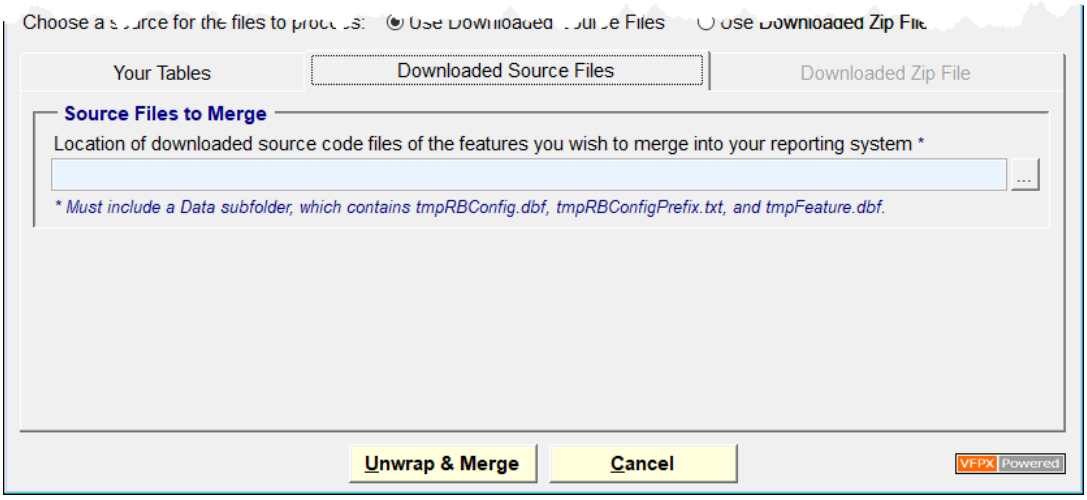



**Figure 5.** Use the Downloaded Source Files page to identify the location of the files you are processing.

Use the ellipsis button [...] to locate the directory containing the files you wish to process. This is the set of files you downloaded from the applicable FeatureRepository subfolder of the fxReports project on VFPx. Or, as previously mentioned, this could be a set of files given to you by a fellow co-worker, mentor, or other business associate. It doesn't necessarily have to come from VFPx.

### *Downloaded Zip File*

The third tab on the pageframe (see **Figure 6**) is where you identify the zip file you wish to process, as well as a temporary location for unzipping that file.

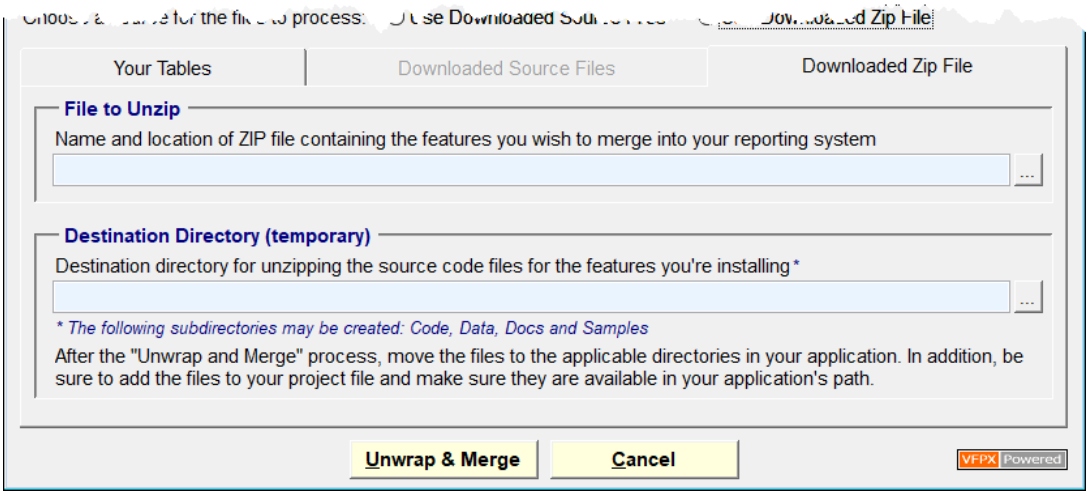



**Figure 6.** Use the Downloaded Zip File page to identify the zip file you are processing, as well as a temporary location for unzipping that file.

Use the ellipsis button [...] to locate the zip file you downloaded from the FeatureRepository subfolder of the fxReports project on VFPx. Or, as previously mentioned, this could be the zip file given to you by a fellow co-worker, mentor, or other business associate. It doesn't necessarily have to come from VFPx.

The last piece of information on this page is the temporary location where you want the zip file unzipped. As indicated on the screen, four subdirectories may be created within the directory you identify; Code, Data, Docs, and Samples.

### *INI file*

You're now done entering the required information to process the custom report feature. However, there's one additional step you can take. At the top of the Unwrap & Merge utility, there's an ellipsis button [...] that allows you to indicate an INI file. If you give it a filename and click the button labeled "Save Defaults", the information you've entered into this utility is saved. The next time you run this utility, you can select the INI file first thing and all the information stored in the INI file will be used to populate the fields. This saves you from having to enter the information each time.

