## Column: From the Editor

## Figures:
## File for Subscriber Downloads:

VFP Data on Your Handheld: An Introduction to SatForms

Whil Hentzen

Last month I introduced you to Satellite Forms, a software package that includes both a programming environment that allows you to create applications that run on a Palm handheld and an ActiveX control that allows you to synchronize data between that Palm app and a desktop application. This month I'm going to delve into this tool and walk you through the creation of a simple Palm application that synchronizes with a matching desktop VFP app.

Wow. From the looks of the email I've received over the last couple of months, we should rename this newsletter "FoxAndPalmTalk" or something closely akin. What's really interesting is running into all sorts of Fox developers up on the Satellite Forms message board. Evidently this is a hot topic for many of you. So let's jump in right away.

Before I start slinging code ("Oh no, he's not going to talk about *design* first, is he?"), it's a good idea to reflect on what you want in this "VFP Data on your Palm" arena. Each implementation is done differently and makes assumptions. Coming into this new, it's good to state what those assumptions are. First of all, you probably have an application running on your desktop or server. This application is the primary repository for the data that you want to carry around (and manipulate) on your Palm.

Second, at some point, you're going to want to download some (but not all) of this data to your handheld. And you're going to want to do this by popping your handheld into its cradle and pushing a button. You don't want to have to execute menu options or click command buttons on software that's running on your desktop, or tap software icons on your handheld.

Next, you're going to take your handheld out of its cradle and work on it for a while. It seems that construction sites, oilfields, and subway trains seem to be the most popular sites, if you just look at the ads in the popular press, but, truthfully, you could work on it, yes, anywhere, except perhaps the shower.

The type of work you'd do "in the field" includes three types – changing existing records, adding new records, and perhaps deleting existing records – that would require synchronization with your desktop app at some later point.

Finally, you're going to traipse back into the office, pop your handheld into its cradle, push the sync button, and have your handheld's changes moved back to your desktop app.

Don't forget that the data on the desktop app might have been changed in the interim – your administrative assistant (why would you have one in this day and age? I dunno, but it could happen) might have updated your calendar, or the marketing weenies might have updated the master copy of the product database that you're carrying around in your pocket.

Satellite Forms Components

SatForms (that's what the Puma Technologies marketing, tech support, and all the cool kids at school call it) comes with four components, and I covered those briefly in last month's column. The key things you have to remember from last month is that there are two essential pieces – the SatForms IDE, with which you build the app that runs on your handheld, and the SatForms ActiveX control, which you place on your desktop app, and that automatically detects HotSyncs and can run code you write to customize the syncing of data.

Overview of development process

Suppose a copy of SatForms just showed up on your desk one day. What would you do to get your application's data onto your handheld?

First, you'd install the SatForms package on your PC. During the installation process, you'll need to have your cradle connected to your PC, your handheld in your cradle and your synchronization software running. Then all you need to do is follow the instructions. During the installation, the SatForms SDK gets installed on your Palm, and the SatForms ActiveX control gets installed and registered on your PC. Takes less than five minutes.

Next, you'll develop the application that you want to run on your Palm, using the SatForms IDE. This usually takes more than five minutes, but might not take that much longer. It took me perhaps twenty minutes to create a single table, a two page form whose controls mapped to fields in that table, and download it to my Palm V.

The third step is to, if this hasn't already been done, develop the application that's going to house your data on your PC. Many times this will already have done, but it will now need one more module – the one that contains the ActiveX control that waits for a HotSync to occur.

In other words, you need to have an application running on your PC at all times – or at least all the time that a HotSync could occur – so that when you press the Sync button on the cradle, your PC can detect this and handle the data transfer for your app.

This may seem strange at first, but when you think about it, you've already got one of these running – the Palm HotSync Manager that's setting down thar in your System Tray. It's kind of like having an NT service on your machine, ready, lying in wait for a HotSync to occur.

Thus, what you'll often do is either add a module to your existing app or create a separate, stand-alone app that has access to your desktop application's data. You'll place the SatForms ActiveX control on a form in this module/stand-alone application, and add code to selected methods in the ActiveX control to control how you want data synchronized. The amount of code you add depends on how sophisticated your synchronization requirements are. In either case, this module/stand-alone application will be running all the time.

The last two steps are to deploy your apps on both pieces of hardware. On your PC, you have multiple choices. You could go whole hog and create a separate installation package that installs your PC app as a separate program, or just run your VFP app from within the interactive development environment. You could even configure your application to run as an NT service. On the handheld, it's much easier. One of the menu choices in the SatForms IDE to download your app and tables to your handheld. Pretty painless.

Overview of day-to-day use

OK, now that you've got your PC application running and your SatForms application installed on your handheld, it's time to use it. I'm going to assume that you've got some data in your PC application to start with. If you don't, you can skip a couple of steps.

