What's Inside this Guide:

Discern

Getting Started with Discern MPages

Table of Contents

Introduction

Discern MPages® is a *Cerner Millennium*® solution that enables customized display of data at the Organizer or Chart Level from within *PowerChart*®, *FirstNet*®, *INet*® and *SurgiNet*®. *MPages* can display output in ASCII, .RTF, .PDF, PostScript and HTML formats. Using HTML allows the creation of interactive output. For example, a summary report can be created that includes links to navigate down to more detailed information.

An updated version of *Discern Explorer*® Help is distributed with *MPages* that includes instructions for using Preference Maintenance (PrefMaint.exe) to create an MPage within *PowerChart*, *FirstNet*, *INet* and *SurgiNet* that will execute a custom *Discern Explorer* program. Programs executed as *MPages* can be created using VisualExplorer.exe (VE), DiscernVisualDeveloper.exe (DVDev) or any text editor. To be executed as an MPage, the program must prompt for an output device and direct the output to that device. The query that produces the final output must use report writer sections or the control options Format and Separator = " ".

Note: *MPages* was originally called Discern Desktop. The solution was briefly renamed Discern Pages before finally being renamed *MPages*.

Context Variables

MPages populates the following context variables:

Context Variable	Definition	Data Type
PAT_PersonID	The person ID of the patient currently loaded in *PowerChart*, *FirstNet*, *INet* or *SurgiNet*. This is available at the Chart level.	Number
VIS_EncntrID	The encounter ID of the patient currently loaded in *PowerChart*, *FirstNet*, *INet* or *SurgiNet*. This is available at the Chart level.	Number
USR_PersonID	The person ID of the user running *PowerChart*, *FirstNet*, *INet* or *SurgiNet*. This is available in Discern Notification, as well as the Organizer and Chart levels.	Number
DEV_Location	The configured location of the device (or client). This is equivalent to the device_location logical configured in the WTS Location Toolkit (WTSLocation.exe). This is available in Discern Notification, as well as the Organizer and Chart levels.	String
APP_AppName	The name of the executable that is currently in context. This is available in Discern Notification, as well the Organizer and Chart level.	String

These context variables are useful for substituting runtime values from the current solution context during the execution of a *Discern Explorer* program. The variable names should be nested within dollar signs ($) in order to substitute the appropriate context data.

The following source code can be used to create a simple program to display the values of the context variables.

```
drop program CCL_MP_TEST_CONTEXT_VARS go
create program CCL_MP_TEST_CONTEXT_VARS
/*
Use this program as an example for testing the setup of a program that is executed as an MPage.
The output of the program will display the value of the context variables that are passed to
MPages.
 */
prompt
  "Output to File/Printer/MINE " = "MINE"
  , "Person ID" = 0
  , "Encounter ID" = 0
  , "User ID" = 0
  , "Location Value" = " "
  , "Application" = " "

with OUTDEV, PID, EID, UID, LOC, APP

SELECT INTO $OUTDEV
    PID_SE = $PID
  , EID_SE = $EID
  , UID_SE = $UID
  , LOC_SE = $LOC
  , APP_SE = $APP
FROM DUMMYT
WITH FORMAT, SEPARATOR = " "

end
go
```

After compiling the above source code, the following process can be used to execute the program as an MPage from *PowerChart*:

1. Run Preference Maintenance (Prefmaint.exe).

2. In the Application field, select *PowerChart*.

3. Filter by the position you want and select the option *Search for Preferences*.

4. Select View > View PVC_Name to filter preferences by the PVC Name.

5. From the Level control, expand the solution tree and the appropriate level (Organizer or Chart).

6. Click the Chart level, and then click the Add Tab button.

7. From the Available Tabs list, select *Discern Report* (the component reference name for MPages). Click the down arrow to move Discern Report to the Existing Tabs box. Discern Report is moved to the end of the Existing Tabs list.

8. Click Apply.

9. Click OK to save the new tab assignment. Discern Report is displayed as an option under the Chart level.

10. Select the Discern Report option. A list of preferences is displayed in the Existing Preference box.

11. To define the title of the MPage, enter **Test MPages Context Variables** in the VIEW_CAPTION preference.

12. In the Level tree, select the *Discern Report* component.

13. Locate the REPORT_NAME preference.

14. Enter **CCL_MP_TEST_CONTEXT_VARS** (the *Discern Explorer* program name) as the REPORT_NAME preference. CCL group names are allowed. For example, use **CCL_MP_TEST_CONTEXT_VARS:group1** if the above source code was compiled using a group1 account to create a group1 program.

15. When using a *Discern Explorer* program in the REPORT_NAME preference, the REPORT_PARAM preference column is optional and can be used to enter a parameter list that will be passed to the program specified in the REPORT_NAME preference. The list of parameters must be comma-separated with text strings in quotations. The context variables can be passed as parameters by encapsulating the variable names in the '$' character. Enter the following in the REPORT_PARAM: preference

"MINE", $PAT_PersonID$, $VIS_EncntrID$, $USR_PersonID$, "$DEV_Location$", "$APP_AppName$".

Since $DEV_Location$ and $APP_AppName$ are string data types they must be enclosed in the quotation characters.

If the REPORT_PARAM preference is not used and the *Discern Explorer* program that was entered in the REPORT_NAME preference has an associated prompt from, the prompt form will be displayed to the user when the MPage is selected.

If the REPORT_PARAM preference is used, the parameter values will be assigned to the prompt name symbols in the order they are listed in the With section of the prompt clause in the program. The CCL_MP_TEST_CONTEXT_VARS example program uses the following prompt clause:

```
prompt
  "Output to File/Printer/MINE " = "MINE"
  , "Person ID" = 0
  , "Encounter ID" = 0
  , "User ID" = 0
  , "Location Value" = " "
  , "Application" = " "

with OUTDEV, PID, EID, UID, LOC, APP
```

Using the REPORT_PARAM preference shown above, the literal value "MINE" is assigned to the prompt named OUTDEV. The value of $PAT_PersonID$ is assigned to the prompt named

PID. The value of $VIS_EncntrID$ is assigned to the prompt named EID, and so on. The program can then reference the values assigned to the prompts by prefixing the $ character to the prompt name, such as $Outdev, $PID, $EID, etc.

16. Click Apply and then OK.

17. If PowerChart is currently open, close it.

18. Re-open *PowerChart*. *Test MPages Context Variables* should be displayed as a tab or table of context option. Opening the tab or table of context option executes the CCL_MP_TEST_CONTEXT_VARS program and displays the values of the context variables that were passed to it.

Example Patient Summary Report with ASCII Output

The following source code creates a program that uses several select statements and record structures to get a patient's problems, vital signs, and allergies. All of this information is available in *PowerChart*. However an MPage can be used to display this information in a single custom view. The output of the program created by the following source code is simple ASCII text. The following source code could be easily created using VE (VISUALEXPLORER.EXE), DVDev (DiscernVisualDeveloper.exe), or any text editor.

```
drop program CCL_MP_GET_ALLERGY_VS_PROB go
create program CCL_MP_GET_ALLERGY_VS_PROB

prompt
  "Output to File/Printer/MINE" = "MINE"
  , "Enter a Person ID" = 0
  , "Enter an Encounter ID " = 0
with OUTDEV, PID, EID

/***********
The following queries are provided as examples to help demonstrate the functionality of MPages.
They are not intended to be used for clinical decision making or support.  Site-specific
modifications need to be made to these examples to ensure they return accurate information
when executed in any environment. You MUST validate the data returned by these example
queries to ensure correctness and accuracy.
************/

record allergy (
 1 allergy_cnt = i4
 1 list [*]
   2 description = vc
   2 onset = c15
   2 reaction = vc   )

record problem (
 1 problem_cnt = i4
 1 list [*]
```

```
   2 description = vc
   2 onset = c15   )

record vs_rec (
  1 vlist [*]
   2 date = vc
   2 temp = c10
   2 temp_type = c5
   2 heart_rate = c10
   2 resp_rate = c10
   2 sys_bp = c10
   2 dia_bp = c10   )



declare reac_string = vc with protect
declare active = f8 with constant(uar_get_code_by("MEANING",12025,"ACTIVE")),protect
declare proposed = f8 with constant(uar_get_code_by("MEANING",12025,"PROPOSED")),protect

/*****
Set Vital Sign Variables

Since the Vital Signs code values are from code_set 72, older versions of uars will be
less efficient than a select from the code_value table.  Therefore these variables are
set using a select instead of a uar.
******/
declare tempo_var = f8 with NoConstant(-1.0),Protect
declare tempe_var = f8 with NoConstant(-1.0),Protect
declare tempr_var = f8 with NoConstant(-1.0),Protect
declare pulse_var = f8 with NoConstant(-1.0),Protect
declare resp_var = f8 with NoConstant(-1.0),Protect
declare sys_var = f8 with NoConstant(-1.0),Protect
declare dias_var = f8 with NoConstant(-1.0),Protect
declare hr_var = f8 with NoConstant(-1.0),Protect
declare apihr_var = f8 with NoConstant(-1.0),Protect

select into "nl:"
  c.code_value
from
  code_value   c
where c.code_set = 72
  and c.display_key in
  ("TEMPERATUREORAL",
  "TEMPERATURETYMPANIC",
  "TEMPERATURERECTAL",
  "PULSERATE",
  "RESPIRATORYRATE",
  "SYSTOLICBLOODPRESSURE",
  "DIASTOLICBLOODPRESSURE",
  "HEARTRATE",
  "APICALHEARTRATE")
  and c.active_ind = 1
  and c.begin_effective_dt_tm <= cnvtdatetime(curdate,curtime3)
  and c.end_effective_dt_tm >= cnvtdatetime(curdate,curtime3)
detail
```

```
  case(c.display_key)
    of "TEMPERATUREORAL"    : tempo_var = c.code_value
    of "TEMPERATURETYMPANIC"  : tempe_var = c.code_value
    of "TEMPERATURERECTAL"    : tempr_var = c.code_value
    of "PULSERATE"      : pulse_var = c.code_value
    of "RESPIRATORYRATE"     : resp_var = c.code_value
    of "SYSTOLICBLOODPRESSURE"   : sys_var = c.code_value
    of "DIASTOLICBLOODPRESSURE"  : dias_var = c.code_value
    of "HEARTRATE"      : hr_var = c.code_value
    of "APICALHEARTRATE"   : apihr_var  = c.code_value
  endcase
with nocounter

/*****
Get Allergies
******/
select into "nl:"
  al.person_id
  , al.allergy_instance_id
  , allergy_desc = if(al.substance_ftdesc > " ")
        substring(1,60,al.substance_ftdesc)
      elseif(n.source_string > " ")
        substring(1,60,n.source_string)
      else
        "Not Specified"
      endif
  , onset = format(al.onset_dt_tm,  "@SHORTDATE4YR")
  , r.reaction_id
  , reaction_desc = if(r.reaction_ftdesc > " ")
        substring(1,60,r.reaction_ftdesc)
      elseif(n2.source_string > " ")
        substring(1,60,n2.source_string)
      else
        "Not Specified"
      endif
from
  allergy   al
  , nomenclature   n
  , reaction   r
  , nomenclature   n2
plan al where
    al.person_id = $pid
    and  al.active_ind=1
    and  al.beg_effective_dt_tm <= cnvtdatetime (curdate, curtime3)
    and  (al.end_effective_dt_tm >= cnvtdatetime (curdate, curtime3)
     or   al.end_effective_dt_tm = null)
     and  al.reaction_status_cd in (active, proposed)
join n where
    n.nomenclature_id = al.substance_nom_id
join r where
    r.allergy_id = outerjoin(al.allergy_id)
join n2 where
    n2.nomenclature_id = outerjoin(r.reaction_nom_id)
order by
    onset desc
```

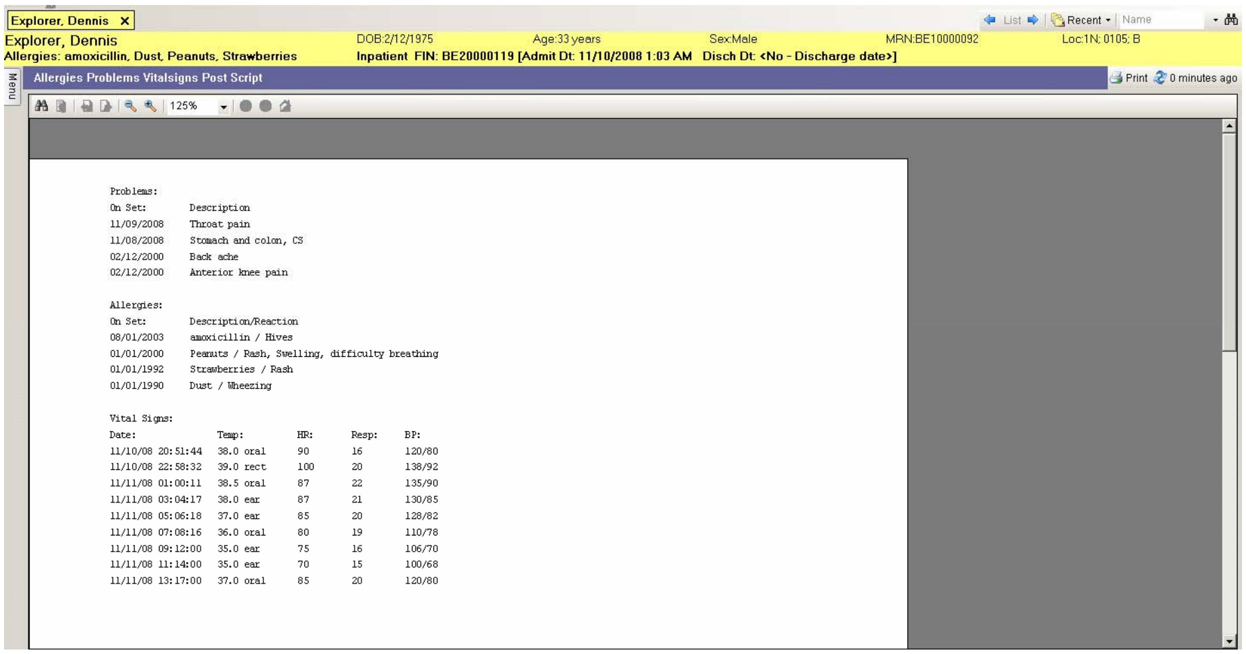
```
     , al.allergy_instance_id
     , 0
head report
  allergy_cnt = 0
head al.allergy_instance_id
  reac_string = " "
  allergy_cnt = allergy_cnt +1
detail
  if(mod(allergy_cnt,10) = 1)
    stat = alterlist(allergy->list, allergy_cnt +9)
  endif
   allergy->list[allergy_cnt].reaction = trim(reaction_desc)
  allergy->list[allergy_cnt].description = allergy_desc
  allergy->list[allergy_cnt].onset = onset
foot al.allergy_instance_id
  allergy->allergy_cnt = allergy_cnt
foot report
  stat = alterlist(allergy->list, allergy_cnt)
with format, separator = " "

;call echorecord(allergy)

/*****
Get Problems
******/
select into "nl:"
  p.problem_instance_id
  , p.annotated_display
  , p_classification_disp = uar_get_code_display(p.classification_cd)
  , p_course_disp = uar_get_code_display(p.course_cd)
  , problem_desc = if(n.nomenclature_id = 0.0)
        p.problem_ftdesc
      else
        n.source_string
      endif
  , onset = format(p.onset_dt_tm,  "@SHORTDATE4YR")
from
  problem   p
  , nomenclature   n
plan p where p.person_id =$pid
join n where p.nomenclature_id = n.nomenclature_id
order by
  p.onset_dt_tm   desc
  , p.problem_instance_id
head report
  problem_cnt = 0
head p.problem_instance_id
  problem_cnt = problem_cnt +1
detail
  if(mod(problem_cnt,10) = 1)
    stat = alterlist(problem->list, problem_cnt +9)
  endif
  problem->list[problem_cnt].description = problem_desc
  problem->list[problem_cnt].onset = onset
foot p.problem_instance_id
```

```
  problem->problem_cnt = problem_cnt
foot report
  stat = alterlist(problem->list, problem_cnt)
with nocounter, separator=" ", format

;call echorecord(problem)

/*****
Get Vital Signs
******/
select into "nl:"
  c.encntr_id
  , c.performed_dt_tm "@SHORTDATETIME"
  , end_dt_tm = format(c.event_end_dt_tm, "@SHORTDATETIME")
  , c.event_cd
  , c_event_disp = uar_get_code_display(c.event_cd)
  , real_result = cnvtreal(c.result_val)
  , c.result_val
  , c_result_units_disp = uar_get_code_display(c.result_units_cd)
  , c_event_dk = uar_get_displaykey(c.event_cd)
  , c_result_units_dk = uar_get_displaykey(c.result_units_cd)
  , c.event_end_dt_tm
from
  clinical_event   c
  , dummyt   d1
plan c where c.encntr_id   = $eid
  and c.person_id = $pid
  and c.event_cd in(tempo_var, tempr_var, tempe_var,pulse_var,
          resp_var,sys_var,dias_var,hr_var,apihr_var)
  /*Uncomment the following line to only get vital signs from the last two days.*/
  ;and c.event_end_dt_tm >= cnvtlookbehind("2,D")
join d1 where cnvtreal(c.result_val) > 0.0

order by
  c.event_end_dt_tm   desc
  , end_dt_tm   desc
  , c_event_disp
head report
  vs_cnt = 0
head end_dt_tm
  vs_cnt = vs_cnt +1
  if(mod(vs_cnt,20) = 1)
    stat = alterlist(vs_rec->vlist, vs_cnt +19)
  endif
  vs_rec->vlist[vs_cnt].date = end_dt_tm
  vs_rec->vlist[vs_cnt].temp   = "-"
  vs_rec->vlist[vs_cnt].temp_type  = "-"
  vs_rec->vlist[vs_cnt].heart_rate  = "-"
  vs_rec->vlist[vs_cnt].resp_rate = "-"
  vs_rec->vlist[vs_cnt].sys_bp  = "-"
  vs_rec->vlist[vs_cnt].dia_bp  = "-"
detail
  case (c.event_cd)
    of tempo_var:
      vs_rec->vlist[vs_cnt].temp_type = "oral"
      vs_rec->vlist[vs_cnt].temp = c.result_val
    of tempr_var:
      vs_rec->vlist[vs_cnt].temp_type = "rect"
      vs_rec->vlist[vs_cnt].temp = c.result_val
    of tempe_var:
      vs_rec->vlist[vs_cnt].temp_type = "ear"
      vs_rec->vlist[vs_cnt].temp = c.result_val
    of pulse_var:
      vs_rec->vlist[vs_cnt].heart_rate = c.result_val
    of resp_var:
      vs_rec->vlist[vs_cnt].resp_rate = c.result_val
    of sys_var:
      vs_rec->vlist[vs_cnt].sys_bp = c.result_val
    of dias_var:
      vs_rec->vlist[vs_cnt].dia_bp = c.result_val
    of hr_var:
      vs_rec->vlist[vs_cnt].heart_rate = c.result_val
    of apihr_var:
      vs_rec->vlist[vs_cnt].heart_rate = c.result_val
  endcase
foot report
  stat = alterlist(vs_rec->vlist, vs_cnt)
with nocounter, separator=" ", format

;call echorecord(vs_rec)

/*****
Display output
******/
select into $outdev
  vDATE = substring(1, 30, vs_rec->vlist[d1.seq].date)
  , vTEMP = concat(trim(vs_rec->vlist[d1.seq].temp), " ",
      trim(vs_rec->vlist[d1.seq].temp_type))
  , vHEART_RATE = vs_rec->vlist[d1.seq].heart_rate
  , vRESP_RATE = vs_rec->vlist[d1.seq].resp_rate
  , vBP = if(vs_rec->vlist[d1.seq].sys_bp != "-")
      concat(trim(vs_rec->vlist[d1.seq].sys_bp), "/", trim(vs_rec->vlist[d1.seq].dia_bp))
    else  "-"
    endif
from
  (dummyt   d1  with seq = value(size(vs_rec->vlist, 5)))

plan d1

order by
  vDATE
head report
  ;display problems
  col 10 "Problems:" row +1
  col 10 "On Set:"
  col 25 "Description" row +1
  for(pcnt = 1 to problem->problem_cnt )
    col 10 problem->list[pcnt].onset
    col 25 problem->list[pcnt].description
    row +1
  endfor
  row +1
  ;display allergies
  col 10 "Allergies:" row +1
  col 10 "On Set:"
  col 25 "Description/Reaction" row +1
  for(acnt = 1 to allergy->allergy_cnt )
    col 10 allergy->list[acnt].onset
    reac_string = concat(allergy->list[acnt].description, " / "
        , allergy->list[acnt].reaction)
    col 25 reac_string
    row +1
  endfor
  row +1
  ;display vital signs header
  col 10 "Vital Signs:" row +1
  col 10 "Date:"
  col 30 "Temp:"
  col 45 "HR:"
  col 55 "Resp:"
  col 65 "BP:"
  row +1
detail  ;display vital signs
  col 10 vDate
  col 30 vTemp
  col 45 vHeart_Rate
  col 55 vResp_Rate
  col 65 vBP
  row +1
with nocounter, separator=" ", format

end
go
```