◆ ◆ ◆

Now that all the information has been entered into the utility, click the "Unwrap & Merge" button at the bottom of the form. This processes the indicated file(s). If you chose a zip file, the utility unzips it into the temporary directory. The utility finishes by merging the records from the temporary Report Builder Configuration table and data-driven feature table into your versions of those files.

### Step 3: Add files to your project

The Unwrap & Merge utility from the previous step may have unzipped some files to a temporary folder or may have included a set of individual source files. Locate those files and look in the Code subdirectory to see the source code for this custom report feature. This source code needs to be copied into an appropriate directory with your application's

source code. If you choose to copy the files into their own directory, you'll need to make sure that directory is accessible in the path of your application when it's running.

It's highly unlikely that every developer who uploads custom report features to the fxReports project on VFPx uses the same directory structure as yourself. As with the fxReports.vcx class library previously described in Step 1 of Getting Started, the relative path for classes in this custom report feature may not be consistent with your own directory structure. The classes should be based on the fxReports.vcx class library, but the other developer may have it located in a directory different than what you do. Most likely, you will need to use your favorite hacking tool, the class browser, or the PEM Editor to fix the relative path of the class parentage in some of the classes.

In addition to the custom report feature's source code, the authoring developer may have created samples and documentation to help you learn how to use their feature. It's up to you on where you permanently store these files.

The files in the data directory contain the records that need to be merged into your Report Builder Configuration file and your data-driven feature table. However, you don't need to do anything with them. The Unwrap & Merge utility already did it for you. I chose to leave the temporary data files there instead of deleting them in case your curiosity gets a hold of you and you want to see the records yourself. Once the Unwrap & Merge utility has been run, you can delete the files from any temporary data directory.

#### Step 4: Add features to your reports

At this point you have the new custom features in place to work with your application. You should be able to create or modify a report and have access to the new custom report features within the Report Designer user-interface. Hopefully the developing author has provided proper documentation to explain how to use the new custom report feature.

#### Step 5: Run your reports using special code

The last step is just a reminder that you have run reports through the special program that sets the stage for the custom report features to work. This was previously described in Step 5 of Getting Started.

◆ ◆ ◆

Now that you've completed steps 1 through 5, you should be all set. You can run your application, run your new report, and the new custom report feature should appear.

When you feel comfortable with using custom report features created by others, it's time to start thinking about creating your own custom report features. In the next section, I describe how to go about doing this.

## Creating custom features

Once you've prepared your system to use the fxReports framework and you've used a few features developed by others, you're ready to create some of your own custom report

features. I'm sure you've wished you could make a report do something that it hasn't done in the past. Well, with a little ingenuity and a little more know-how, you can do it.

The third page on the fxReports main form is called "Creating Custom Features" (see **Figure 7**) and it walks you through the steps for creating your own custom report features. Actually, it doesn't hold your hand and give you detailed instructions for this. That is way beyond the scope of this session. What it does do is give you a very general guideline.