Put your handheld in its cradle, and hit the Sync button. As the sync process runs, your PC app will detect the HotSync, and run the code in the appropriate methods. The data on your PC will be moved down to your handheld as you defined in the code you wrote in the methods of the SatForms ActiveX control. Take your handheld out, and work with it as desired. At the end of the day (or whatever), put your handheld back in its cradle again. Make sure that the PC application is still running, and hit Sync on the cradle.

Details about the SatForms Data Structures

There's actually an intermediate step in this process that I've kept hidden from view to this point. You might be wondering how the data in the tables in a VFP database get moved down to the Palm – surely the SatForms app on you Palm doesn't use DBFs as the data storage mechanism, does it?

In a word, no. Satellite Forms stores data in a proprietary .PDB file format. On of the chores of the SatForms ActiveX control is to convert the data on the PC to the PDB file format. Of course, to do so, it would seem that the data on the PC would need to be in some sort of standard or recognizable format. Funny enough, Puma Technologies decided upon the dBASE 5 .DBF file format as their standard file format for the PC. Thus, the ActiveX control looks for data in a dBASE 5 .DBF, and converts that data to the PDB format used on the handheld.

This means that, unless you're using dBASE 5 DBFs in your Visual FoxPro app (hee hee), you'll need to move data from your application's data store to dBASE 5 DBFs. The code you write in the SatForms ActiveX control's methods are, in large part, responsible for this chore. Obviously, it's not a big task – not nearly as big as if you were using, say, Access or Paradox, each of which have their own particular "challenges" when it comes to working with DBFs.

So here's what happens underneath the hood. When the SatForms ActiveX control in your PC application detects a HotSync, it will call a method that creates a set of intermediate dBASE 5 DBFs, and then converts data from the application's data store to these DBFs. (Note that I said "create" – if they were already there, they get completely overwritten. More on this in a second.) You write the code in this method. Then the ActiveX control will call a second method that automatically converts those dBASE 5 DBFs to PDB format and transfers the data to the handheld.

On the return trip, when the handheld data is being moved back to the PC, the opposite happens. The ActiveX control detects a HotSync, and calls a method that grabs the PDB data from the handheld, and creates the appropriate dBASE 5 DBFs again. Then the ActiveX control calls a second method (that you put code in) that converts those dBASE 5 DBFs to the data format used by your application.

What this means, of course, is that you can keep your application's data in any format that your app can read – free tables, VFP database tables, or a remote data source that your application accesses through remote views or SPT. And, as I mentioned briefly in last month's article, you don't even need to do this on your own desktop – if you want to spend the money, you can get an "Enterprise version" of SatForms that syncs with a server like Oracle instead of a desktop.

OK, back to this business of the dBASE 5 DBFs getting overwritten. The fundamental problem with synchronizing data between two sources that data could be changed on both sides, and you have to figure out which data is newer, and resolve conflicts when the same data on both sides has changed. The SatForms ActiveX control doesn't do anything about trying to resolve these conflicts – it just assumes the data in the dBASE 5 DBFs is "the data, all the data, and nothing but the data." There is no conflict resolution performed between the DBF and the PDB on the handheld.

Thus, on the front end, it is your responsibility to make sure that the DBF contains exactly the data you want it to contain because it's all going down to the handheld. Then, it's your responsibility to grab the entire DBF and handle any possible conflict between it and the data on your desktop. You do this by writing code in the two methods I mentioned earlier that move data to and from the DBF and your desktop.

But, hey, you're a database developer – so that's the fun part, right?

Stay tuned!

So that's how the mechanics of Satellite Forms works. In the following months, I'll dig into the Satellite Forms IDE, describe the SatForms object model, and discuss how to build multi-table, event driven applications on the Palm. I'll also cover writing code for the SatForms ActiveX control that synchronizes data between the intermediate dBASE 5 DBFs and your PC application's data store.

## sidebar

Whatcha gonna name it?

You'll see me use a number of terms, seemingly interchangeably, to refer to the physical palm-sized computer device. Why am I appearing to be schizophrenic?

Way back a zillion years ago, the original device from 1996 was called a Pilot. The next generation, in 1997, added the "Palm" moniker, and they became "PalmPilots." Then someone from the Pilot Pen corporation noticed, and picked up the phone to their legal department. As all good lawyers do, a flurry of "cease and desist from using our trademark" letters landed in 3Com's mailbox. The Pilot moniker fell out of favor and the next generation of devices were simply "Palm" handhelds: Palm III, Palm IIIx, Palm V, Palm VII, and so on.

But that's not all. The original inventors of the Palm device and OS, Jeff Hawkins and Donna Dubinski, kept pestering 3Com to spin off Palm to take advantage of the IPO feeding frenzy of 1999, and when it didn't happen, they left and created their own company. Handspring Corp.'s first product was the Visor, a "Palm-compatible" that runs on the Palm OS, but that has more features, something like the original Compaq PCs were compatible with the IBM PC, but had additional features.

As a result, the correct term to refer to all of this hardware would be "Palm or Palm-compatible handheld organizers." Yeah, right. I'll use "Palm" and "handheld" interchangeably to refer to the entire family of Palm and Handspring devices.