The preceding example contains prompts for an Output device, a Person ID, and an Encounter ID. These prompts allow the program created by compiling the above source code example to be executed as a prompt program from ExplorerMenu.exe (EM), DVDev (DiscernVisualDeveloper.exe), VE (VISUALEXPLORER.EXE) or a back-end interactive Discern Explorer session. Executing the program using one of those methods requires the user to enter a Person ID and an Encounter ID. The Discern Prompt Builder can be used to create a prompt form that allows the user to select a Person ID and an Encounter ID from a list.

The program created by compiling the above source code can be set up to execute as an MPage from *PowerChart* by using Preference Manager (PrefMaint.exe) and entering **CCL_MP_GET_ALLERGY_VS_PROB** as the REPORT_NAME preference and **"MINE", $PAT_PersonID$, $VIS_EncntrID$** as the REPORT_PARAM preference.

The output from executing the above program as an MPage should look something like the following example:



Explorer, Dennis DOB:2/12/1975 Age:33 years Sex:Male MRN:BE10000092 Loc:1N; 0105; B
Allergies: amoxicillin, Dust, Peanuts, Strawberries Inpatient FIN: BE20000119 [Admit Dt: 11/10/2008 1:03 AM Disch Dt: <No - Discharge date>]

Allergies Problems Vitalsigns Post Script

```
Problems:
On Set:        Description
11/09/2008     Throat pain
11/08/2008     Stomach and colon, CS
02/12/2000     Back ache
02/12/2000     Anterior knee pain

Allergies:
On Set:        Description/Reaction
08/01/2003     amoxicillin / Hives
01/01/2000     Peanuts / Rash, Swelling, difficulty breathing
01/01/1992     Strawberries / Rash
01/01/1990     Dust / Wheezing

Vital Signs:
Date:              Temp:        HR:      Resp:    BP:
11/10/08 20:51:44  38.0 oral    90       16       120/80
11/10/08 22:58:32  39.0 rect    100      20       138/92
11/11/08 01:00:11  38.5 oral    87       22       135/90
11/11/08 03:04:17  38.0 ear     87       21       130/85
11/11/08 05:06:18  37.0 ear     85       20       128/82
11/11/08 07:08:16  36.0 oral    80       19       110/78
11/11/08 09:12:00  35.0 ear     75       16       106/70
11/11/08 11:14:00  35.0 ear     70       15       100/68
11/11/08 13:17:00  37.0 oral    85       20       120/80
```

Your display will be different based on your *PowerChart* preferences and your data.

The CCL_MP_GET_ALLERGY_VS_PROB example uses several selects and record structures to gather data and display it as a simple ASCII summary page. This example can easily be expanded to gather and display additional data.

Review:
Any *Discern Explorer* program that

- prompts for an output device
- selects into that output device
- uses either the control options Format and Separator = " " or report writer clauses

can be set up to execute as an MPage. The program can be created using VE (VISUALEXPLORER.EXE), DVDev (DiscernVisualDeveloper.exe), or any text editor. The program can create ASCII output or use DIO to create PostScript, .PDF or other common output types. Layout programs created using DVDev (DiscernVisualDeveloper.exe) and Layout Builder can also be executed as an MPage.

Example Patient Summary Report with PostScript/PDF Output Created in Layout Builder

The appearance of the preceding CCL_MP_GET_ALLERGY_VS_PROB example can be greatly improved using Layout Builder within DVDev (DiscernVisualDeveloper.exe) to create PostScript or .PDF style output. There is a Layout Builder tutorial on the Discern Explorer CMSG on Cerner.com that provides a comprehensive introduction to using the Layout Builder in specific code releases. The following examples were created using the 2007.18 release of Layout Builder and assume you are familiar with using Layout Builder to create layout programs. If you are using a different version of Layout Builder the exact steps you use may be different from this example. In that case you may need to complete or review the tutorial to determine how to complete a step using your specific version of Layout Builder.

The first step in converting the CCL_MP_GET_ALLERGY_VS_PROB example to a layout program using PostScript or PDF output is to recreate the variables, record structures, and select statements that populate the record structures. This could be done using the various tools in a layout program. However, it may be easier to just copy those commands from the CCL_MP_GET_ALLERGY_VS_PROB example into an include file and then reference that include file in the layout program.

1. Copy and paste the following code example into a blank file in DVDev (DiscernVisualDeveloper.exe), then save that file as an include file. For example you may want to name the file *1_initials*_GET_ALLERGY_VS_PROB.INC and save it in a standard back-end location. If you are working in a multi-node AIX or HP_UX environment, the .inc file must be saved on all nodes or in a directory that is accessible from all nodes.

```
/***********
The following queries are provided as examples to help demonstrate the functionality of MPages.
They are not intended to be used for clinical decision making or support.  Site-specific
modifications need to be made to these examples to ensure they return accurate information
when executed in any environment. You MUST validate the data returned by these example
queries to ensure correctness and accuracy.
************/

record allergy (
  1 allergy_cnt = i4
  1 list [*]
    2 description = vc
    2 onset = c15
    2 reaction = vc   )

record problem (
  1 problem_cnt = i4
  1 list [*]
    2 description = vc
    2 onset = c15   )

record vs_rec (
  1 vlist [*]
    2 date = vc
    2 temp = c10
    2 temp_type = c5
    2 heart_rate = c10
    2 resp_rate = c10
    2 sys_bp = c10
    2 dia_bp = c10   )
```

```
declare reac_string = vc with protect
declare active = f8 with constant(uar_get_code_by("MEANING",12025,"ACTIVE")),protect
declare proposed = f8 with constant(uar_get_code_by("MEANING",12025,"PROPOSED")),protect

/*****
Set Vital Sign Variables

Since the Vital Signs code values are from code_set 72, older versions of uars will be
less efficient than a select from the code_value table.  Therefore these variables are
set using a select instead of a uar.
******/
declare tempo_var = f8 with NoConstant(-1.0),Protect
declare tempe_var = f8 with NoConstant(-1.0),Protect
declare tempr_var = f8 with NoConstant(-1.0),Protect
declare pulse_var = f8 with NoConstant(-1.0),Protect
declare resp_var = f8 with NoConstant(-1.0),Protect
declare sys_var = f8 with NoConstant(-1.0),Protect
declare dias_var = f8 with NoConstant(-1.0),Protect
declare hr_var = f8 with NoConstant(-1.0),Protect
declare apihr_var = f8 with NoConstant(-1.0),Protect

select into "nl:"
  c.code_value
from
  code_value   c
where c.code_set = 72
  and c.display_key in
  ("TEMPERATUREORAL",
  "TEMPERATURETYMPANIC",
  "TEMPERATURERECTAL",
  "PULSERATE",
  "RESPIRATORYRATE",
  "SYSTOLICBLOODPRESSURE",
  "DIASTOLICBLOODPRESSURE",
  "HEARTRATE",
  "APICALHEARTRATE")
  and c.active_ind = 1
  and c.begin_effective_dt_tm <= cnvtdatetime(curdate,curtime3)
  and c.end_effective_dt_tm >= cnvtdatetime(curdate,curtime3)
detail
  case(c.display_key)
    of "TEMPERATUREORAL"    : tempo_var = c.code_value
    of "TEMPERATURETYMPANIC"   : tempe_var = c.code_value
    of "TEMPERATURERECTAL"    : tempr_var = c.code_value
    of "PULSERATE"       : pulse_var = c.code_value
    of "RESPIRATORYRATE"      : resp_var = c.code_value
    of "SYSTOLICBLOODPRESSURE"    : sys_var = c.code_value
    of "DIASTOLICBLOODPRESSURE"  : dias_var = c.code_value
    of "HEARTRATE"       : hr_var = c.code_value
    of "APICALHEARTRATE"    : apihr_var  = c.code_value
  endcase
with nocounter

/*****
```

```
Get Allergies
******/
select into "nl:"
  al.person_id
  , al.allergy_instance_id
  , allergy_desc = if(al.substance_ftdesc > " ")
        substring(1,60,al.substance_ftdesc)
      elseif(n.source_string > " ")
        substring(1,60,n.source_string)
      else
        "Not Specified"
      endif
  , onset = format(al.onset_dt_tm,  "@SHORTDATE4YR")
  , r.reaction_id
  , reaction_desc = if(r.reaction_ftdesc > " ")
        substring(1,60,r.reaction_ftdesc)
      elseif(n2.source_string > " ")
        substring(1,60,n2.source_string)
      else
        "Not Specified"
      endif
from
  allergy   al
  , nomenclature   n
  , reaction   r
  , nomenclature   n2
plan al where
    al.person_id = $pid
    and  al.active_ind=1
    and  al.beg_effective_dt_tm <= cnvtdatetime (curdate, curtime3)
    and  (al.end_effective_dt_tm >= cnvtdatetime (curdate, curtime3)
     or   al.end_effective_dt_tm = null)
     and  al.reaction_status_cd in (active, proposed)
join n where
    n.nomenclature_id = al.substance_nom_id
join r where
    r.allergy_id = outerjoin(al.allergy_id)
join n2 where
    n2.nomenclature_id = outerjoin(r.reaction_nom_id)
order by
    onset desc
    , al.allergy_instance_id
    , 0
head report
  allergy_cnt = 0
head al.allergy_instance_id
  reac_string = " "
  allergy_cnt = allergy_cnt +1
detail
  if(mod(allergy_cnt,10) = 1)
    stat = alterlist(allergy->list, allergy_cnt +9)
  endif
   allergy->list[allergy_cnt].reaction = trim(reaction_desc)
  allergy->list[allergy_cnt].description = allergy_desc
  allergy->list[allergy_cnt].onset = onset
```

```
foot al.allergy_instance_id
  allergy->allergy_cnt = allergy_cnt
foot report
  stat = alterlist(allergy->list, allergy_cnt)
with format, separator = " "

;call echorecord(allergy)

/*****
Get Problems
******/
select into "nl:"
  p.problem_instance_id
  , p.annotated_display
  , p_classification_disp = uar_get_code_display(p.classification_cd)
  , p_course_disp = uar_get_code_display(p.course_cd)
  , problem_desc = if(n.nomenclature_id = 0.0)
        p.problem_ftdesc
      else
        n.source_string
      endif
  , onset = format(p.onset_dt_tm,  "@SHORTDATE4YR")
from
  problem   p
  , nomenclature   n
plan p where p.person_id =$pid
join n where p.nomenclature_id = n.nomenclature_id
order by
  p.onset_dt_tm   desc
  , p.problem_instance_id
head report
  problem_cnt = 0
head p.problem_instance_id
  problem_cnt = problem_cnt +1
detail
  if(mod(problem_cnt,10) = 1)
    stat = alterlist(problem->list, problem_cnt +9)
  endif
  problem->list[problem_cnt].description = problem_desc
  problem->list[problem_cnt].onset = onset
foot p.problem_instance_id
  problem->problem_cnt = problem_cnt
foot report
  stat = alterlist(problem->list, problem_cnt)
with nocounter, separator=" ", format

;call echorecord(problem)

/*****
Get Vital Signs
******/
select into "nl:"
  c.encntr_id
  , c.performed_dt_tm "@SHORTDATETIME"
  , end_dt_tm = format(c.event_end_dt_tm, "@SHORTDATETIME")
  , c.event_cd
  , c_event_disp = uar_get_code_display(c.event_cd)
  , real_result = cnvtreal(c.result_val)
  , c.result_val
  , c_result_units_disp = uar_get_code_display(c.result_units_cd)
  , c_event_dk = uar_get_displaykey(c.event_cd)
  , c_result_units_dk = uar_get_displaykey(c.result_units_cd)
  , c.event_end_dt_tm
from
  clinical_event   c
  , dummyt   d1
plan c where c.encntr_id   = $eid
  and c.person_id = $pid
  and c.event_cd in(tempo_var, tempr_var, tempe_var,pulse_var,
          resp_var,sys_var,dias_var,hr_var,apihr_var)
  /*Uncomment the following line to only get vital signs from the last two days.*/
  ;and c.event_end_dt_tm >= cnvtlookbehind("2,D")
join d1 where cnvtreal(c.result_val) > 0.0

order by
  c.event_end_dt_tm   desc
  , end_dt_tm   desc
  , c_event_disp
head report
  vs_cnt = 0
head end_dt_tm
  vs_cnt = vs_cnt +1
  if(mod(vs_cnt,20) = 1)
    stat = alterlist(vs_rec->vlist, vs_cnt +19)
  endif
  vs_rec->vlist[vs_cnt].date = end_dt_tm
  vs_rec->vlist[vs_cnt].temp   = "-"
  vs_rec->vlist[vs_cnt].temp_type  = "-"
  vs_rec->vlist[vs_cnt].heart_rate  = "-"
  vs_rec->vlist[vs_cnt].resp_rate = "-"
  vs_rec->vlist[vs_cnt].sys_bp  = "-"
  vs_rec->vlist[vs_cnt].dia_bp   = "-"
detail
  case (c.event_cd)
    of tempo_var:
      vs_rec->vlist[vs_cnt].temp_type = "oral"
      vs_rec->vlist[vs_cnt].temp = c.result_val
    of tempr_var:
      vs_rec->vlist[vs_cnt].temp_type = "rect"
      vs_rec->vlist[vs_cnt].temp = c.result_val
    of tempe_var:
      vs_rec->vlist[vs_cnt].temp_type = "ear"
      vs_rec->vlist[vs_cnt].temp = c.result_val
    of pulse_var:
      vs_rec->vlist[vs_cnt].heart_rate = c.result_val
    of resp_var:
      vs_rec->vlist[vs_cnt].resp_rate = c.result_val
    of sys_var:
      vs_rec->vlist[vs_cnt].sys_bp = c.result_val
    of dias_var:
        vs_rec->vlist[vs_cnt].dia_bp = c.result_val
      of hr_var:
        vs_rec->vlist[vs_cnt].heart_rate = c.result_val
      of apihr_var:
        vs_rec->vlist[vs_cnt].heart_rate = c.result_val
  endcase
foot report
  stat = alterlist(vs_rec->vlist, vs_cnt)
with nocounter, separator=" ", format

;call echorecord(vs_rec)
```

2. Next use DVDev (DiscernVisualDeveloper.exe) to create a new layout program named something like: *1_initials*_ALLERGY_VS_PROB_PS. Note there is a 30 character limit on the length of the layout program name. When prompted choose PostScript as the Output Type.

