Simplifying Report Selection for Users

by Dani Beaubien and Guy Algot

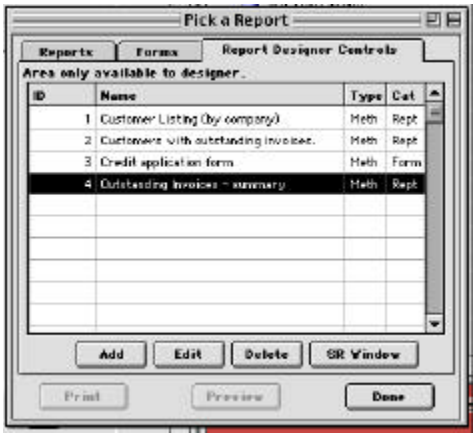
The problem

I didn't even know that I had a problem until much later. The client already had an office system based on 4D, there were a lot of changes to be made and they wanted them yesterday. They were requesting new reports, lots of them. Some were Quick Reports, some SuperReports and some using methods and layouts. Many of the existing reports had their own "way" of being invoked, some by menus, some by buttons and others by choice lists. At one time there were over 30 reports scattered throughout the system. People had a hard time figuring out which report they wanted and where to find it. The administrative reports that were used at month end were scattered across three separate areas. The more reports there are, the harder it is to figure out what each report does. It is nice to have a description and the ability to hide a report if it is not appropriate to view that report in the area that you are in.

The system that I inherited had all the reports stored on disk. I hate to ask the user to go to the disk to find the report that they want as this is very hard to manage in a client/server environment. If the reports are stored centrally on some file server then you rely on the fact that the user has that file server mounted. There are some serious cross-platform issues that need to be dealt with, naming problems, etc.

This article is about the user interface solution that Guy and I have come up with. Guy started the process and I finished it (throughout this article I use "I" rather than "we" just to make it a little easier to read). The "Pick a Report" interface that I have designed is a wonderfully flexible system that hides most of the details from the common user. I try to limit the number of users that have access to report creation. I have found that training users to create their own reports is an expensive task, there are also security issues. The projects where I have done this usually end up with lots of support issues and most of them evolve around report creation. If I restrict report creation to only a few "computer intelligent" users then it makes supporting the product much easier. It also can provide an additional revenue stream when they hire you to do some complicated report rather than trying to do it themselves.

This screen shot shows the UI that a user would see. There are two tabs, one for reports and one for forms. To the right of the report title it states which area the report is for. The description for the report is below the list.



If the user is the designer, there is a third tab as shown above. This tab is used to edit the reports that are stored in the data file. I also added a button to open a super report area just for convenience. You might want to add one for quick reports.

The solution

There were a number of issues that needed to be solved by the new user interface:

- Be able to provide a description of the reports.
- Handle client/server with out having reports all over the place (reports should be stored in datafile).

- Hide the report creation interfaces (for security and support reasons).
- Common interface for all types of reports (Quick Reports, SuperReports, Methods and 4D Write).
- Ability to limit reports that are available in certain areas.
- Ability to have a report based on the current selection.
- Ability to provide a preview capability for the reports.

Figure 1

My solution revolves around storing the reports in a special table (see figure 1). This table contains all the information that a particular report needs to execute. When the report is requested by the user, the report is loaded either directly (SuperReport Pro & 4D Write) or it is written to disk and then referred to (Quick Report) or it is executed (4D method).

When I was doing my initial design work to implement the solution, I divided the problem into two pieces: Building a consistent user interface, and printing the report.

Controlling the user interface

I will not go into every scrap of code that I have used to create this interface. Most is straightforward and is just there to manage the user interface. There are three scripts that contain the essence of the interface. They are in the "Onload" button, in the tab object aReportTabs and in the form method for the form [Predefined_Reports];"Pick_A_Report_d."

```
` define the tabs
ARRAY TEXT(aReportTabs;2)
aReportTabs{1}:="  Reports  "
aReportTabs{2}:="  Forms  "
If (Current user="Designer")
  INSERT ELEMENT(aReportTabs;3;1)
  aReportTabs{3}:="  Report Designer Controls  "
End if
aReportTabs:=1  ` default to the first tab
```

The bOnload button (see above) has code that executes on loading (obviously). This code initializes the tabs. I could have put it in the actual tab object but I didn't.