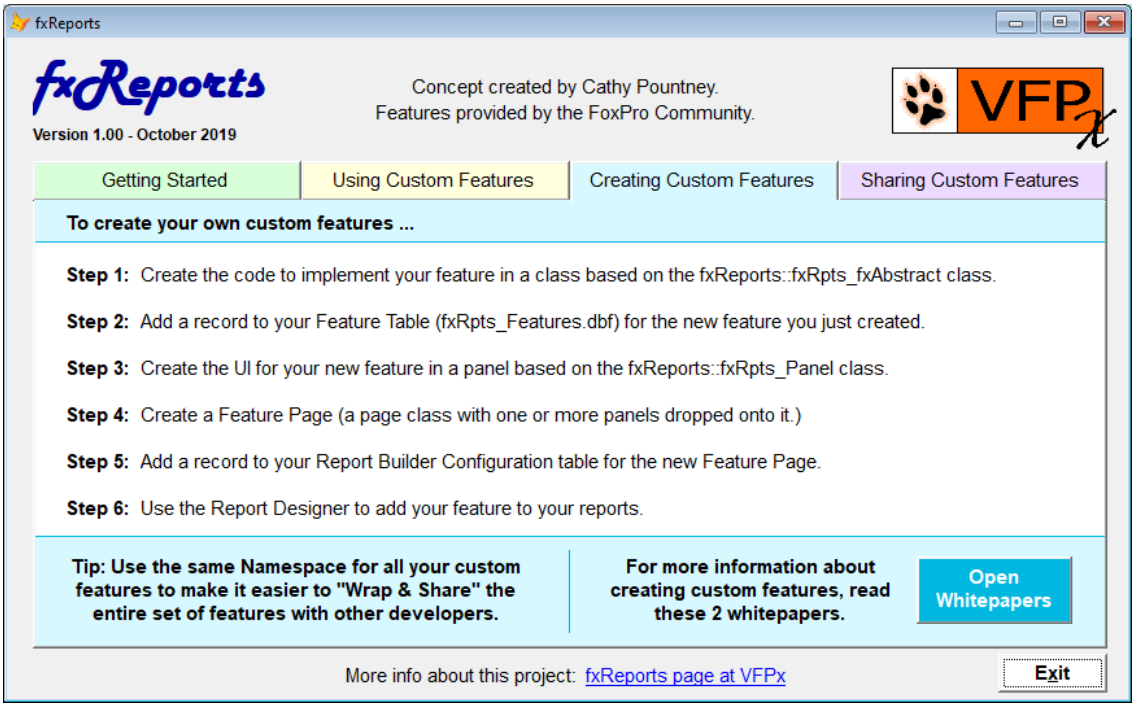



**Figure 7.** Use the Creating Custom Features page to walk you through the steps of creating your own custom report feature.

### Open whitepapers

At a previous Southwest Fox conference, I gave a two-part series of sessions that covered the Visual FoxPro 9 SP2 report changes in great detail. Much of what I covered is the basis behind the fxReports concept. Rather than repeat myself, I'm including both whitepapers from those sessions so you can read through them and learn about creating custom report features. For convenience, I've also included those whitepapers in the core framework section of the fxReports download from VFPx. For even more convenience, I've added a button labeled "Open Whitepapers" to the fxReports utility which opens the whitepapers using your default PDF reader. I strongly encourage you to read these.

### Step 1: Create the code to implement your feature

The code that implements your custom report feature on a report is created in a class which has been subclassed from the fxRpts_fxAbstract class in the fxReports class library. Refer to the whitepapers for more information on how to do this.

**Step 2: Add a record to the data-driven your feature table**

Once you create a new custom report feature, you have to let the fxReports framework know about the feature. That is done by adding a record to your data-driven feature table (fxRpts_Features.dbf). Refer to the whitepapers for more information on how to do this.

**Step 3: Create the UI panel for your feature**

A panel is nothing more than a container class that holds the objects necessary to gather information from the developer when editing a report in the Report Designer. For example, the Watermark feature I created has to ask the developer for the name of the graphic file as well as sizing and alignment information. This panel should be created as a subclass of the fxRpts_Panel class in the fxReports class library. Refer to the whitepapers for more information on how to do this.

**Step 4: Create the UI page for your feature**

Once you've created a panel, you have to create a page and drop the panel on the page. More than one panel may be dropped on a page. Each page you create is added to the pages already shown in Properties dialog in the Report Designer. You need to add code to a few methods to save and restore the data for the report. At the risk of repeating myself, refer to the whitepapers for more information on how to do this.

**Step 5: Add a record to your Report Builder Configuration file**

The last step in creating a custom report feature is to tell the Report Designer about the new UI page you created. This is done by adding a record to your Report Builder Configuration file. You guessed it … Refer to the whitepapers for more information on how to do this.

◆ ◆ ◆

Creating custom report features looks a little intimidating at first but once you learn how to do this, it's not all that bad. Read through the whitepapers to fully grasp the concept. I'm sure once you do, you'll come up with all kinds of custom report features you want to create. And if you're thinking of them, you can bet other developers are thinking of them too. If you figure out how to implement your desired feature, please share it with other FoxPro developers, as described in the next section.

## Sharing custom features

As I mentioned at the beginning, the FoxPro Community is all about giving and sharing. The entire fxReports concept won't work if developers don't participate and don't share their custom report features. I urge you to get on board and share custom report features you've created. The last page on the fxReports main form is called "Sharing Custom Features" (see **Figure 8**) and it can be used to guide you through the sharing process.