During the building and testing process it may be easier to execute the layout program as a prompt program from within DVDev (DiscernVisualDeveloper.exe) than it would be to execute it as an MPage from *PowerChart*. Also, having prompts allows passing context variables to the program when it is executed as an MPage.

3. Use Prompt Builder to create the following example:

- A prompt for an output device named OutDev
- A prompt for a Person ID named PID
- A prompt for an Encounter ID named EID

Verify the Prompt_Type for PID and EID is expression. Next, add the include file created above to the layout program.

The include file creates the problem, allergy, and vital signs record structures. It also executes queries to get the problems, allergies, and vital signs and store them in the record structure. In the CCL_MP_GET_ALLERGY_VS_PROB example, a final query was used to display the data from the record structures. We want to duplicate that final query in the layout program. To do that, use the Query Builder to create the following query. It is recommended that you name the query **Display_Data**. Verify the Associate Layout option is selected when you add the query. Also, if you copy the following query to the clipboard you can use the Insert Query From Clipboard option to import the query.

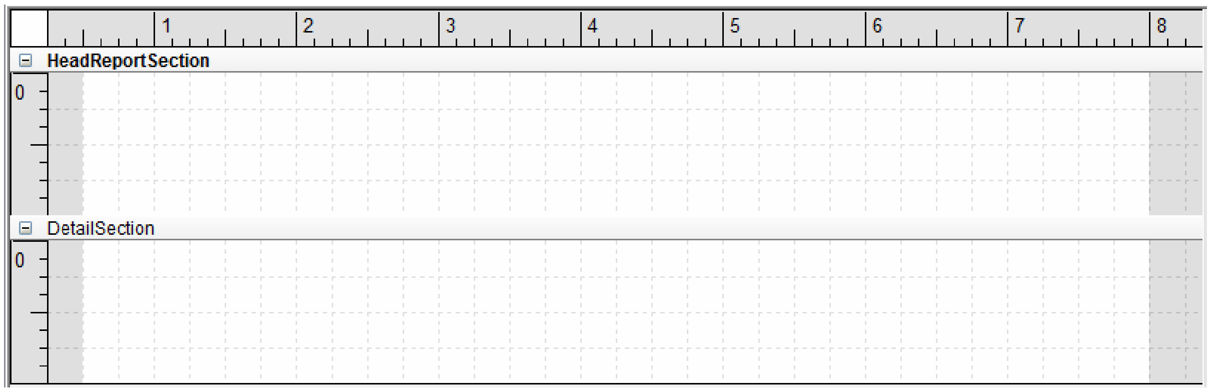
```
select
  vDATE = substring(1, 30, vs_rec->vlist[d1.seq].date)
  , vTEMP = concat(trim(vs_rec->vlist[d1.seq].temp), " ",
      trim(vs_rec->vlist[d1.seq].temp_type))
  , vHEART_RATE = vs_rec->vlist[d1.seq].heart_rate
  , vRESP_RATE = vs_rec->vlist[d1.seq].resp_rate
  , vBP = if(vs_rec->vlist[d1.seq].sys_bp != "-")
      concat(trim(vs_rec->vlist[d1.seq].sys_bp), "/", trim(vs_rec->vlist[d1.seq].dia_bp))
    else  "-"
    endif
from
  (dummyt   d1  with seq = value(size(vs_rec->vlist, 5)))

plan d1
```

```
order by
  vDATE
with nocounter, separator=" ", format
```

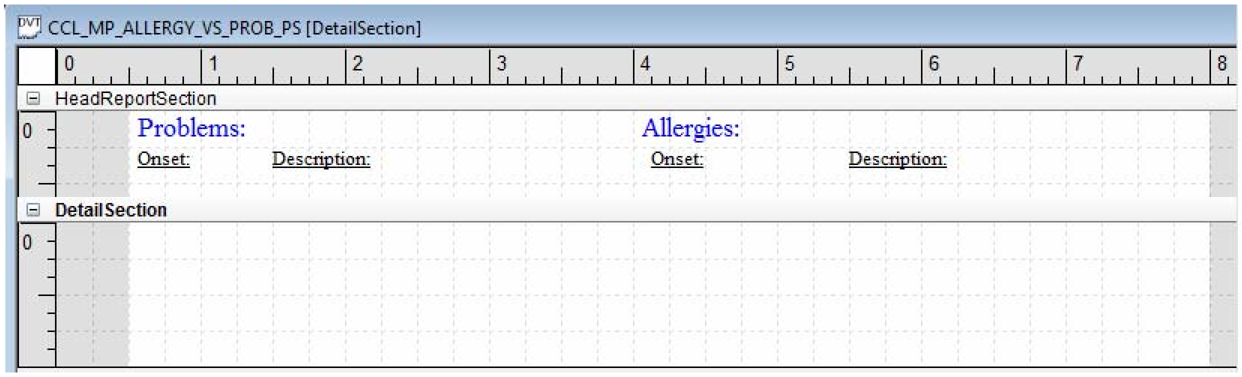
The CCL_MP_GET_ALLERGY_VS_PROB example used this same query to get the vital signs information out of the record structure and display it in the Detail report writer section. We will use a layout section to accomplish the same thing in the layout program we are now creating. Note that it is not required to first load data into a record structure and then select from that record structure to display the output as an MPage. We are simply using the record structures to gather data from multiple selects to display in a final query, DVDev (DiscernVisualDeveloper.exe) automatically creates a prompt for an output device. The output of the query associated with the layout is directed to that output device, so that any layout program can be executed as an MPage.

Now that we have the data gathered and a query associated with the layout, we can begin using layout sections to display the output. When the layout program was created, a single layout section named DetailSection was added to the layout. This layout section is associated with the Detail report writer section.

4. Using the Select/Modify Sections option, add a Head Report Reportwriter section. Layout Builder prompts to see if you want to create a layout section and associate it with the report writer section. Create and associate the layout section. Your layout should look similar to the following example:

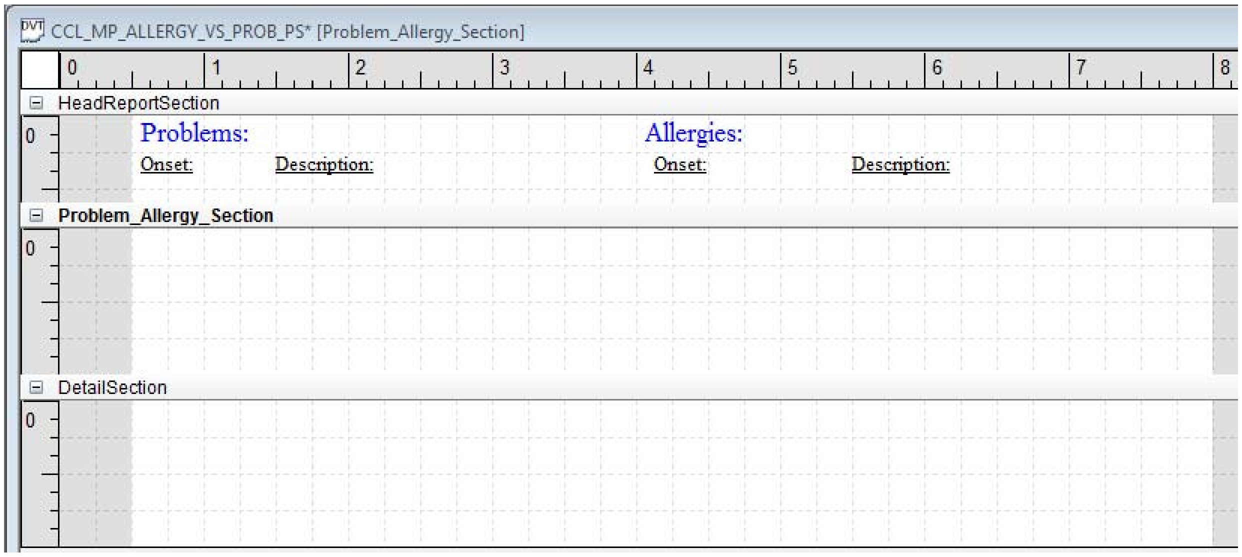


5. Add some labels to the HeadReportSection to identify the Onset and Description of both allergies and Problems similar to the following examples:

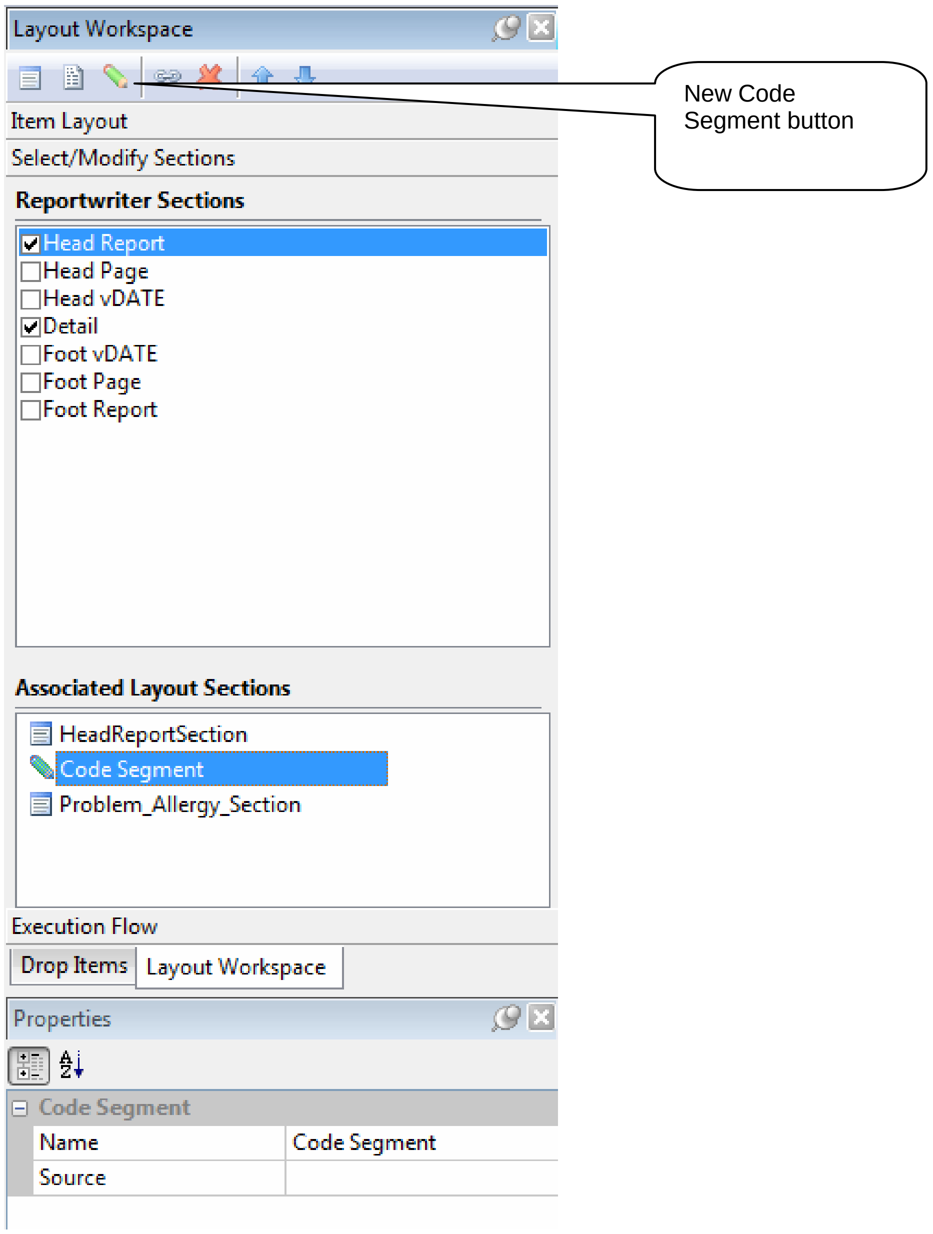


6. Next, we want to display the actual problems and allergies. To do that, we need to create a loop that will traverse the allergy and problem record structures, and inside that loop we need a layout section that will display the values from the record structures.

7. Add a layout section to the Head Report Reportwriter section. Name the section **Problem_Allergy_Section**. Your layout should look similar to the following example:



8. Next, we need to add a code segment before the Problem_Allergy_Section. This code segment will contain the code to begin the loop to traverse the record structures. In the 2007.18 release of Layout Builder, the code segment can be added using the New Code Segment button in the Layout Workspace as shown below.



Reportwriter Sections:

- [✓] Head Report
- [ ] Head Page
- [ ] Head vDATE
- [✓] Detail
- [ ] Foot vDATE
- [ ] Foot Page
- [ ] Foot Report

Associated Layout Sections:

- HeadReportSection
- Code Segment
- Problem_Allergy_Section

Properties — Code Segment

Name	Code Segment
Source	

Note: In releases previous to 2007.18, the code segment can be added using Add/Order button on the Select/Modify Sections dialog.