The aReportTabs tab contains code (see below) for managing page switching. Because of the requirement of the print button (see the next section for what I mean) for the report record to be "loaded," I must ensure that when the user switches pages that the appropriate record is loaded. The print button works on the currently loaded report's record.

```
GOTO PAGE(aReportTabs)  ` switch pages
ENABLE BUTTON(bPrint)
ENABLE BUTTON(bPreview)

Case of
  : (aReportTabs=1)
    oldChoice:=-1
     ` force a refresh of the report tab (see the form method)

  : (aReportTabs=2)
    oldFormChoice:=-1
     ` force a refresh of the forms tab (see the form method)

  : (aReportTabs=3)
     ` don't want to allow the printing from this tab.
     ` this tab is just for editing
    DISABLE BUTTON(bPrint)
    DISABLE BUTTON(bPreview)

    MESSAGES OFF
    ALL RECORDS([Predefined_Reports])
    ORDER BY([Predefined_Reports];
                [Predefined_Reports]Report_ID;>)
    MESSAGES ON
End case   ` (aReportTabs=3)
```

The majority of the code is in the form method (see figure 2). The first section deals with the initialization of the various arrays that are used to track the forms and reports. I also setup a variable to track if the user has changed their selection. This variable is used as a flag to see if the report record needs to be reloaded. For example, if oldFormChoice is not the same as aForms then I know that the user has made a choice in the list of forms. So I find that report and display the description. The second section does that checking and updating of the currently highlighted report or form.

```
` Form Method: Pick_A_Report_d
` File/Layout: [Predefined_Reports];"Pick_A_Report_d"

C_INTEGER(oldChoice;oldFormChoice)
C_BOOLEAN(forceReload)
C_LONGINT(selectedReportID)

MESSAGES OFF

If (Form event=On Load) | (forceReload)
  selectedReportID:=0

   ` ••• Load the report arrays
  QUERY([Predefined_Reports];
          [Predefined_Reports]Category="Rept";*)
  QUERY([Predefined_Reports]; & ;
          [Predefined_Reports]ShowInAllTables=True)
  CREATE SET([Predefined_Reports];"t1")
  QUERY([Predefined_Reports];
```

```
        [Predefined_Reports]Category="Rept";*)
QUERY([Predefined_Reports]; & ;
        [Predefined_Reports]ShowInAllTables=False;*)
QUERY([Predefined_Reports]; & ;
        [Predefined_Reports]TableNumber=Table(pTable))
CREATE SET([Predefined_Reports];"t2")
UNION("t1";"t2";"t1")
USE SET("t1")
ORDER BY([Predefined_Reports];[Predefined_Reports]Name;>)
ARRAY STRING(40;aReports;
            Records in selection([Predefined_Reports]))
ARRAY STRING(30;aRepArea;
            Records in selection([Predefined_Reports]))
FIRST RECORD([Predefined_Reports])
For ($i;1;Records in selection([Predefined_Reports]))
  aReports{$i}:=[Predefined_Reports]Name
  aRepArea{$i}:=Table name(
                [Predefined_Reports]TableNumber)
  If (Not([Predefined_Reports]ShowInAllTables))
    aRepArea{$i}:="only "+aRepArea{$i}
      ` let the user that is only on this table
  End if
  NEXT RECORD([Predefined_Reports])
End for
aReports:=Num(Size of array(aReports)>0)
aRepArea:=aReports
theDesc:=""
oldChoice:=-1

  ` ••• Load the form arrays
QUERY([Predefined_Reports];
        [Predefined_Reports]Category="Form";*)
QUERY([Predefined_Reports]; & ;
        [Predefined_Reports]ShowInAllTables=True)
CREATE SET([Predefined_Reports];"t1")
QUERY([Predefined_Reports];
        [Predefined_Reports]Category="Form";*)