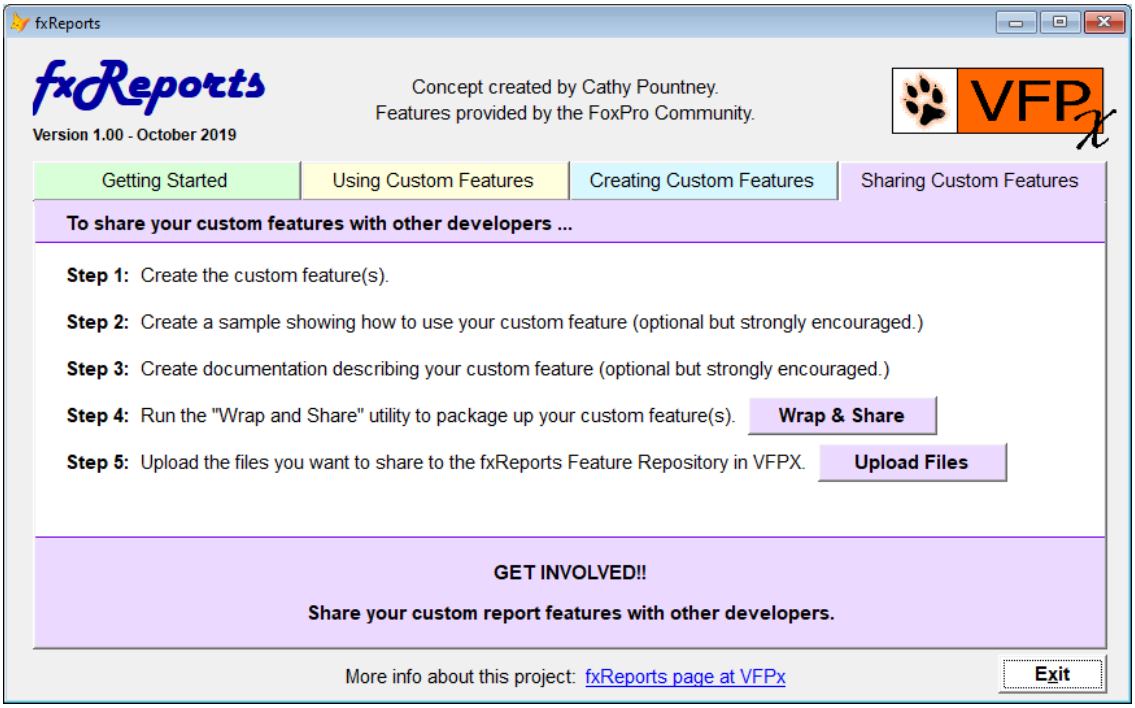



**Figure 8.** Use the Sharing Custom Features page on the fxReports main utility to share your custom report features with other developers.

### Step 1: Create the custom features

This is kind of a no-brainer, but you have to create the custom report feature before you can share it. See the previous section titled "Creating Custom Features" for more information on how to do this.

### Step 2: Create samples

You created your custom report feature and you know how it runs. But think about the other developers who may want to use your feature. If you would create a simple example of how this feature works on a report, it would go a long way with helping another developer understand your feature. This step isn't mandatory, but I highly recommend you take the time to do this. Remember; Do unto others as you want others to do unto you.

### Step 3: Create documentation

Along the same lines of creating samples, it would be most helpful if you also created documentation showing how to use your custom report feature. Again, this is optional but highly encouraged.

### Step 4: Run the Wrap & Share utility

Previously, I explained how to use the Unwrap & Merge utility which processes files so you can utilize custom report features created by others. The Wrap & Share utility (see **Figure 9**) is the converse. It's designed to package up everything needed to implement your custom report feature so it can be shared with others. This is the same utility that was used

to prepare files you download from a FeatureRepository subfolder in the fxReports project of VFPx.

The first object at the top of the form is an ellipsis button [...] for choosing an INI file. This the first time you run this utility. I'll discuss this in more detail later. The next object on the form is a pageframe with six pages; RB Config, Your Features, Your Code, Your Samples, Your Documentation, and Output.