9. Use the following as the source for the code segment that was added above:

```
/*
Make the allergy->list and the problem->list the same size.
This allows a single for loop to be used to display all the allergies and problems
*/

if(problem->problem_cnt > allergy->allergy_cnt ) ;more problems than allergies
   lcnt = problem->problem_cnt
   stat = alterlist(allergy->list ,lcnt) ;pad the allergy list
elseif( problem->problem_cnt < allergy->allergy_cnt) ;more allergies than problems
```

```
  lcnt = allergy->allergy_cnt
  stat = alterlist(problem->list, lcnt) ;pad the problem list
else  ;equal number of allergies and problems
  lcnt = allergy->allergy_cnt
endif

;start the for loop
for(lcnt2 = 1 to lcnt)
  if(allergy->list[lcnt2].description > " ")
     reac_string = concat(allergy->list[lcnt2].description, " / ", allergy->list[lcnt2].reaction)
  else
     reac_string = " "
  endif
```

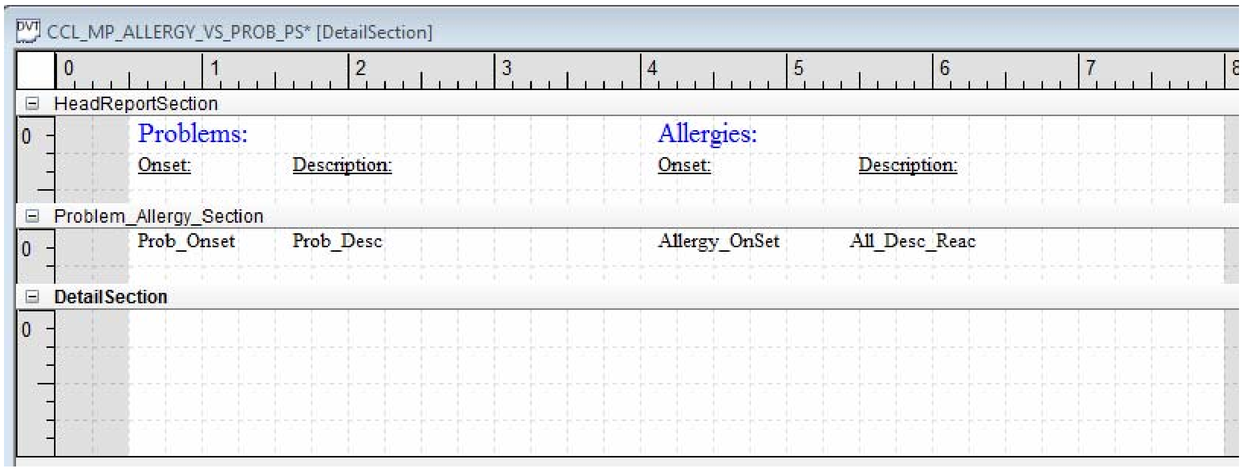
10. Add a code segment after the Problem_Allergy_Section. This code segment will contain the code to end the loop that traverses the record structures. Use the following as the source for the code segment that was added above:

```
endfor
```

11. You can now add four text items to the Problem_Allergy_Section to display the problem onset and description and the allergy onset and description. Use the following as the source for these four text items:

```
problem->list [lcnt2].onset
problem->list [lcnt2].description
allergy->list [lcnt2].onset
reac_string
```

Note that the source code you placed in the code segment to start the loop also sets the reac_string variable. Your layout should look something like the following example:



Note: In the above example, "Prob_Onset", "Prob_Desc", "Allergy_OnSet", and "All_Desc_Reac" are the names that were assigned to the four text items added to the Problem_Allergy_Section. The names displayed in your layout will most likely be different. It is a good idea to save and execute your layout at this point, and correct any syntax errors before continuing.

You are now ready to use the DetailSection to display the Vital Signs information. However, adding some labels and column headings will help ensure the output is meaningful. Since the labels and column headings are only displayed once, using another layout section in the Head Report Reportwriter section is best way to create them.

12. Add another layout section to the Head Report Reportwriter section. Name the new section **VS_HeadSection**, then add the following labels to this section:

Vital Signs: (Use a larger font and different color to make this label stand out.)

Date:

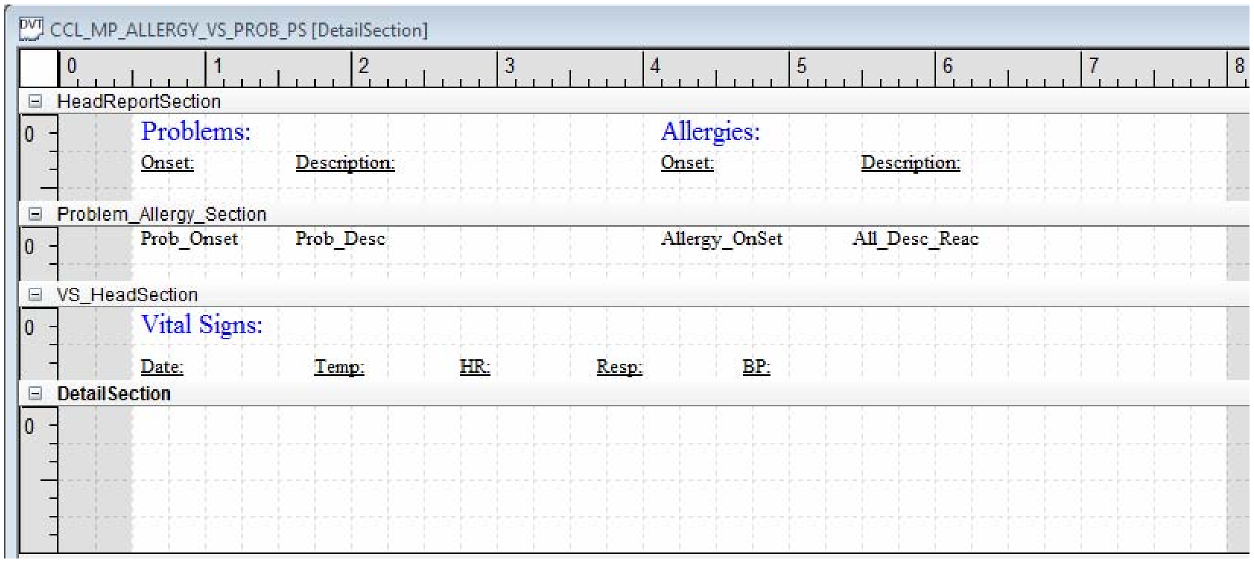
Temp:

HR:

Resp:

BP:

Your layout should look something like the following example:



13. You can now use the drop items functionality or create text items to display the following select expressions:

vDate

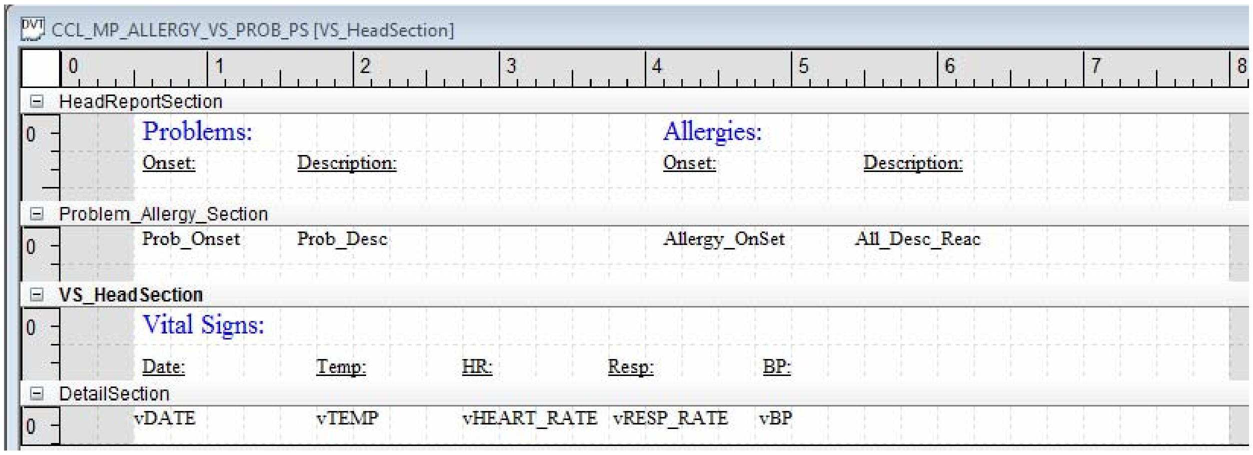
vTemp

vHeart_Rate

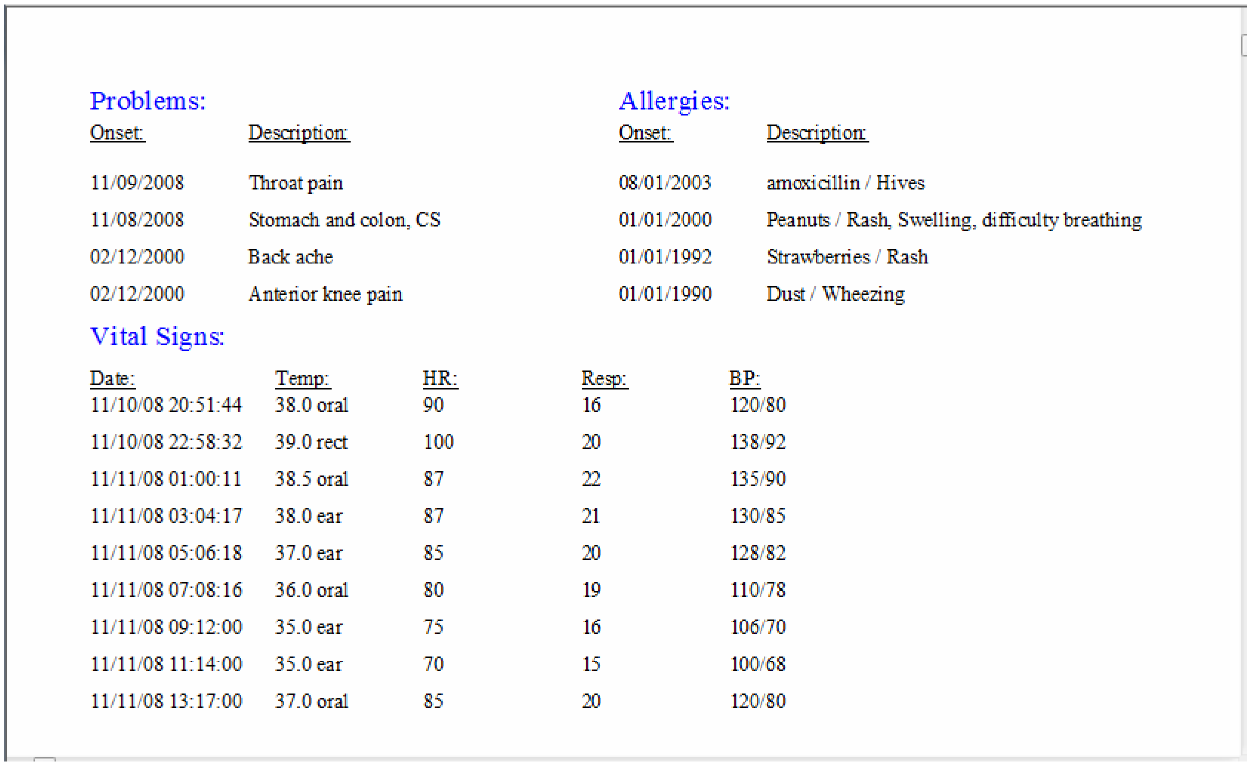
vResp_Rate

vBP

These select expressions are created by the query that was named Display_Data and associated with the layout earlier in this section. Your layout should look similar to the following example:



If you save and execute your layout as a prompt program, the output should look similar to the following example:



Problems:

Onset:	Description:
11/09/2008	Throat pain
11/08/2008	Stomach and colon, CS
02/12/2000	Back ache
02/12/2000	Anterior knee pain

Allergies:

Onset:	Description:
08/01/2003	amoxicillin / Hives
01/01/2000	Peanuts / Rash, Swelling, difficulty breathing
01/01/1992	Strawberries / Rash
01/01/1990	Dust / Wheezing

Vital Signs:

Date:	Temp:	HR:	Resp:	BP:
11/10/08 20:51:44	38.0 oral	90	16	120/80
11/10/08 22:58:32	39.0 rect	100	20	138/92
11/11/08 01:00:11	38.5 oral	87	22	135/90
11/11/08 03:04:17	38.0 ear	87	21	130/85
11/11/08 05:06:18	37.0 ear	85	20	128/82
11/11/08 07:08:16	36.0 oral	80	19	110/78
11/11/08 09:12:00	35.0 ear	75	16	106/70
11/11/08 11:14:00	35.0 ear	70	15	100/68
11/11/08 13:17:00	37.0 oral	85	20	120/80

The above output is looks better than the plain ASCII output created in the CCL_MP_GET_ALLERGY_VS_PROB example.

A common request is for the output to use alternating background colors on rows of data, so that every other row of information is displayed on a different background. This often makes the output easier to read. In layout builder, alternating row colors can be accomplished using conditional formatting.

14. Click in the DetailSection and use Ctrl+A to select all items in the section.

15. On the properties page, select the Conditional Formatting property and then click the ellipsis ... button to open the Conditional Formatting dialog.

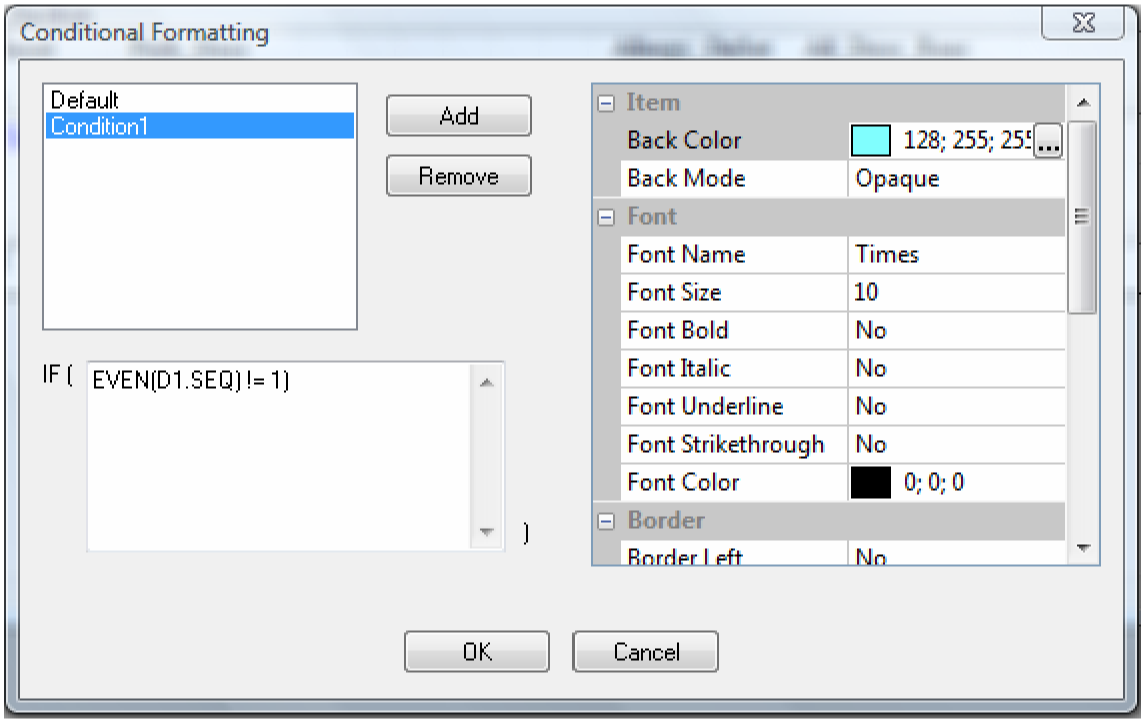
16. Click the Add button to add a condition and then select the newly added condition to activate the IF() control.

17. Enter **EVEN(D1.SEQ) != 1** in the IF() control.

18. Change the Back Mode property from Transparent to Opaque.

19. Select a light color for the Back Color property.

Your Conditional Formatting dialog should look something like the following example:



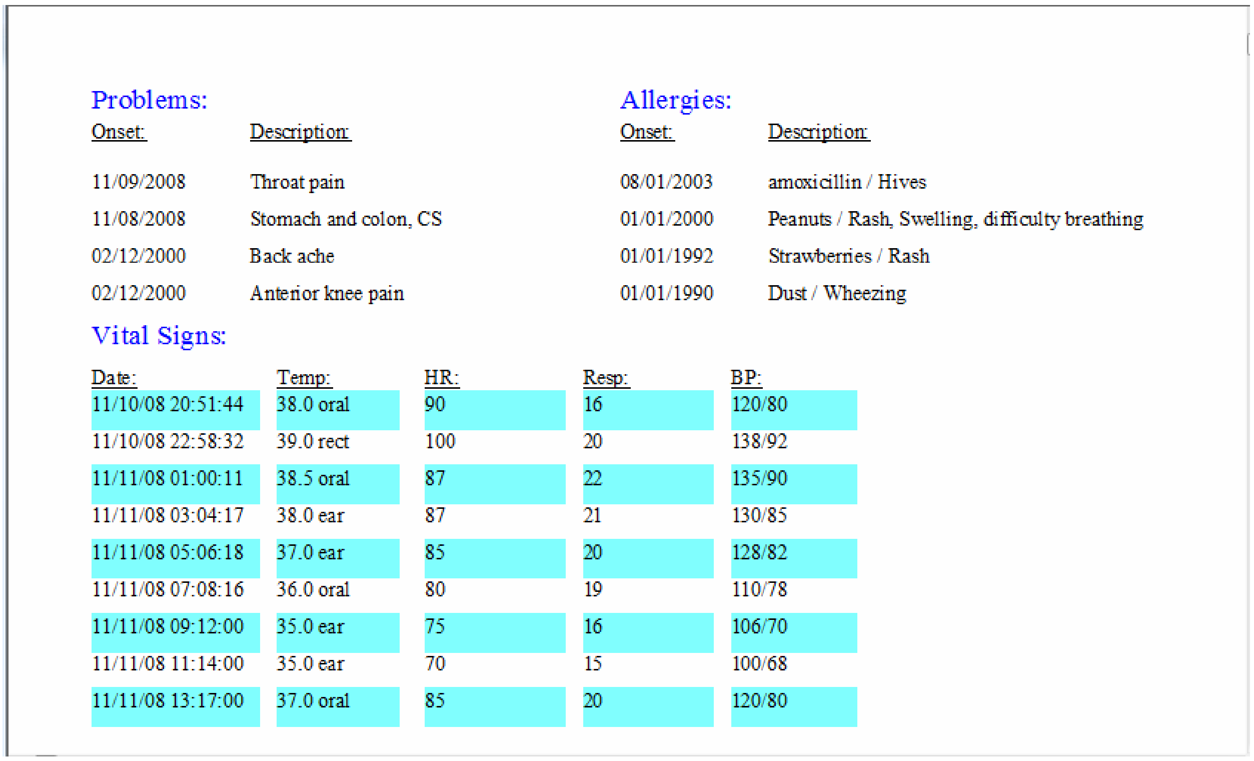
IF (EVEN(D1.SEQ) != 1))

Item	
Back Color	🟦 128; 255; 25
Back Mode	Opaque
Font	
Font Name	Times
Font Size	10
Font Bold	No
Font Italic	No
Font Underline	No
Font Strikethrough	No
Font Color	⬛ 0; 0; 0
Border	
Border Left	No

Note that the query associated with the layout uses the Dummyt table to traverse the record structure containing the vital signs information. That query uses D1 as the alias for the Dummyt table and will set the seq value to one and increment it by one for each position in the record structure list, so the condition EVEN(D1.SEQ) !=1 will be true for every other row returned by the query. Thus, every other row of the output will be displayed using the background color you have selected.