QUERY([Predefined_Reports]; & ;
        [Predefined_Reports]ShowInAllTables=False;*)
QUERY([Predefined_Reports]; & ;
        [Predefined_Reports]TableNumber=Table(pTable))
CREATE SET([Predefined_Reports];"t2")
UNION("t1";"t2";"t1")
USE SET("t1")
ORDER BY([Predefined_Reports];[Predefined_Reports]Name;>)
ARRAY STRING(60;aForms;
            Records in selection([Predefined_Reports]))
ARRAY STRING(30;aFrmArea;
            Records in selection([Predefined_Reports]))
FIRST RECORD([Predefined_Reports])
For ($i;1;Records in selection([Predefined_Reports]))
  aForms{$i}:=[Predefined_Reports]Name
 aFrmArea{$i}:=Table name([Predefined_Reports]TableNumber)
  If (Not([Predefined_Reports]ShowInAllTables))
    aFrmArea{$i}:="only "+aFrmArea{$i}
  End if
  NEXT RECORD([Predefined_Reports])
End for
aForms:=Num(Size of array(aForms)>0)
aFrmArea:=aForms
theFormDesc:=""
oldFormChoice:=-1

  ` Reset our flag
  forceReload:=False

  ` do this for the third tab
  MESSAGES OFF
  ALL RECORDS([Predefined_Reports])
  ORDER BY([Predefined_Reports];
            [Predefined_Reports]Report_ID;>)
  MESSAGES ON

  CLEAR SET("t1")
  CLEAR SET("t2")
End if


  ` used to update the list
If (Form event=On Load) | (Form event=On Clicked) |
        (Form event=On Double Clicked)

  Case of
      ` Reports Tab
    : (aReportTabs=1) & (Size of array(aReportTabs)>1)
    If (Size of array(aReports)<aReports)
      aReports:=Size of array(aReports)
    End if

    If (oldChoice # aReports)
      If (aReports=0)
          ` make sure that the user has not clicked a blank line
        aReports:=oldChoice

      Else
        If (aReports>0)
          QUERY([Predefined_Reports];
                [Predefined_Reports]Name=aReports{aReports})
          theDesc:=[Predefined_Reports]Description
          selectedReportID:=[Predefined_Reports]Report_ID
          If ([Predefined_Reports]disablePreview)
            DISABLE BUTTON(bPreview)
          Else
            ENABLE BUTTON(bPreview)
          End if
        End if
        oldChoice:=aReports
      End if  ` (aReports=0)
    End if   ` (oldChoice # aReports)

    If (aReports=0)
      theDesc:=""
      selectedReportID:=0
    End if
    aRepArea:=aReports


      ` Forms Tab
    : (aReportTabs=2) | (Size of array(aReportTabs)=1)
      ` just forms displayed
    If (Size of array(aForms)<aForms)
      aForms:=Size of array(aForms)
    End if

    If (oldFormChoice # aForms)
      If (aForms=0)
          ` make sure that the user has not clicked a blank line
        aForms:=oldFormChoice
```

```
    Else
      If (aForms>0)
      QUERY([Predefined_Reports];[Predefined_Reports]Name=
                  aForms{aForms})
        theFormDesc:=[Predefined_Reports]Description
        selectedReportID:=[Predefined_Reports]Report_ID
        If ([Predefined_Reports]disablePreview)
          DISABLE BUTTON(bPreview)
        Else
          ENABLE BUTTON(bPreview)
        End if
      End if
      oldFormChoice:=aForms
    End if   ` (aForms=0)
  End if   ` (oldFormChoice # aForms)

  If (aForms=0)
    theFormDesc:=""
    selectedReportID:=0
  End if
  aFrmArea:=aForms
 End case

End if   ` (Form event=On Load) | (Form event=On Clicked)

MESSAGES ON
```

Figure 2

If the current user is the designer then they have access to a third tab. This tab gives access to the maintenance of the reports themselves. The input screen for the [Predefined_Reports] table is pretty straightforward (see figure 3).