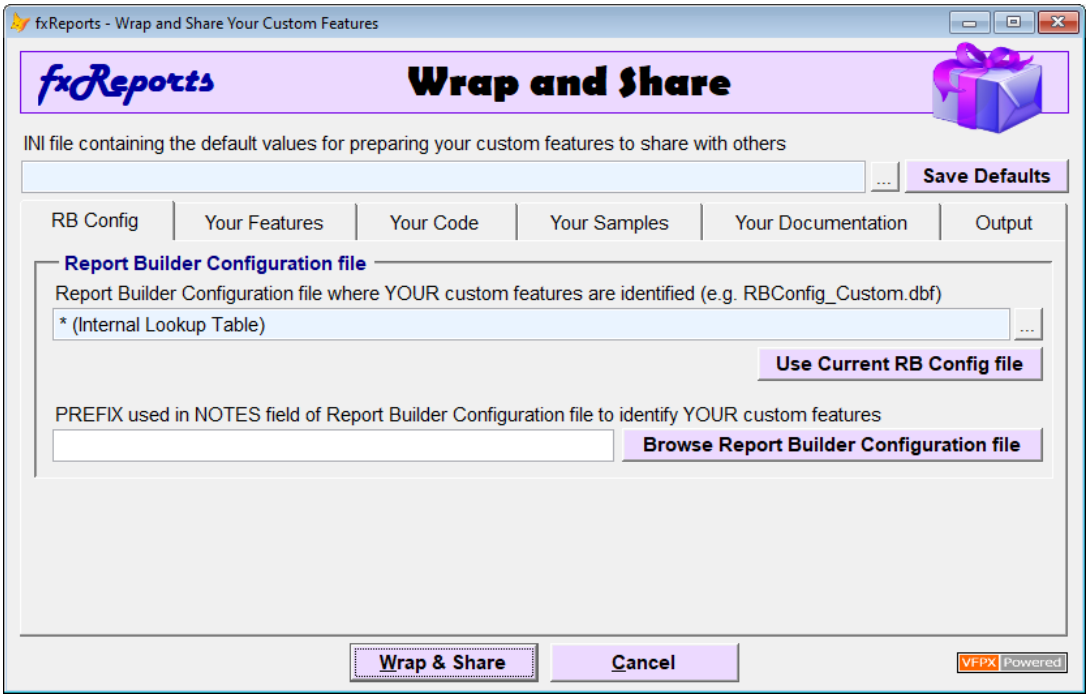


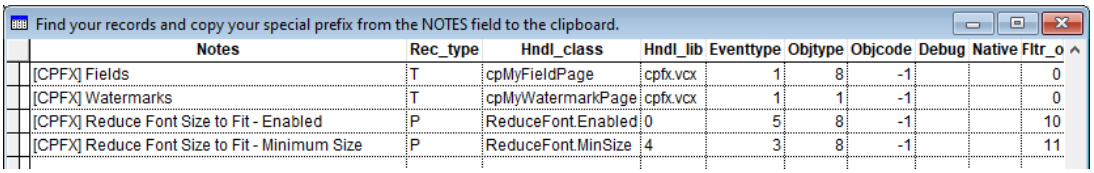

**Figure 9.** Use the Wrap & Share utility to package up your custom report features to share with other developers.

### *RB Config*

The "RB Config" page is where you tell it about your Report Builder Configuration file.

Use the ellipsis button [...] to locate your custom Report Builder Configuration file. If you're currently running with this file, you can use the "Use Current RB Config file" button to automatically fill in the name and location of the current file.

The next piece of information to enter is the unique prefix you used on all the records you added to the Report Builder Configuration file. This is important because the utility needs to create a temporary data file with just the records related to your custom features. Those records will later be merged into the other developers Report Builder Configuration file by the Unwrap & Merge utility they run on their system. If you know the prefix you used, you can enter it in the appropriate textbox. If you're like me and have troubling remembering what you had for dinner last night, you can click the "Browse Report Builder Configuration file" button to invoke a browse window (see **Figure 10**.)




| Notes | Rec_type | Hndl_class | Hndl_lib | Eventtype | Objtype | Objcode | Debug | Native | Fltr_o |
|---|---|---|---|---|---|---|---|---|---|
| [CPFX] Fields | T | cpMyFieldPage | cpfx.vcx | 1 | 8 | -1 | | | 0 |
| [CPFX] Watermarks | T | cpMyWatermarkPage | cpfx.vcx | 1 | 1 | -1 | | | 0 |
| [CPFX] Reduce Font Size to Fit - Enabled | P | ReduceFont.Enabled | 0 | 5 | 8 | -1 | | | 10 |
| [CPFX] Reduce Font Size to Fit - Minimum Size | P | ReduceFont.MinSize | 4 | 3 | 8 | -1 | | | 11 |

**Figure 10.** Browse the Report Builder Configuration file to assist with remembering the prefix used on your custom report features.

Peruse the Notes field and find the records related to your features. When you find a record, copy the prefix into your clipboard (it should begin and end with square brackets). Next, close the browse window and return to the Wrap & Share utility. The text you just copied in your clipboard is automatically inserted in the Prefix field.

### *Your Features*

The "Your Features" page (see **Figure 11**) is where you tell it about your data-driven features table.

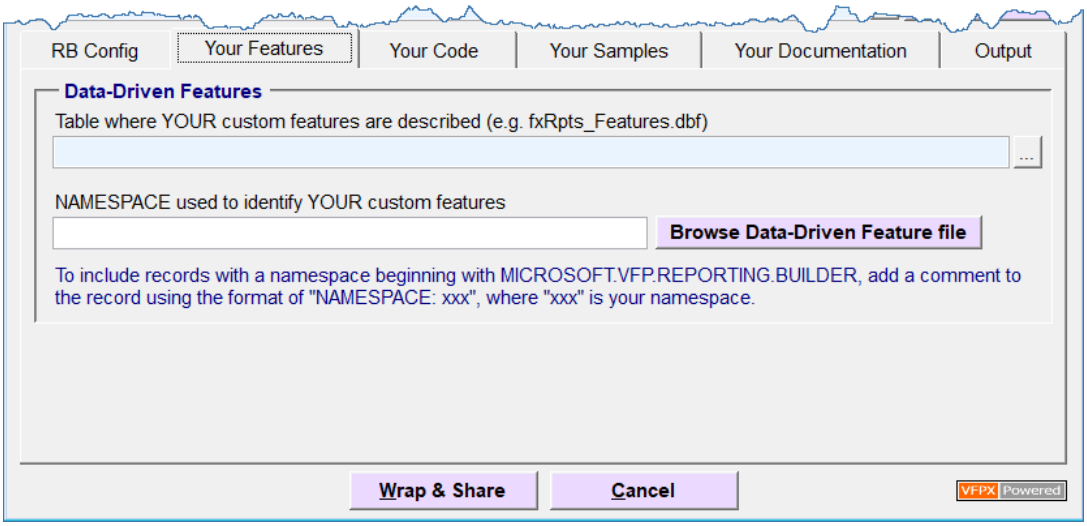
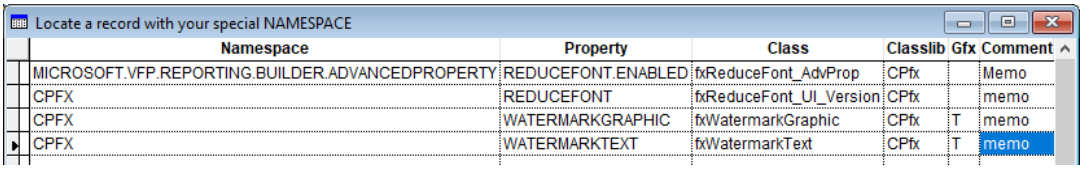