If the query was not setting the D1.SEQ field, then some other method would be needed to test for the condition. For example you could use a code segment to increment a counter variable in the Detail report writer section, and then add a condition using that counter variable.

The output from executing the layout using the conditional formatting set above may look something like the following example:



Problems:

Onset:	Description:
11/09/2008	Throat pain
11/08/2008	Stomach and colon, CS
02/12/2000	Back ache
02/12/2000	Anterior knee pain

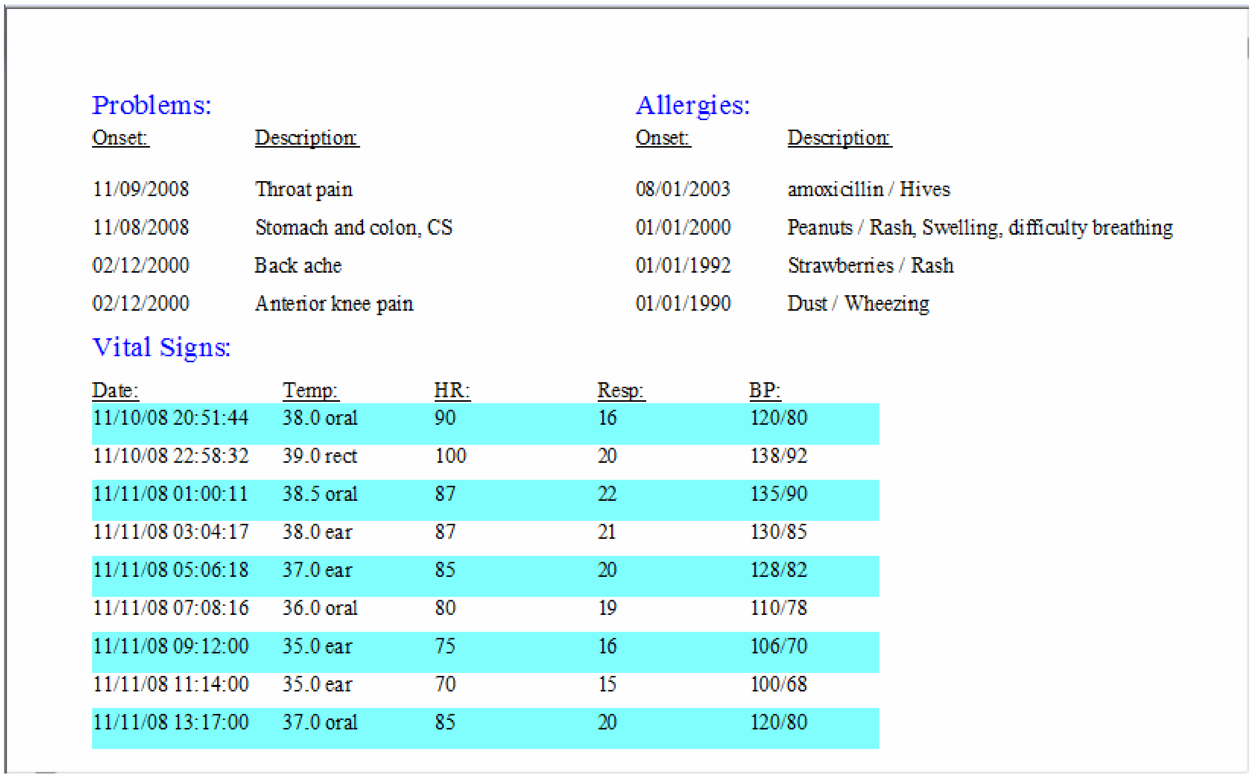
Allergies:

Onset:	Description:
08/01/2003	amoxicillin / Hives
01/01/2000	Peanuts / Rash, Swelling, difficulty breathing
01/01/1992	Strawberries / Rash
01/01/1990	Dust / Wheezing

Vital Signs:

Date:	Temp:	HR:	Resp:	BP:
11/10/08 20:51:44	38.0 oral	90	16	120/80
11/10/08 22:58:32	39.0 rect	100	20	138/92
11/11/08 01:00:11	38.5 oral	87	22	135/90
11/11/08 03:04:17	38.0 ear	87	21	130/85
11/11/08 05:06:18	37.0 ear	85	20	128/82
11/11/08 07:08:16	36.0 oral	80	19	110/78
11/11/08 09:12:00	35.0 ear	75	16	106/70
11/11/08 11:14:00	35.0 ear	70	15	100/68
11/11/08 13:17:00	37.0 oral	85	20	120/80

Notice that the conditional formatting is applied to each of the text items in the layout section. In the above example there is white space between the text items. If you want the entire row to have the alternating background color with no white space between, you can simply expand the width of the items to remove the space that is between them. For example, drag the right side of the date item over so that it is touching the left side of the temperature item. The following example shows the output after resizing the items so that the right side of one item touches the left side of the next item.



Problems:

Onset:	Description:
11/09/2008	Throat pain
11/08/2008	Stomach and colon, CS
02/12/2000	Back ache
02/12/2000	Anterior knee pain

Allergies:

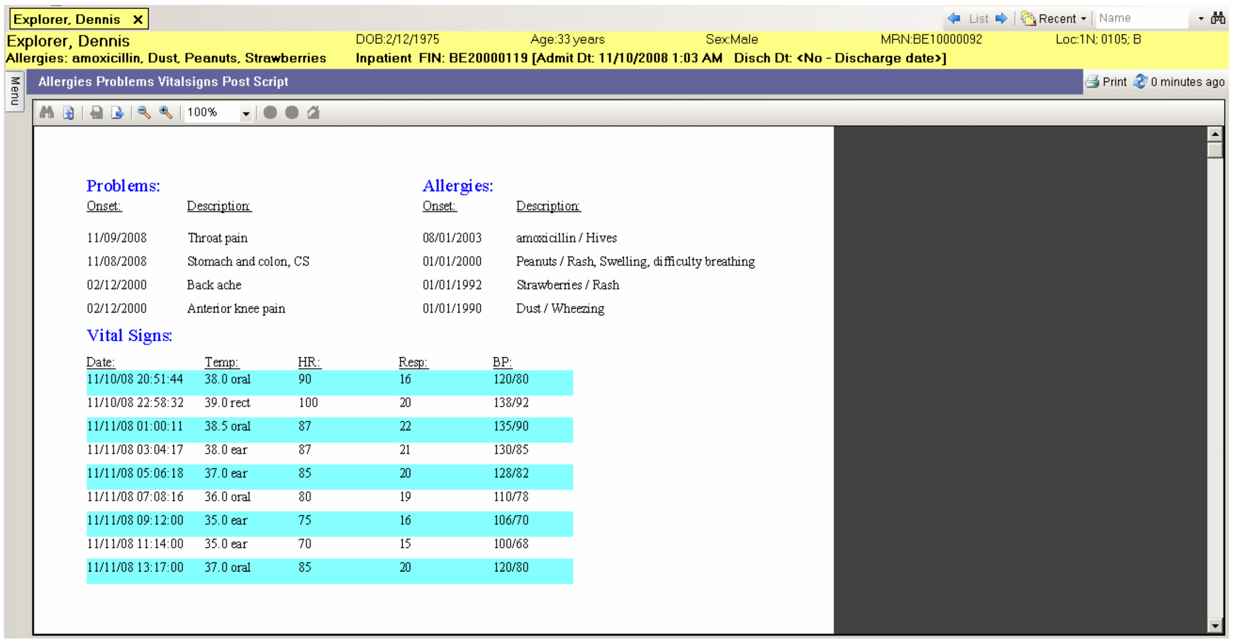
Onset:	Description:
08/01/2003	amoxicillin / Hives
01/01/2000	Peanuts / Rash, Swelling, difficulty breathing
01/01/1992	Strawberries / Rash
01/01/1990	Dust / Wheezing

Vital Signs:

Date:	Temp:	HR:	Resp:	BP:
11/10/08 20:51:44	38.0 oral	90	16	120/80
11/10/08 22:58:32	39.0 rect	100	20	138/92
11/11/08 01:00:11	38.5 oral	87	22	135/90
11/11/08 03:04:17	38.0 ear	87	21	130/85
11/11/08 05:06:18	37.0 ear	85	20	128/82
11/11/08 07:08:16	36.0 oral	80	19	110/78
11/11/08 09:12:00	35.0 ear	75	16	106/70
11/11/08 11:14:00	35.0 ear	70	15	100/68
11/11/08 13:17:00	37.0 oral	85	20	120/80

The layout program created in the above example can be set up to execute as an MPage from *PowerChart* by using Preference Maintenance (PrefMaint.exe) and entering ***1_initials*****_ALLERGY_VS_PROB_PS** as the REPORT_NAME preference and **"MINE", $PAT_PersonID$, $VIS_EncntrID$** as the REPORT_PARAM preference.

The output from executing the above program as an MPage should something like the following example:



Explorer, Dennis DOB:2/12/1975 Age:33 years Sex:Male MRN:BE10000092 Loc:1N; 0105; B
Allergies: amoxicillin, Dust, Peanuts, Strawberries Inpatient FIN: BE20000119 [Admit Dt: 11/10/2008 1:03 AM Disch Dt: <No - Discharge date>]

Allergies Problems Vitalsigns Post Script

Problems:

Onset:	Description:
11/09/2008	Throat pain
11/08/2008	Stomach and colon, CS
02/12/2000	Back ache
02/12/2000	Anterior knee pain

Allergies:

Onset:	Description:
08/01/2003	amoxicillin / Hives
01/01/2000	Peanuts / Rash, Swelling, difficulty breathing
01/01/1992	Strawberries / Rash
01/01/1990	Dust / Wheezing

Vital Signs:

Date:	Temp:	HR:	Resp:	BP:
11/10/08 20:51:44	38.0 oral	90	16	120/80
11/10/08 22:58:32	39.0 rect	100	20	138/92
11/11/08 01:00:11	38.5 oral	87	22	135/90
11/11/08 03:04:17	38.0 ear	87	21	130/85
11/11/08 05:06:18	37.0 ear	85	20	128/82
11/11/08 07:08:16	36.0 oral	80	19	110/78
11/11/08 09:12:00	35.0 ear	75	16	106/70
11/11/08 11:14:00	35.0 ear	70	15	100/68
11/11/08 13:17:00	37.0 oral	85	20	120/80

Example Patient Summary Report with HTML Output Created in Layout Builder Version 2007.18

Note: This example assumes you are using version 2007.18 or higher of Layout Builder. If you are using an earlier version of Layout Builder some of the functionality described in this section will not be available. Specifically the 2007.18 release of Layout Builder, allows code segments to be added around table row sections using the New Code Segment button in the Layout Workspace. This example uses this functionality to create a For loop to call a table row multiple times to display information stored in a record structure.

One of the biggest benefits of MPages is the ability to display HTML output. Using an HTML format allows you to create interactive output. For example, links can be added to the output that can navigate down to subsequent detail, launch another application, or open a specific website.

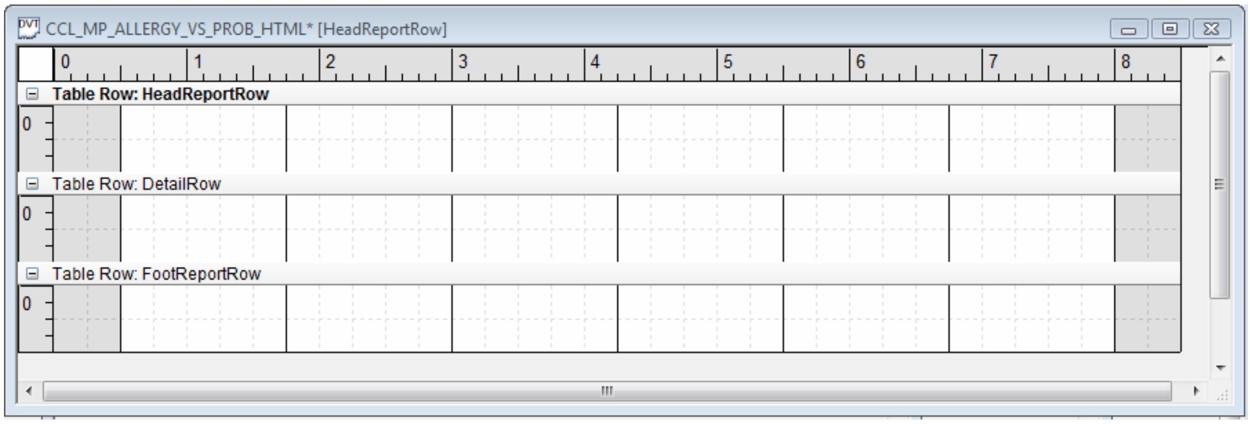
Layout Builder can be used to create HTML output. Using Layout Builder to create HTML output is very similar to creating ASCII or PostScript output. ASCII and PostScript output use fixed locations to display data elements. In an ASCII report the location of data elements is controlled by row and column references, and in a PostScript report the location of data elements is controlled by x and y coordinates. However, unlike ASCII or PostScript output, HTML output created by Layout Builder is displayed in a table and the relative size of the cells in the table are used to determine where items will be displayed.

To demonstrate some of this functionality, we will use Layout Builder to create HTML output for two programs. The first program will be a patient summary report that displays problems, allergies, and vital signs. We will add a link to the patient summary program to execute a second report to display the medications for the patient. Obviously, the medications can be added directly to the patient summary report, but we will use a second program to demonstrate the linking functionality of using HTML output.

When you completed the, Example Patient Summary Report with PostScript/PDF output Created in Layout Builder section, you created an include file named, *1_initials*_GET_ALLERGY_VS_PROB.INC and saved it in a standard location. If you do not have a *1_initials*_GET_ALLERGY_VS_PROB.INC file saved in a standard location, go back to the Example Patient Summary Report with PostScript/PDF output Created in Layout Builder section, and follow the steps to create the *1_initials*_GET_ALLERGY_VS_PROB.INC file and save it in a standard back-end location. If you are working in a multi-node AIX or HP_UX environment, the .inc file must be saved on all nodes or in a directory that is accessible from all nodes.

1. Use DVDev (DiscernVisualDeveloper.exe) to create a new layout program named something like *1_initials*_ALLERGY_VS_PROB_HTML. Note there is a 30 character limit on the length of the layout program name.

2. Using the 2007.18 version of Layout Builder, select HTML as the Output Type, select Table View as the Report Layout, set the Number of Columns to 6 and click the Finish button. Your layout should look similar to the following example:



During the building and testing process it may be easier to execute the layout program as a prompt program from within DVDev (DiscernVisualDeveloper.exe) than it would be to execute it as an MPage from *PowerChart*. Also, having prompts will allow passing context variables to the program when it is executed as an MPage. If you completed the, Example Patient Summary Report with PostScript/PDF output Created in Layout Builder, section you can use the Tools>Transfer Objects option in DVDev (DiscernVisualDeveloper.exe) to copy the prompt form from the *1_initials*_ALLERGY_VS_PROB_PS program to create the prompt form for your *1_initials*_ALLERGY_VS_PROB_HTML program. If you did not complete the, Example Patient Summary Report with PostScript/PDF output Created in Layout Builder, section, use the Prompt Builder to create the following prompts:

- A prompt for an output device named OutDev
- A prompt for a Person ID named PID
- A prompt for an Encounter ID named EID

3. Verify that the Prompt_Type for PID and EID is expression.

4. Next, add the *1_initials*_GET_ALLERGY_VS_PROB.INC include file that was created above to the layout program.

The include file will create the problem, allergy, and vital signs record structures. It also executes queries to get the problems, allergies, and vital signs, and stores them in the record structures. We want to use Layout Builder to format and display the information from those record structures.