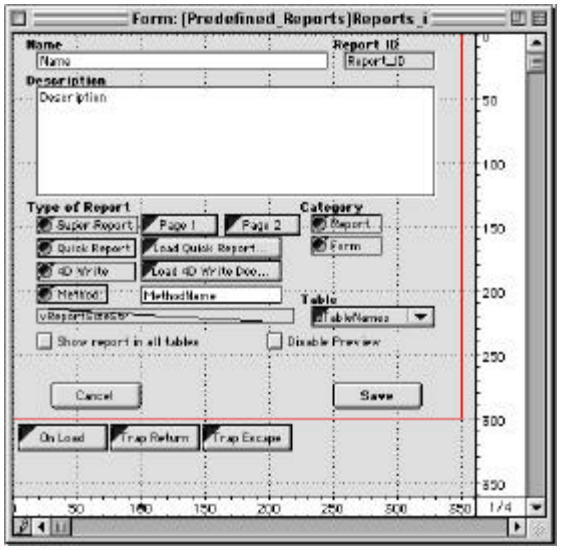


Figure 3

Check out the sample database (“*ACI_Video.sea*”) for the code that is in this form. It is pretty straight forward. There is a 2nd and 3rd page that contains SuperReport Pro external areas and the 4th page has a 4D Write external area on it. These are used to import the various reports into the record.

Printing the selected report

All of the printing code is in the bPrint button. Depending on the type of the report, different code is used to print (or preview) the report.

There are two buttons that can cause a report to print. One is “Print” and the other is “Preview.” Rather than having two copies of the code in each of these buttons, when the user presses the preview button a process variable is set and the preview button posts a Cmd-P event which triggers the print button. It works very well. The code from the bPrint button is in figure 4:

```
---------------------------------------------------------------------------------
 ` Object Method: bPrint
 ` By:  Dani Beaubien & Guy Algot
 ` File/Layout:  [zDialogs];"Pick_a_Report_d"
 ` Description:  This is the meat of the pick a report system.
 ` Depending on the type of the report different code is used
 ` to print/preview the report.

 `#Start method

C_BOOLEAN(doPreview)
C_LONGINT(sr_windowID)
C_PICTURE(vSRArea;vSRArea2)

If (selectedReportID # 0)
   ` make sure that the selected report is loaded

 QUERY([Predefined_Reports];[Predefined_Reports]Report_ID=
            selectedReportID)

 If (Records in selection([Predefined_Reports])=1)
  Case of

     ` •• print a SUPER REPORT
   : ([Predefined_Reports]Type="SRep")
    If (Picture size([Predefined_Reports]theReport_p1)>0) |
            (Picture size([Predefined_Reports]theReport_p2)>0)
      OK:=1
      ` is there a page 1 to print?
    If (Picture size([Predefined_Reports]theReport_p1)>0) &
                        (OK=1)
      vSRArea:=[Predefined_Reports]theReport_p1
      $windowType:=8
      If (doPreview)
        sr_windowID:=SR Preview (vSRArea;50;50;600;
                  Screen height-50;$windowType;"Preview: "+
                  [Predefined_Reports]Name)
      Else
        $windowTitle:=Get window title
        ` doing this will name the document properly when printing
        SET WINDOW TITLE([Predefined_Reports]Name)
        $Error:=SR Print Report (vSRArea;3;0)
        SET WINDOW TITLE($windowTitle)
      End if
      OK:=1
    End if



      ` is there a page 2 to print?
    If (Picture size([Predefined_Reports]theReport_p2)>0) &
                        (OK=1)
```

```
        vSRArea2:=[Predefined_Reports]theReport_p2
        If (doPreview)
          sr_windowID:=SR Preview (vSRArea2;50;50;
              Screen width-50;Screen height-50;$windowType;
              " Page 2 Preview: "+[Predefined_Reports]Name)
        Else
          $windowTitle:=Get window title
          ` doing this will name the document properly when printing
          ` 11.18.98.gj
          SET WINDOW TITLE([Customers]CustomerID+" page 2")
          $Error:=SR Print Report (vSRArea2;3;0)
          SET WINDOW TITLE($windowTitle)
        End if
      End if
    Else
      BEEP
      ALERT("No report defined for this item")
    End if


` •• print 4D Write Doc
  : ([Predefined_Reports]Type="4Dwt")
    If (doPreview)
      If (Screen width>650)
        $width:=642
      Else
        $width:=600
      End if
      $height:=Screen height-60
      vLeft:=(Screen width-$width)/2
      vTop:=70+(Screen height-$height)/3
      vRight:=vLeft+$width
      vBottom:=vTop+$height
      Open window(vLeft;vTop;vRight;vBottom;5;"Preview: "+
                  [Predefined_Reports]Name;"CloseWindow")
      DIALOG([zDialogs];"View4DWriteDoc_d")
      CLOSE WINDOW
    Else
      C_LONGINT(someWriteDoc2)
      someWriteDoc2:=WR Picture to offscreen area (
                  [Predefined_Reports]theReport_p1)
      WR DO COMMAND (someWriteDoc2;208) `Select all text
      WR DO COMMAND (someWriteDoc2;713)
        ` freeze document
      WR PRINT (someWriteDoc2;1) ` print one copy
      WR DELETE OFFSCREEN AREA (someWriteDoc2)
    End if


    ` •• print quick report
  : ([Predefined_Reports]Type="QRep")
    If (BLOB size([Predefined_Reports]theBLOB)>0)
      OK:=1
      BLOB TO DOCUMENT("zzRepXX";
                  [Predefined_Reports]theBLOB)
       ` if the hard drive is full, should trap the error here
       ••••••••••••••••••••••••
       ` should we check that on launch rather than here??
      SET PRINT PREVIEW(doPreview)
      $windowTitle:=Get window title
      ` doing this will name the document properly when printing
      SET WINDOW TITLE([Predefined_Reports]Name)
      REPORT(Table([Predefined_Reports]TableNumber)->;
            "zzRepXX";*)
      SET WINDOW TITLE($windowTitle)
      DELETE DOCUMENT("zzRepXX")
  Else
      BEEP
      ALERT("No report defined for this item.")
    End if


      ` •• call the method
    : ([Predefined_Reports]Type="Meth")
      EXECUTE([Predefined_Reports]MethodName)
  End case
  SET PRINT PREVIEW(False)

 Else
  BEEP
  BEEP
  ALERT("bPrint button method";"selected report ID doesn't
        exist")
  TRACE
 End if
Else
 BEEP
 ALERT("You must select a report or a form to print first.")
End if

doPreview:=False
```