**Figure 11**. Use the Your Features page to tell the utility about your data-driven features table.

Use the ellipsis button […] to locate your data-driven feature table. After choosing a table, you need to enter the Namespace you used in this table to identify your custom report features. Similar to the Report Builder Configuration file Prefix, you can enter the Namespace manually or use the "Browse Data-Driven Feature file" button to invoke a browse window (see **Figure 12**).




| Namespace | Property | Class | Classlib | Gfx | Comment |
|---|---|---|---|---|---|
| MICROSOFT.VFP.REPORTING.BUILDER.ADVANCEDPROPERTY | REDUCEFONT.ENABLED | fxReduceFont_AdvProp | CPfx | | Memo |
| CPFX | REDUCEFONT | fxReduceFont_UI_Version | CPfx | | memo |
| CPFX | WATERMARKGRAPHIC | fxWatermarkGraphic | CPfx | T | memo |
| CPFX | WATERMARKTEXT | fxWatermarkText | CPfx | T | memo |

**Figure 12.** Browse the Data-Driven Feature file to assist with remembering the namespace used on your custom report features.

Peruse the table and find one of your records. When you close the browse window, the namespace of the record you were on is inserted into the Namespace textbox on the Wrap & Share utility.

> *NOTE: The Visual FoxPro Report Builder requires the use of the specific namespace of "MICROSOFT.VFP.REPORTING.BUILDER.ADVANCEDPROPERTIES" for any properties added to the Advanced tab. To identify those records for this utility, add a comment of "NAMESPACE: xxx", where "xxx" is the namespace you are including. The utility will include all records with the given namespace and all records with the comment described above.*

### *Your code*

The third page on the pageframe is where you identify all the source code needed to implement your custom report features (see **Figure 13**.) Use the Add button to add a file to the list. Be sure to include any class library, program, image, and other files needed by your custom report feature.

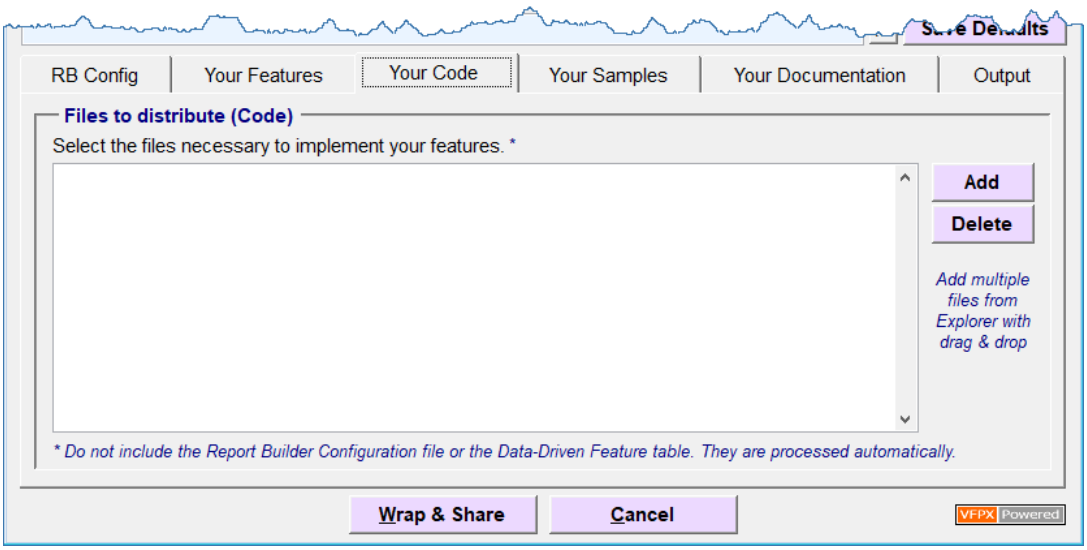



**Figure 13.** Use the "Your Code" page to identify all the source code needed to implement your custom report features.

To add files, you may drag and drop one or more files from Explorer into the listbox. You may also select the Add button to display a file dialog for choosing an individual file. To delete a file from the list, highlight the file and then select the Delete button.

### *Your samples*

The fourth page on the pageframe is where you identify all the samples you've prepared to showcase your custom report features (see **Figure 14.**) Use the Add button to add a file to the list. Be sure to include any class library, program, image, report, and other files involved in your sample.