5. To do that, use the Query Builder to create the following query (Cerner recommends naming the query **Display_Data**). Verify that the Associate Layout option is selected when you add the query. Also, if you copy the following query to the clipboard, you can use the Insert Query From Clipboard option to import the query.

```
select
  vDATE = substring(1, 30, vs_rec->vlist[d1.seq].date)
  , vTEMP = concat(trim(vs_rec->vlist[d1.seq].temp), " ",
      trim(vs_rec->vlist[d1.seq].temp_type))
  , vHEART_RATE = vs_rec->vlist[d1.seq].heart_rate
  , vRESP_RATE = vs_rec->vlist[d1.seq].resp_rate
  , vBP = if(vs_rec->vlist[d1.seq].sys_bp != "-")
      concat(trim(vs_rec->vlist[d1.seq].sys_bp), "/", trim(vs_rec->vlist[d1.seq].dia_bp))
    else  "-"
    endif
from
  (dummyt   d1  with seq = value(size(vs_rec->vlist, 5)))

plan d1

order by
  vDATE
with nocounter, separator=" ", format
```

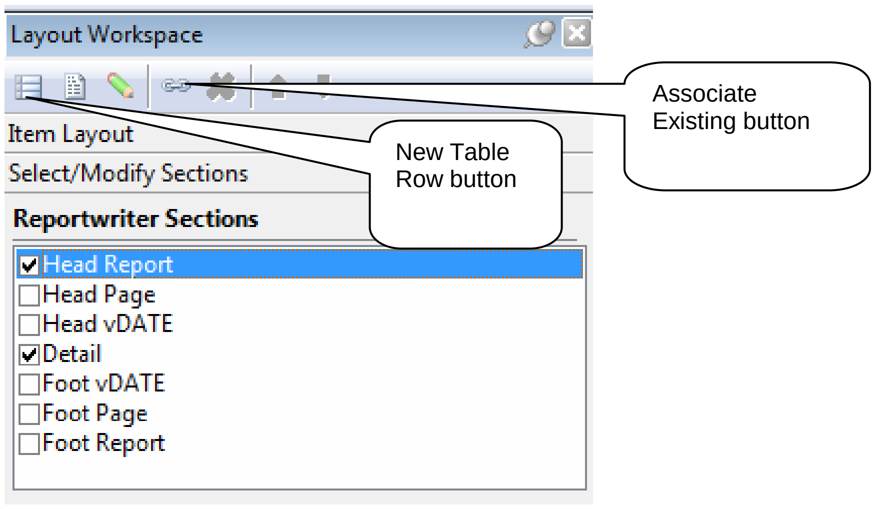
The earlier examples used this same query to get the vital signs information out of the record structure and display it in the Detail report writer section. We will now use a layout section to do the same thing in the layout program we are creating. Note that it is not required to first load data into a record structure and then select from that record structure to display the output as an MPage. We are simply using the record structures to gather data from multiple selects so that we can display it in a final query. DVDev (DiscernVisualDeveloper.exe) automatically creates a prompt for an output device. The output of the query associated with the layout is directed to that output device, so any layout program can be executed as an MPage.

Now that we have the data gathered and a query associated with the layout, we can begin using layout sections to display the output. When using the HTML table view in the 2007.18 version of layout builder, each layout section is treated as a row in the layout table and associated with a report writer section of the layout query. For this example, we need several rows or sections in the Head Report Reportwriter section that can be used for labels, column headings, and data for the problems and allergies.

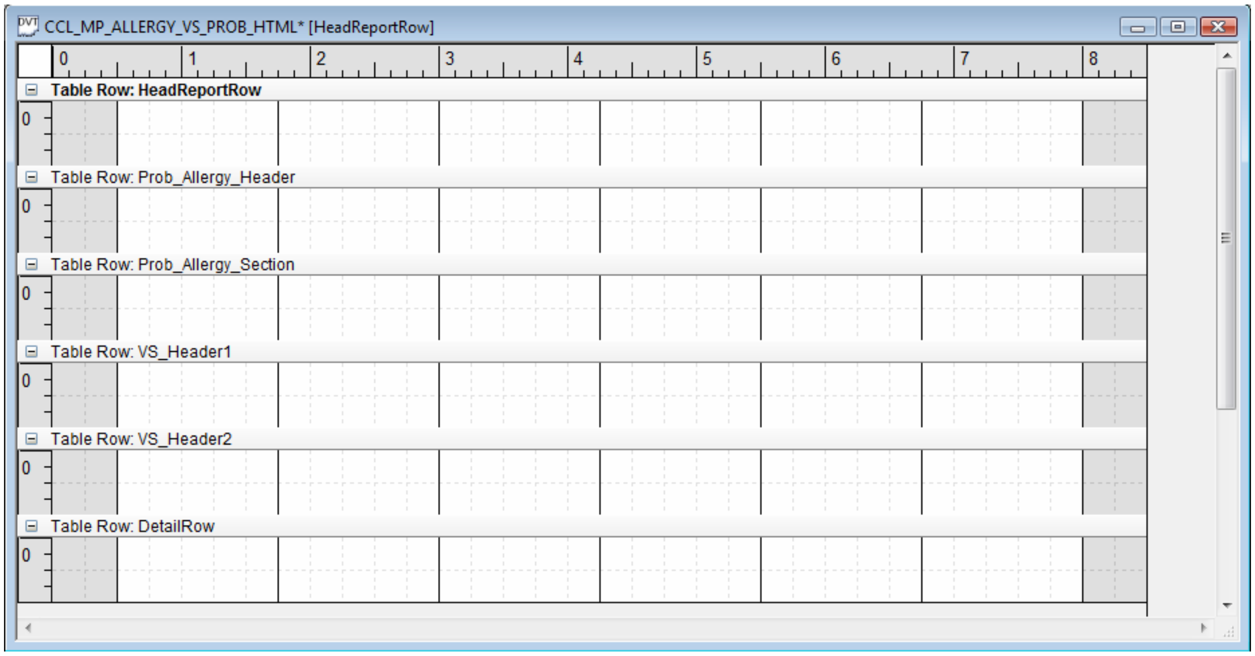
6. Add four rows to the layout table. Verify that these rows are associated with the Head Report Reportwriter section. Name the sections as follows:

- Prob_Allergy_Header
- Prob_Allergy_Section
- VS_Header1
- VS_Header2

In the 2008.18 version of Layout Builder there are a couple of ways to add the rows to the table and associate them with the Head Report Reportwriter section. First, in the Layout Workspace you can select the Head Report Reportwriter section and then click the New Table Row button. You can also use the Insert Row option on the Table menu and then use the Associate Existing button in the Layout Workspace to associate the new row with the Head Report Reportwriter section.



Your layout should look similar to the following example:


CCL_MP_ALLERGY_VS_PROB_HTML* [HeadReportRow]
Table Row: HeadReportRow
Table Row: Prob_Allergy_Header
Table Row: Prob_Allergy_Section
Table Row: VS_Header1
Table Row: VS_Header2
Table Row: DetailRow

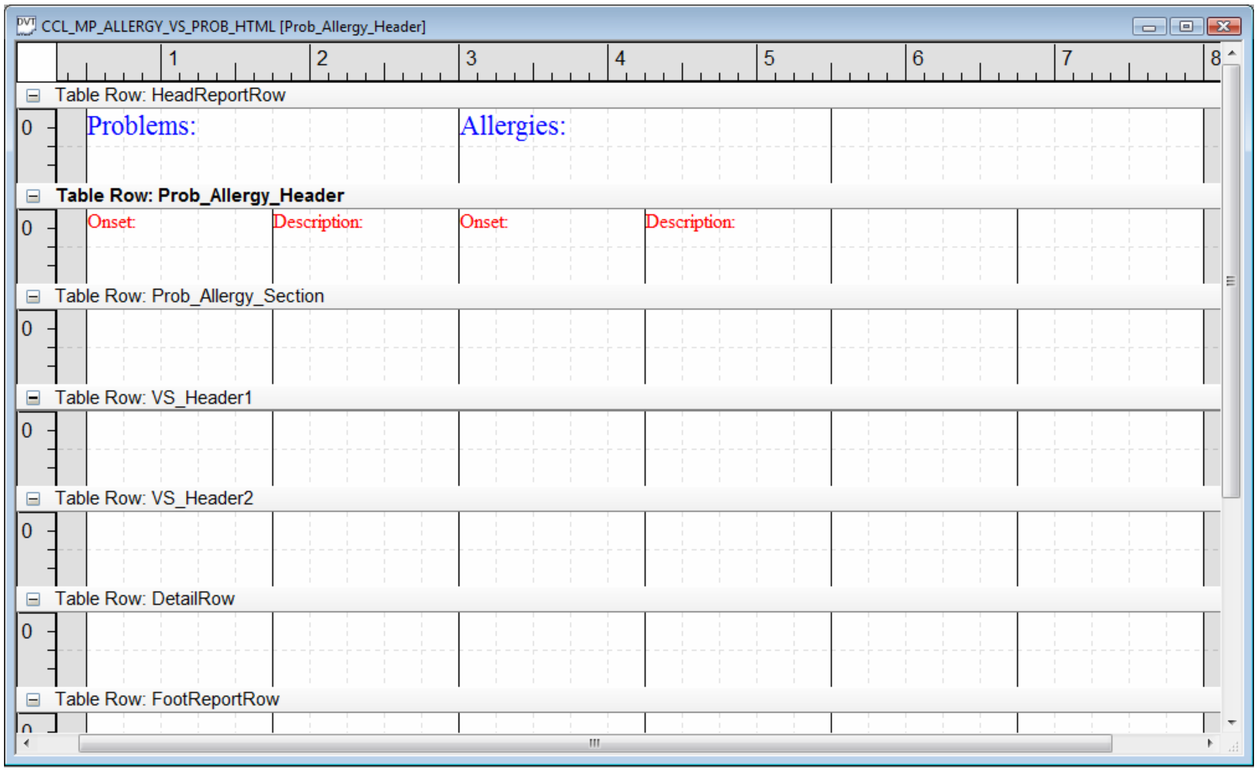
Currently all the rows in the table have six columns that are evenly spaced across the page. At the top of the layout we want to display a couple of labels for Problems and Allergies that are wider than the default column width. To create a couple of cells that are wider than the default, we need to remove some of the existing cells.

7. While pressing the Ctrl key, click in the first, third, and fifth cells in the HeadReportRow section, then choose Remove Cell from the Table menu. After removing these cells, you should have only three cells in the HeadReportRow section and each of those cells should be double the width of the default columns.

8. Enter **"Problems:"** and **"Allergies:"** as the Source property for the first and second cells in the HeadReportRow section.

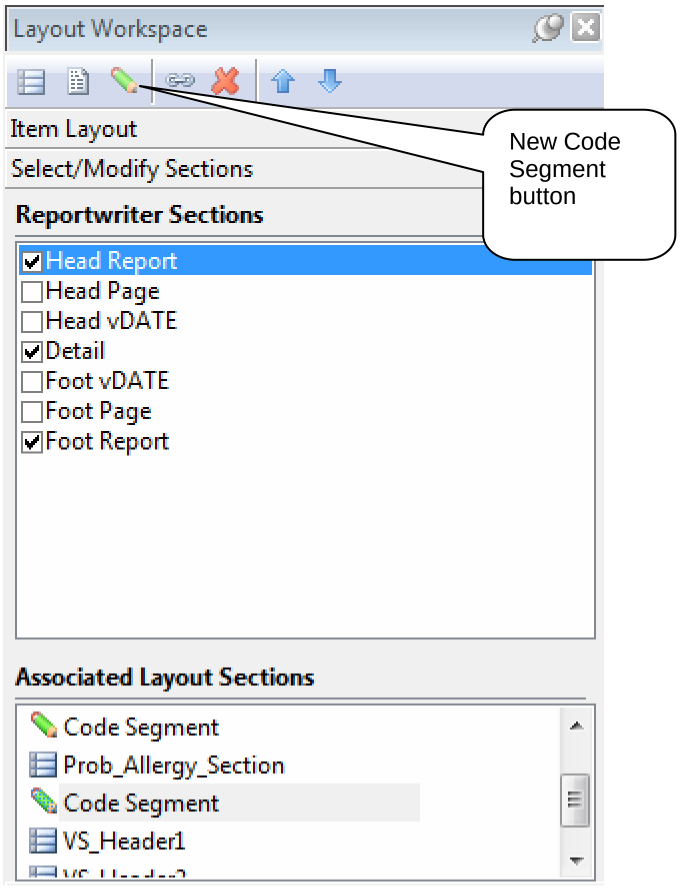
9. Enter **"Onset:"** and **"Description:"** as the source property for the first and second cells under Problems: and then again for the third and fourth cells under Allergy:. You can change the font color to make these labels stand out.

Your layout should look something like the following example:



Next, we want to display the actual problems and allergies. To do so, we need to create a loop that will traverse the allergy and problem record structures. The Prob_Allergy_Section must be embedded inside the loop and used to display the values from the record structures. Beginning with the 2007.18 release of Layout Builder, code segments can be added around table row sections using the New Code Segment button in the Layout Workspace. The code segments can be used to create the loop.

10. Using the New Code Segment button in the Layout Workspace, add two new code segments. Use the up and down arrows in the Layout Workspace to move one of the new code segments above the Prob_Allergy_Section and the other segment below it. Your Layout Workspace should look similar to the following example:



Reportwriter Sections: ☑ Head Report, ☐ Head Page, ☐ Head vDATE, ☑ Detail, ☐ Foot vDATE, ☐ Foot Page, ☑ Foot Report

Associated Layout Sections: Code Segment, Prob_Allergy_Section, Code Segment, VS_Header1

11. Add the following as the source for the code segment that was added above the Prob_Allergy_Section:

```
/*
Make the allergy->list and the problem->list the same size.
This will allow a single for loop to be used to display all the allergies and problems
*/

if(problem->problem_cnt > allergy->allergy_cnt ) ;more problems than allergies
   lcnt = problem->problem_cnt
   stat = alterlist(allergy->list ,lcnt) ;pad the allergy list
elseif( problem->problem_cnt < allergy->allergy_cnt) ;more allergies than problems
   lcnt = allergy->allergy_cnt
   stat = alterlist(problem->list, lcnt) ;pad the problem list
else  ;equal number of allergies and problems
   lcnt = allergy->allergy_cnt
endif

;start the for loop
for(lcnt2 = 1 to lcnt)
   if(allergy->list[lcnt2].description > " ")
       reac_string = concat(allergy->list[lcnt2].description, " / ", allergy->list[lcnt2].reaction)
   else
       reac_string = " "
   endif
```

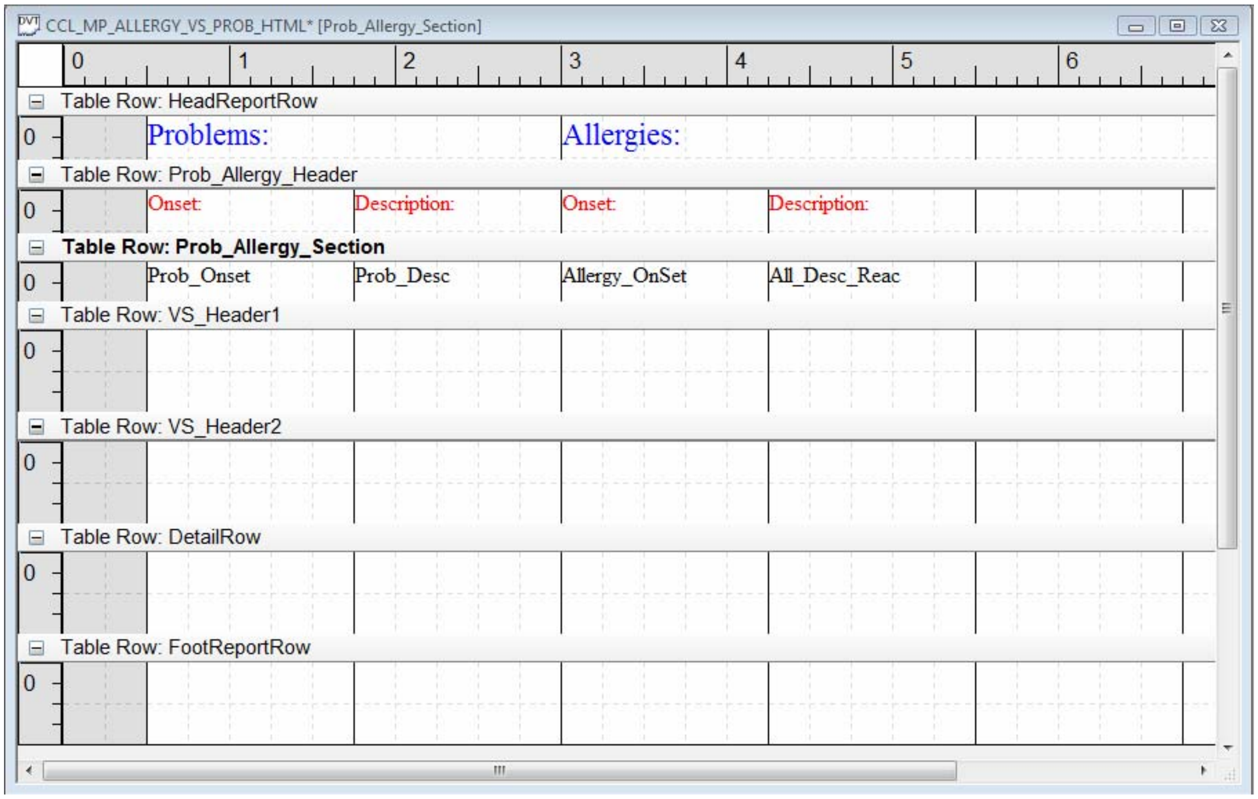
12. Use the following as the source for the code segment that was added below the Prob_Allergy_Section:

```
endfor
```