Figure 4

You might want to cut the code out of the bPrint button and put it into a method. This way you could call it from different places in your code. Just pass the report ID of the report that you want to print and watch it come out. Very handy.

I have also created reports and have hidden them entirely from allowing the user to choose them, by setting it to appear only in my constants table. Then I can have the report printed from another part of the system. I have used this for progress notes from input screens. The client loves it.

Final comments and possible enhancements

I don't believe in a canned solution that I apply to all my clients. I customize my standard suite of interfaces to suit each and every project. I find that the most interesting things that I do are in these customizations. If it is useful then I incorporate it in my standard suite of interfaces.

The sample database that I have provided is a modified version of the ACIVIDEO sample database. In the customers area, I have added a button labeled "Pick a Report" that opens the "Pick a Report" interface. I have left the reporting interface that was already in the sample database for you to compare against. I urge for you to take a look and hopefully there is something that Guy and I have done that you can use in your future projects.

The biggest gotcha that I have come across is that since I am printing many different types of reports, the previews and printing windows and dialogs that are shown to the user are different. I can give a common user experience all the way to the point that they press the "Print" button. Perhaps in a future

version of 4D there will be a way to have a little more control for us developers.

One enhancement I added was for a particular client who had a lot of reports that relied on user specified date ranges. They didn't want to do date searches all the time to find the records to report on. My solution was to add a Boolean field to the reports table to indicate if the user should be asked for the date range. If this field was true then the user was asked and the values were placed in process variables vStartDate and vEndDate. The report was then invoked and it was able to use those two variables to do the pre-searches that were needed. Other enhancements that could be made are:

- Add security features so that the reports that a user sees are based on their access level or by the department that the user is in.
- Utilize v6.5 features to move form objects around to stored "custom" layouts that the users have designed. Very cool feature!
- Add Labels to the possibilities of the type of reports to print.
- Have different categories of reports and display them in the interface by category. For example, admin. reports, financial reports, etc.
- Allow end users to add their own reports, and who has access to those reports. This means that they could develop their own personal reports just for their usage.

Please feel free to send me an email if you have any questions or if you would like me to send you a copy of the sample database that I put together.

About the authors

Dan Beaubien and Guy Algot have been working together with 4D for four years. Guy has been using 4D since 1990 and Dan since 1994. They currently devote most of their 4D expertise to developing really cool special education solutions for EIS, located in Carrboro, NC, although they both love living in Alberta, Canada. They can be contacted at *dbeaubien@powersurfr.com* or *guya_gdc@agt.net*.