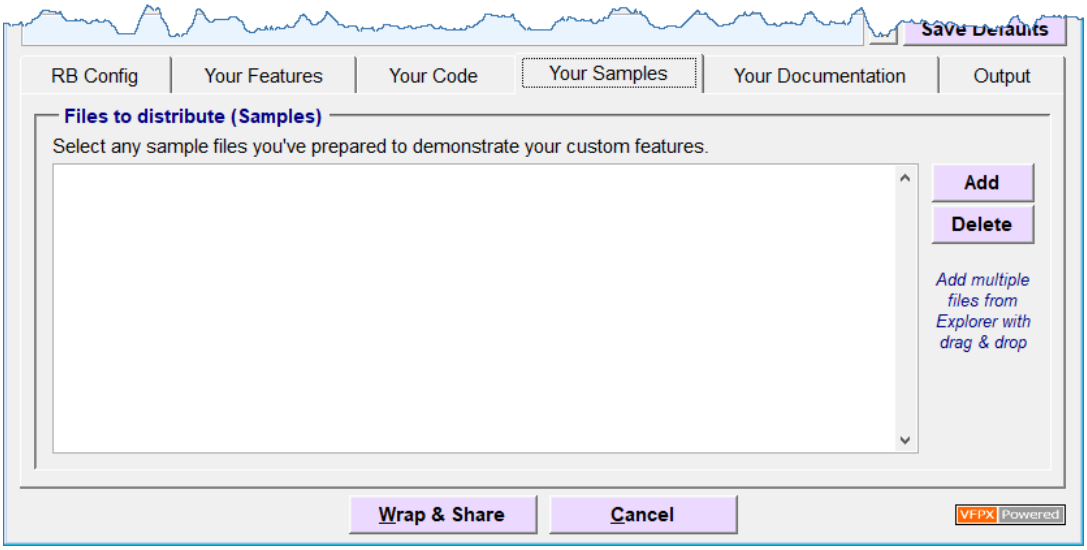



**Figure 14.** Use the "Your Samples" page to identify any samples prepared to showcase your custom report features.

To add files, you may drag and drop one or more files from Explorer into the listbox. You may also select the Add button to display a file dialog for choosing an individual file. To delete a file from the list, highlight the file and then select the Delete button.

Samples aren't required, but I strongly encourage you to create them to assist other developers in understanding how to use your custom report features.

### *Your documentation*

The fifth page on the pageframe is where you identify any documentation you've prepared for your custom report features (see **Figure 15**.) Use the Add button to add a file to the list.

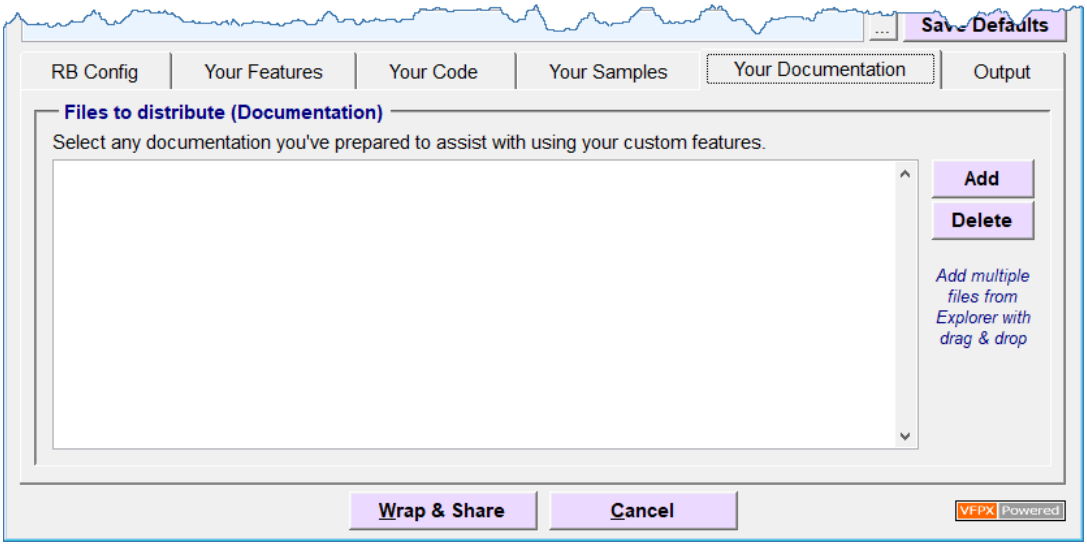



**Figure 15.** Use the "Your Documentation" page to identify documentation you've created for your custom report features.

To add files, you may drag and drop one or more files from Explorer into the listbox. You may also select the Add button to display a file dialog for choosing an individual file. To delete a file from the list, highlight the file and then select the Delete button.

Documentation isn't required, but I strongly encourage you to create documentation to help other developers in understanding how to use your custom report features. Just think about how much you'd appreciate documentation from another developer about his or her custom report feature. All I ask is that you return the favor and provide the same.

### *Output*

The last page on the pageframe (see **Figure 16**) is where you tell the Wrap & Share utility to store the files it's going to create.