13. You can now use the cells in the Prob_Allergy_Section to display the problem and allergy onsets and descriptions that were loaded into the Problem and Allergy record structures by the code in the include file. Use the following as the source for the four cells in the Prob_Allergy_Section:

```
problem->list [lcnt2].onset
problem->list [lcnt2].description
allergy->list [lcnt2].onset
reac_string
```

Note that the source code you placed in the code segment to start the for loops sets the reac_string variable. Your layout should look something like the following example:



Note that in the above example, *Prob_Onset*, *Prob_Desc*, *Allergy_OnSet*, and *All_Desc_Reac* are the names that were assigned to the cells in the Prob_Allergy_Section by the programmer. The names displayed in your layout will most likely be different. At this point, it is a good idea to save and execute your layout. The output of your layout should look similar to the following example:

Problems:		Allergies:			
Onset:	Description:	Onset:	Description:		
11/09/2008	Throat pain	08/01/2003	amoxicillin / Hives		
11/08/2008	Stomach and colon, CS	01/01/2000	Peanuts / Rash, Swelling, difficulty breathing		
02/12/2000	Back ache	01/01/1992	Strawberries / Rash		
02/12/2000	Anterior knee pain	01/01/1990	Dust / Wheezing		

By default Layout Builder displays lines around each cell in the table. The lines around the unused cells to the right of the allergy description can give the output a strange look. There are two ways to remedy this issue. First, you can remove the lines by setting the Pen Size table property to 0.0, which eliminates all of the lines around all of the cells. A second option is to remove the empty cells to the right of the allergy description by right-clicking in the cells and selecting Remove Cell from the context menu. After the cells to the right of the allergy description were removed with this method, the layout would look similar to the following example:

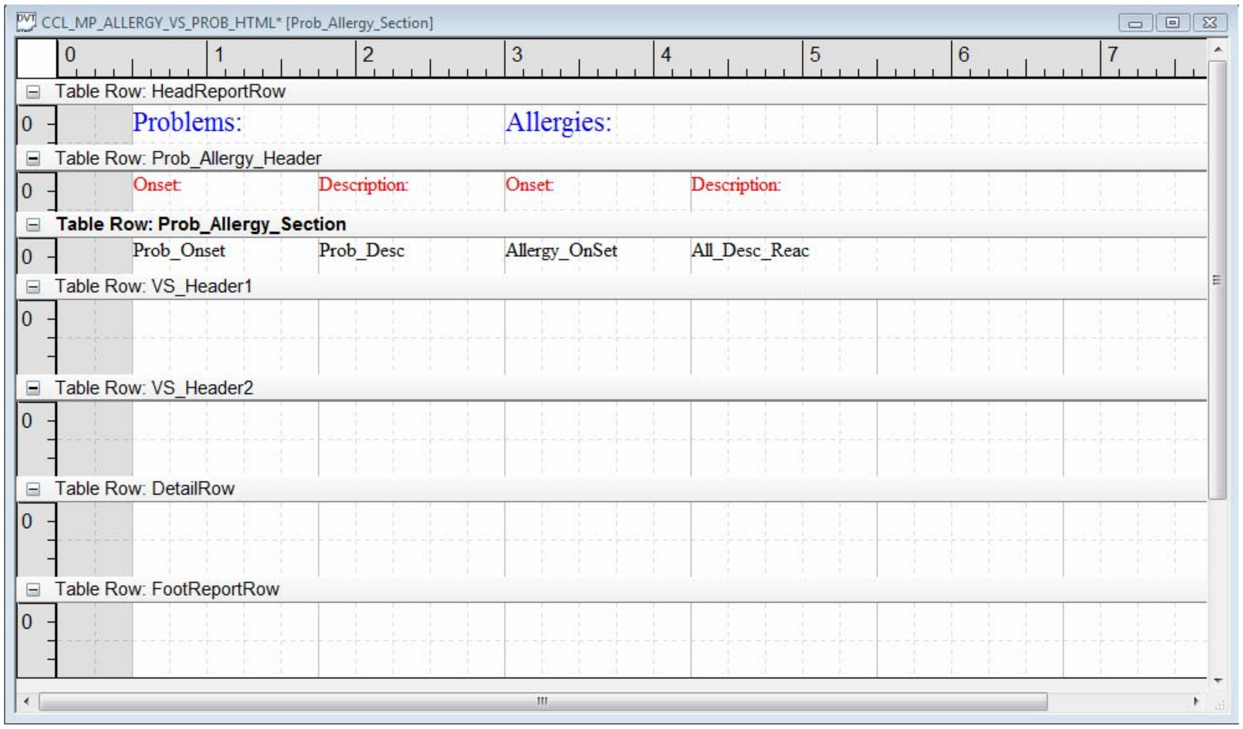
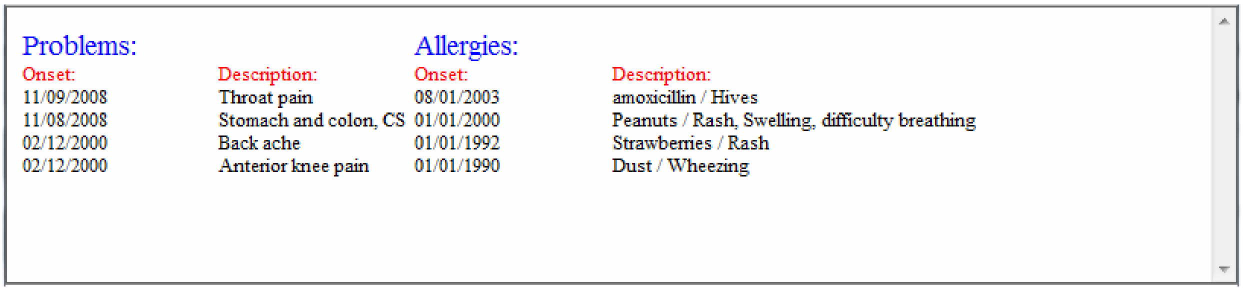

CCL_MP_ALLERGY_VS_PROB_HTML* [Prob_Allergy_Section]

Table Row: HeadReportRow
Problems: Allergies:
Table Row: Prob_Allergy_Header
Onset: Description: Onset: Description:
Table Row: Prob_Allergy_Section
Prob_Onset Prob_Desc Allergy_OnSet All_Desc_Reac
Table Row: VS_Header1
Table Row: VS_Header2
Table Row: DetailRow
Table Row: FootReportRow

Removing the unused cells also allows the allergy description cell to consume the rest of the layout area. The following example shows the output after the cells are removed:

Problems:		Allergies:		
Onset:	Description:	Onset:	Description:	
11/09/2008	Throat pain	08/01/2003	amoxicillin / Hives	
11/08/2008	Stomach and colon, CS	01/01/2000	Peanuts / Rash, Swelling, difficulty breathing	
02/12/2000	Back ache	01/01/1992	Strawberries / Rash	
02/12/2000	Anterior knee pain	01/01/1990	Dust / Wheezing	

The following example shows the output after removing the cells to the right of the description and setting the table Pen Size property to 0.0



Problems:

Onset:	Description:
11/09/2008	Throat pain
11/08/2008	Stomach and colon, CS
02/12/2000	Back ache
02/12/2000	Anterior knee pain

Allergies:

Onset:	Description:
08/01/2003	amoxicillin / Hives
01/01/2000	Peanuts / Rash, Swelling, difficulty breathing
01/01/1992	Strawberries / Rash
01/01/1990	Dust / Wheezing

Conditional formatting can be used to display every other row of the output on a different color background.

14. Click in the Prob_Allergy_Section and use Ctrl+A to select all items in the section.

15. On the Properties page, select the Conditional Formatting property and click the ellipsis ... button to open the Conditional Formatting dialog.

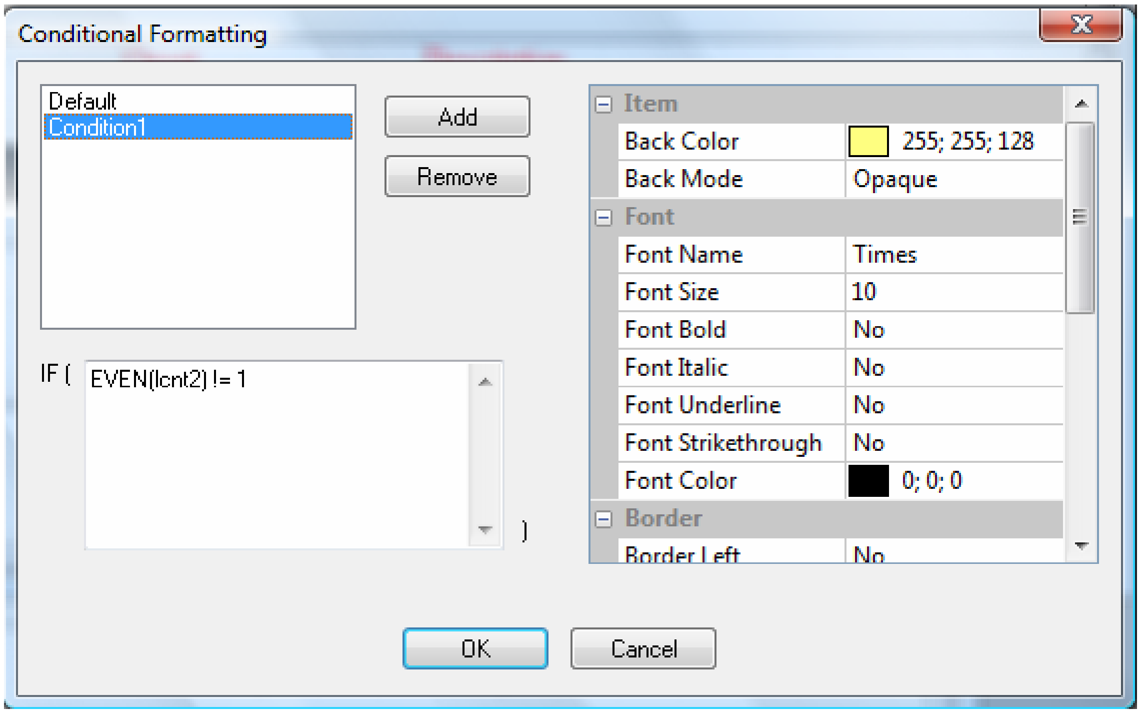
16. Click the Add button to add a condition and then select the newly added condition to activate the IF() control.

17. Enter **EVEN(lcnt2) != 1** in the IF() control.

18. Change the Back Mode property from Transparent to Opaque.

19. Select a light color for the Back Color property.

Your Conditional Formatting dialog should look something like the following example:

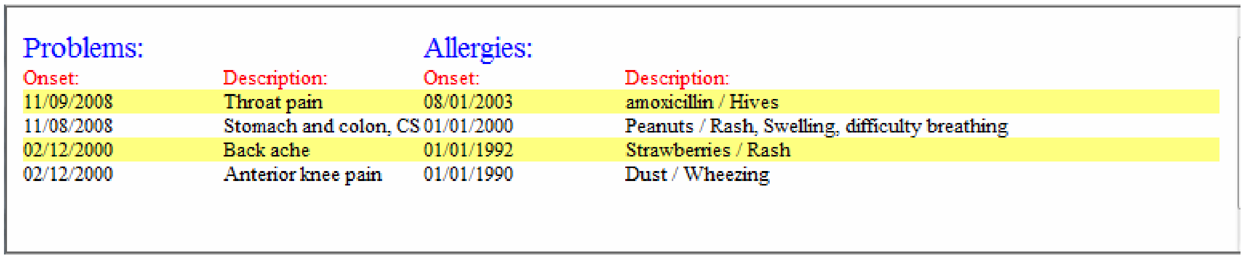


IF (EVEN(lcnt2) != 1)

Item	
Back Color	🟨 255; 255; 128
Back Mode	Opaque
Font	
Font Name	Times
Font Size	10
Font Bold	No
Font Italic	No
Font Underline	No
Font Strikethrough	No
Font Color	⬛ 0; 0; 0
Border	
Border Left	No

Note that the for loop in the code segments you created above sets the lcnt2 variable to one and increments it by one for each position in the record structure lists, so the condition EVEN(lcnt2) !=1 will be true for every other row returned by the query. Thus, every other row of the output will be displayed using the background color you selected.

The output from executing the layout using the conditional formatting set above may look something like the following example:



Problems:		Allergies:	
Onset:	Description:	Onset:	Description:
11/09/2008	Throat pain	08/01/2003	amoxicillin / Hives
11/08/2008	Stomach and colon, CS	01/01/2000	Peanuts / Rash, Swelling, difficulty breathing
02/12/2000	Back ache	01/01/1992	Strawberries / Rash
02/12/2000	Anterior knee pain	01/01/1990	Dust / Wheezing

You are now ready to use the VS_Header1, VS_Header2, and DetailRow sections to display the vital sign information.

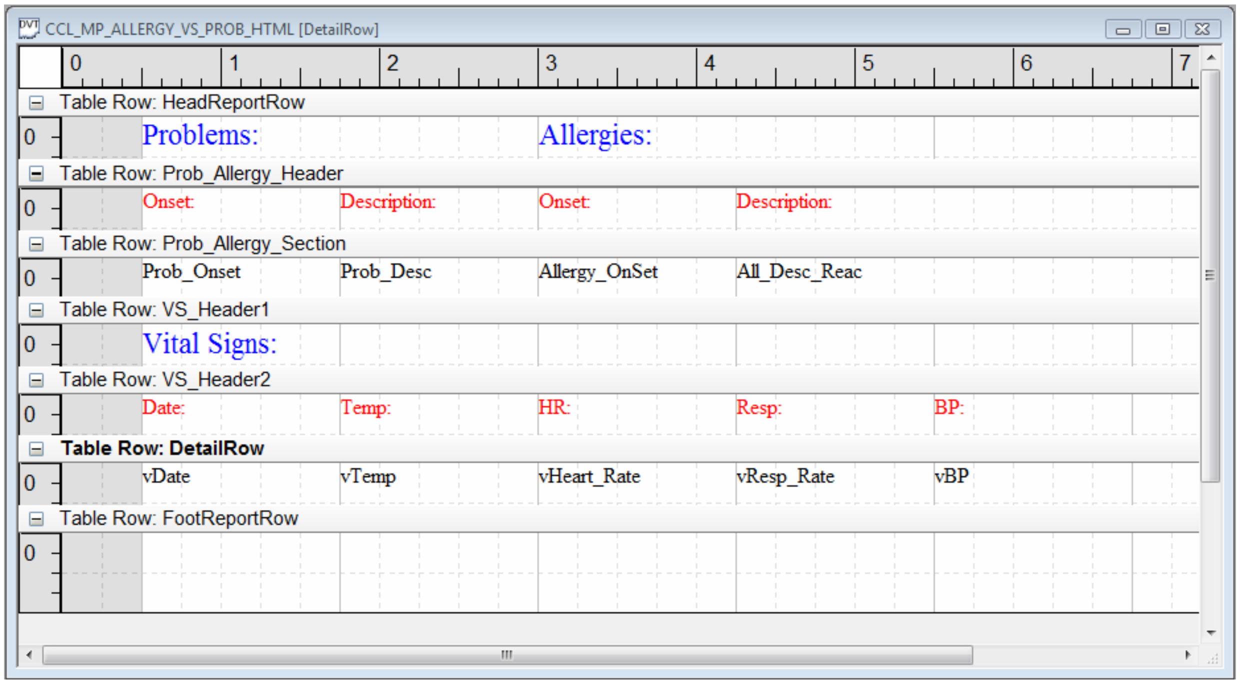
20. Set the source property for the first cell in the VS_Header1 section to "Vital Signs:". Use the same font size and color that was used for the Problems: and Allergies: labels.

