* I’ve modified ID-Card\_Split\_FE.dbo.qryLotLaneWithOwners\_Master  
  In Sql Server Management Studio, open SunriverDatabase\SubmittalProposal\DatabaseObjects\qryLotLaneWithOwners\_Master.sql, and then execute it (making sure that your current database is ID-Card\_Split\_FE).
  + Ditto with uspSellCheckInspectionUpdate
* I’ve added the following roles
  + ITAdmin (for the new IT Admin page)
  + CanViewOwnerProperty
  + CanDoReportsOwnerProperty (this is for doing envelopes)
* Don’t forget to back up your Crystal Report files (.rpt), and then restore them. You’ve probably spent some time establishing the connection string so you don’t have to re-key in the parameters once the report starts.

Instructions for adding to the ITAdmin page

I am, of course, happy to add things to this page for you. But if you want to do it yourself, here are some instructions.

* Adding new Procedures  
  You can follow the example that I’ve given. It should be straightforward. Note how in the.aspx file I do a *confirm* in javascript in order to give a warning before executing the stored procedure. Also, note how in this example, I’m using RVStorageQLConnectionString as the Connection String. You’ll, of course, want to use the one where your stored procedure is at.
* Adding Queries  
    
  The .aspx side of ITAdmin is obvious. In the code-behind (the ITAdmin.cs file), you should be able to easily copy the code from the example. The code simply redirects to an .aspx page. About that aspx page:
  + Create an .aspx page (right click SubmittalProposal | Add | New item, and then choose **Web Form using a Master Page**. When it prompts you for which Master Page, choose **AbstractQuery2 Master**. Then go over to the code-behind (the .cs page … short-cut: F7), and inherit it from AbstractQueryPage. You’ve got 3 methods that you have to define. Have a look at Query\_CrossReference3.cs. It should be pretty obvious.
* Adding Reports  
    
  Both the ITAdmin.aspx.cs page and the code-behind (ITAdmin.cs) code is simple to emulate. The next part is very much like with Queries. Create the aspx page from the subdirectory Reports. (Right-click Reports). Again, create a Web Form using Master Page; but this time, choose Reports/Reports.Master as the master page. And after doing this, F7 (to go to the code-behind (.cs)) file, and have it inherit from AbstractReport. Back to the .aspx page: you’ll see that by virtue of having built it using Reports.Master as the Master page, that you’ll have a section predefined for you called Content2. Here is where you put in the UI for any input parameters that you may have. Have a look at any of the .aspx pages in the Reports subdirectory for examples.  
    
  Back at the code-behind, you’ll have 6 procedures that you must override
  + child\_Page\_Load  
    In case you want to do something when the first gets loaded. (If it’s an “init” type of thing that only needs to execute once at the beginning, put it inside of the “if (!IsPostBack)” block.)
  + getReportDocument  
    You will have already created a Crystal Reports class. Return an instance of that class.
  + getIgnoreSubreportsWhenBuildingParameters  
    Return true. (If you get into building Crystal Reports with “subreports”: I usually add Parameters to my main report, and use Crystal Reports’ own linking feature; in which case, you can still return true. Only if you add subreports with their own parameters that you haven’t linked to a parameter in the main report … then you would return false (in which case, the framework will go to the trouble of finding their names and pulling their values from getReportParameters(see below))).
  + AbstractReport\_Click  
    Usually, the only line of code you’ll put here is **base.AbstractReport\_Click(sender, e);**  
    But you can utilize this section to do some input editing, and abort running the report if the input is invalid. You can see how I did this in RVPastDue.
  + getReportParameters  
    If the report has parameters, you create a Hashtable of them here.
  + ConnectionString