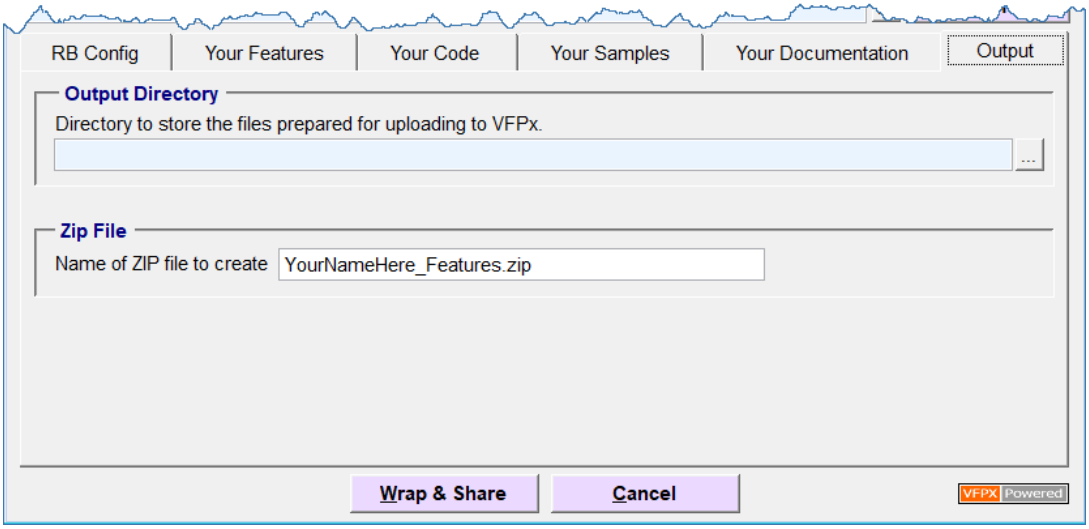



**Figure 16.** Use the "Zip File" page to indicate where the newly created zip file should be stored.

Use the ellipsis button [...] to locate a directory where all the files prepared by this utility will be stored.

Enter the name of a zip file, which will be created by this utility. The default file name is YourNameHere_Features.zip. Please replace the "YourNameHere" portion with, um, your name!

### *The INI file*

You're now done entering the required information to process your custom report features and make them available to others. However, there's one additional step you can take. At the top of the Wrap & Share utility, there's an ellipsis button [...] that allows you to indicate an INI file. If you give it a filename and click the button labeled "Save Defaults", the information you've entered into this utility is saved. The next time you run this utility, you can select the INI file first thing and all the information stored in the INI file will be used to populate the fields. This saves you from having to enter information each time.

◆ ◆ ◆

Now that all the information has been entered into the utility, click the "Wrap & Share" button at the bottom of the form. First, temporary data files are created which contain the appropriate records from the indicated Report Builder Configuration file and the data-driven feature table. It also copies the indicated files from the Code, Samples, and Documentation pages into the folder indicated on the Output page. Lastly, a .ZIP file is created.

#### Step 5: Upload the files

Once you've finished the Wrap & Share process, the final step in sharing your custom report features with other developers is to upload the files to the fxReports project in VFPx. Click the "Upload Files" button to navigate to the FeatureRepository subfolder using your default internet browser.

Update the TOC.md file in the FeatureRepository subfolder, listing each of the features provided by you.

Upload the zip file created by the Wrap & Share utility to the FeatureRepository subfolder.

Upload the other files created by the Wrap & Share utility to a subfolder of the FeatureRepository. Name the subfolder after yourself.

As a note, you don't have to make your features available on the fxReports project in VFPx. You could just send the files to a co-worker or customer. However, I strongly encourage you to share with the entire FoxPro Community.

## Summary

I can't stress enough how much the FoxPro Community gives back to each other. We truly are a "community". I'm very excited to launch the fxReports concept and start the ball rolling so we can all take advantage of the wealth of knowledge and skill available to us. I encourage everyone to get involved with this project. Start by using custom report features in your application and provide feedback to the developers who have created those features. Once you get more comfortable, take the time to create some custom report features of your own and reciprocate by sharing your features with the entire FoxPro Community.

## Biography

*Cathy Pountney has been developing software since 1982 and has been heavily involved in the FoxPro Community for years. She's written numerous articles for various FoxPro magazines. She authored The Visual FoxPro Report Writer: Pushing it to the Limit and Beyond and co-authored Visual FoxPro Best Practices for the Next Ten Years and Making Sense of Sedna and SP2. Over the years, she has spoken at numerous FoxPro conferences in the U.S., Canada, and Germany. Cathy is proud to have earned the Visual FoxPro Community Lifetime Achievement Award in 2012 and equally proud to have received the Microsoft Visual FoxPro MVP Award eight years in a row (until Microsoft retired the program). Cathy also had the honor of working as a subcontractor onsite in Redmond with the Microsoft Fox Team in 2001. Cathy's*

*career has consisted of writing software for a variety of industries as an employee, subcontractor, and running her own consulting company. She currently works for White Light Computing.*