21. Next use the following as the source for the first five cells in the VS_Header2 section:
    "Date:"
    "Temp:"
    "HR:"
    "Resp:"
    "BP:"

22. Use the same font size and color that was used for the Onset: and Description: labels you created earlier.

23. You can now use the following select expressions that were created by the query that was named Display_Data and associated with the layout earlier as the source for the five cells in the DetailRow section.
    vDate
    vTemp
    vHeart_Rate
    vResp_Rate
    vBP

Your layout should look similar to the following example:



Note that in the above example, "vDate", "vTemp", "vHeart_Rate", "vResp_Rate" and "vBP" are the names that were assigned to the cells in the DetailRow. However, the names displayed in your layout will most likely be different. For example, you may have FieldName0 – FieldName4 displayed. To change the name to something more meaningful, enter the appropriate text in the Name property for the cell.

Conditional formatting can be added to the cells in the DetailRow to cause every other row of the output to be displayed on a different color background.

24. Click in the DetailRow and use Ctrl+A to select all items in the section.

25. On the properties page select the Conditional Formatting property and then click the ellipsis ... button to open the Conditional Formatting dialog.

26. Click the Add button to add a condition and then select the newly added condition to activate the IF() control.

27. Enter **EVEN(D1.Seq) != 1** in the IF() control.

28. Change the Back Mode property from Transparent to Opaque.

29. Select a light color for the Back Color property.

Making the above changes causes every other row of vital sign information to be displayed on a different color background. However, suppose you also wanted to flex the font color of the blood pressure if the systolic blood pressure is 140 or greater or the diastolic blood pressure is 90 or greater. Multiple conditions can be added to the conditional formatting to accomplish this.

30. Select the cell in your DetailRow that displays the blood pressure (the above example named this cell vBP).

31. On the properties page select the Conditional Formatting property and then click the ellipsis ... button to open the Conditional Formatting dialog.

Currently the IF() control for Condition1 is EVEN(D1.SEQ) != 1. This condition sets the back mode to opaque and the back color to a light color.

32. Change the IF() control for Condition1 from

```
EVEN(D1.SEQ) != 1
```

to

even(d1.seq) != 1
and cnvtreal(vs_rec->vlist[d1.seq].sys_bp) >= 140
or
cnvtreal(vs_rec->vlist[d1.seq].dia_bp) >=90

and set the font color property to RED.

The formatting for Condition1 will display the blood pressure value in red on a light color background if d1.seq is an odd number and either the systolic blood pressure is >= 140 or the diastolic blood pressure is >= 90.

33. Add another condition and set the IF() control for Condition2 to

even(d1.seq) = 1
and cnvtreal(vs_rec->vlist[d1.seq].sys_bp) >= 140
or
cnvtreal(vs_rec->vlist[d1.seq].dia_bp) >=90

Also, set the font color property to RED.

The formatting for Condition2 will display the blood pressure value in red on the default color background if d1.seq is an even number and either the systolic blood pressure is >= 140 or the diastolic blood pressure is >= 90.

34. Add another condition. Set the IF() control for Condition3 to

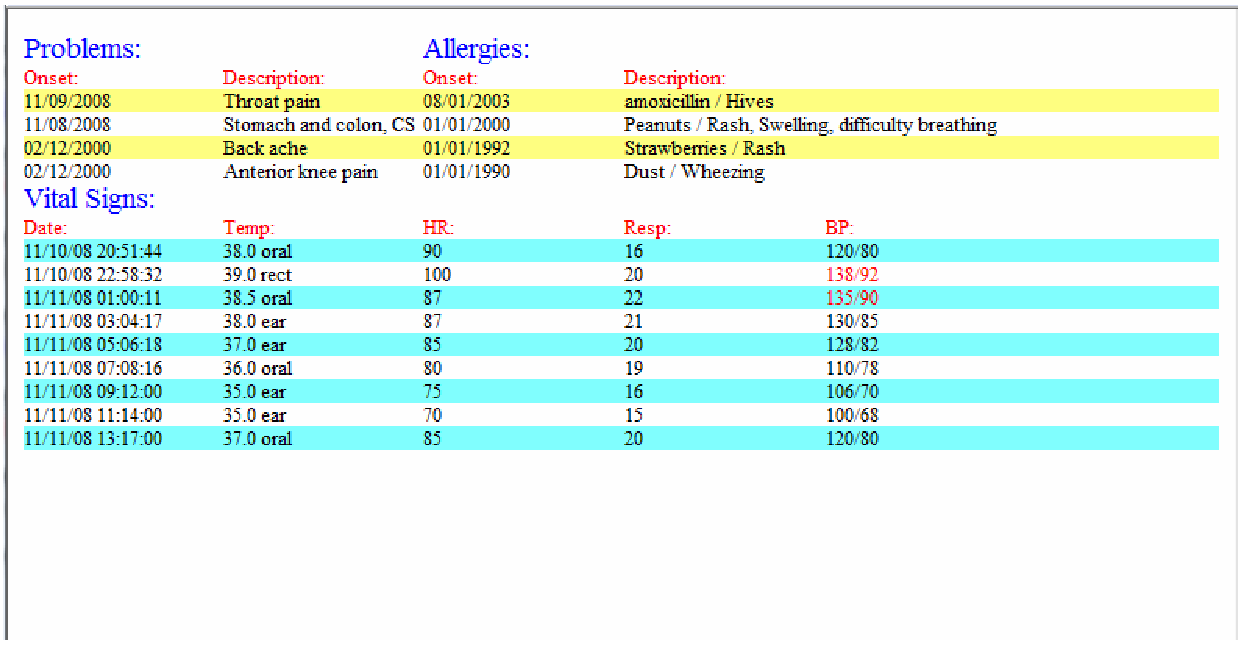
even(d1.seq) != 1

Change the Back Mode property from Transparent to Opaque select a light color for the Back Color property.

The formatting for Condition3 will display the blood pressure value in the default font color on a light color background if d1.seq is an odd number and both the systolic blood pressure is < 140 and the diastolic blood pressure is < 90.

If none of the conditions are met, the default formatting will be used to display the blood pressure value in the default font color on the default color background.

The output from executing the layout using the conditional formatting set above may look something like the following example:



Problems:

Onset:	Description:
11/09/2008	Throat pain
11/08/2008	Stomach and colon, CS
02/12/2000	Back ache
02/12/2000	Anterior knee pain

Allergies:

Onset:	Description:
08/01/2003	amoxicillin / Hives
01/01/2000	Peanuts / Rash, Swelling, difficulty breathing
01/01/1992	Strawberries / Rash
01/01/1990	Dust / Wheezing

Vital Signs:

Date:	Temp:	HR:	Resp:	BP:
11/10/08 20:51:44	38.0 oral	90	16	120/80
11/10/08 22:58:32	39.0 rect	100	20	138/92
11/11/08 01:00:11	38.5 oral	87	22	135/90
11/11/08 03:04:17	38.0 ear	87	21	130/85
11/11/08 05:06:18	37.0 ear	85	20	128/82
11/11/08 07:08:16	36.0 oral	80	19	110/78
11/11/08 09:12:00	35.0 ear	75	16	106/70
11/11/08 11:14:00	35.0 ear	70	15	100/68
11/11/08 13:17:00	37.0 oral	85	20	120/80

Example Adding a link to Medication Report from the Patient Summary Report

One of the biggest benefits of MPages is the ability to display HTML output. Using an HTML format allows you to create interactive output. For example, links can be added to the output that navigate down to subsequent detail, launch another application, or open a specific web site. To demonstrate some of this functionality, we will add a link to the patient summary program to execute a second report to display the medications for the patient. Obviously, the medications could be added directly to the patient summary report, but we will use a second program to demonstrate the linking functionality of using HTML output.

1. Use DVDev (DiscernVisualDeveloper.exe) to create a new layout program named something like ***1_initials*_GET_MEDS_HTML** (note there is a 30 character limit on the length of the layout program name).

2. Using the 2007.18 version of Layout Builder, select HTML as the Output Type, select Table View as the Report Layout, set the Number of Columns to 3 and click the Finish button.

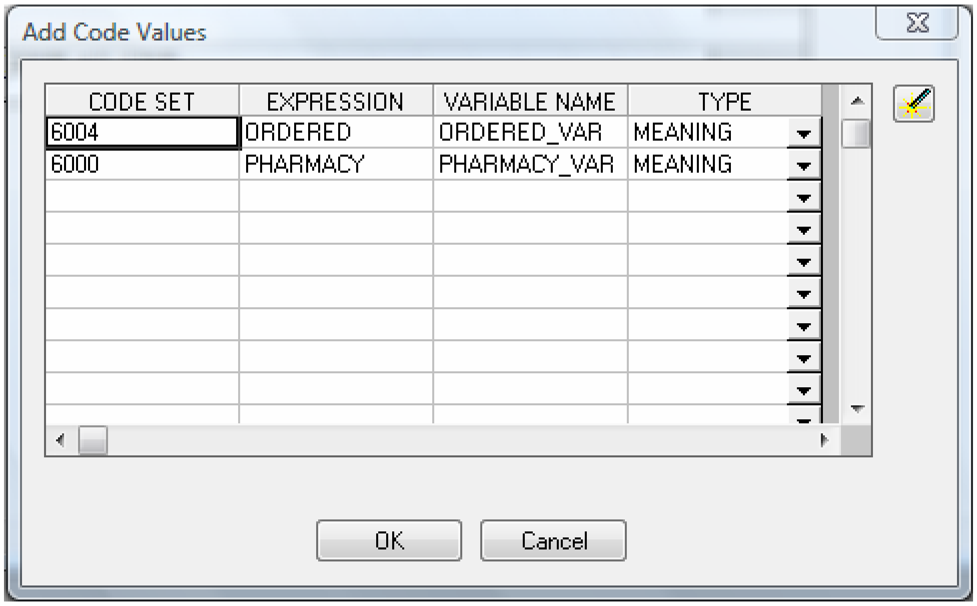
During the building and testing process, it may be easier to execute the layout program as a prompt program from within DVDev (DiscernVisualDeveloper.exe) than it would be to execute it as an MPage from PowerChart. Also, having prompts allows the passing of context variables to the program when it is executed as an MPage.

3. Use the Tools>Transfer Objects option in DVDev (DiscernVisualDeveloper.exe) to copy the prompt form from the *1_initials*_ALLERGY_VS_PROB_HTML program to create the prompt form for your *1_initials*_GET_MEDS_HTML program.

4. Use the Tools>Add Code Values… menu option to create a variable named ORDERED_VAR using Code Set 6004 and CDF_Meaning of ORDERED.

5. Use the Tools>Add Code Values… menu option to create a variable named PHARMACY_VAR using Code Set 6000 and CDF_Meaning of PHARMACY.

Your Add Code Values dialog should look similar to the following example:

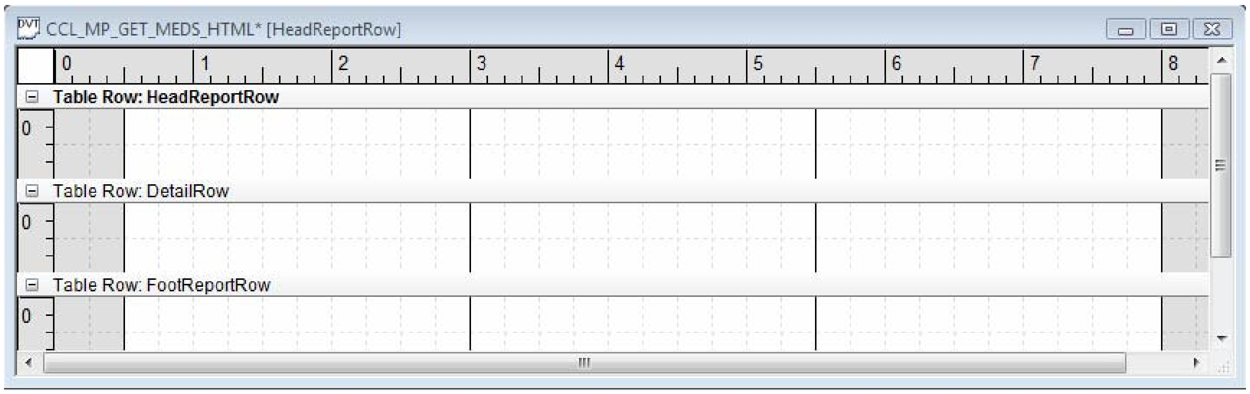


CODE SET	EXPRESSION	VARIABLE NAME	TYPE
6004	ORDERED	ORDERED_VAR	MEANING
6000	PHARMACY	PHARMACY_VAR	MEANING

Use the query builder to add the following query to your *1_initials*_GET_MEDS_HTML program. You may want to name the query Get_Data.

```
SELECT
  O_CATALOG_DISP = UAR_GET_CODE_DISPLAY(O.CATALOG_CD)
  , O.ORDER_MNEMONIC
  , O.CLINICAL_DISPLAY_LINE

FROM
  ORDERS   O

WHERE O.PERSON_ID = $PID
 AND O.ENCNTR_ID = $EID
 AND O.CATALOG_TYPE_CD+0 = PHARMACY_VAR
 AND O.ORDER_STATUS_CD+0 = ORDERED_VAR

WITH NOCOUNTER, SEPARATOR=" ", FORMAT
```

Your layout should look similar to the following example:

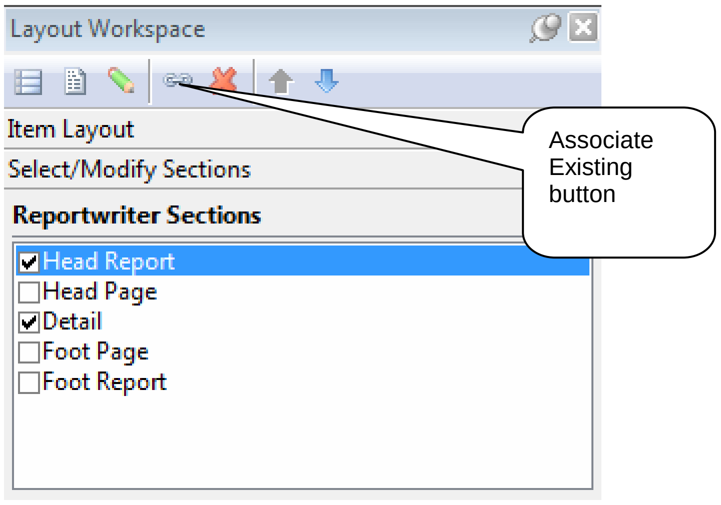


6. Adjust the size of the columns so that the first column is about 1.25 inches wide and the second column is about 1.75 inches wide.

7. Click in the Table Row: FootReportRow section and then use the Edit>Remove>Section menu option to delete that section.

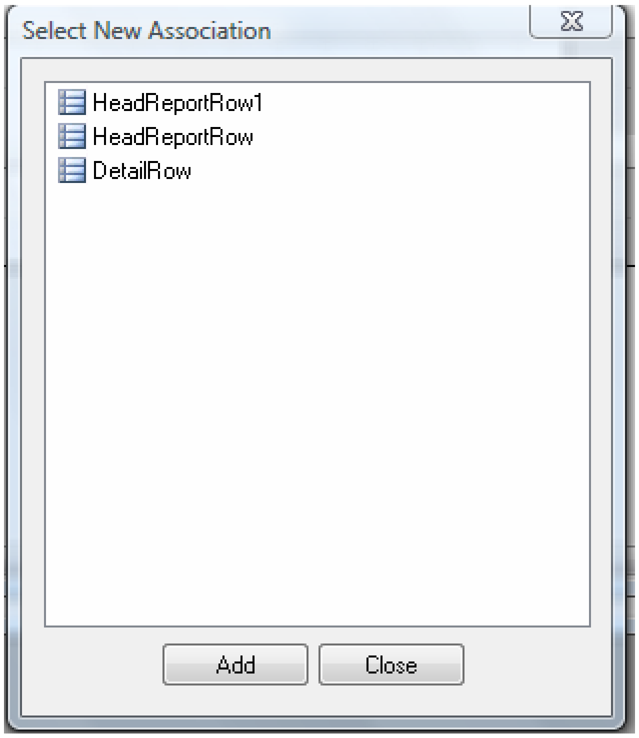
8. Click in the Table Row: HeadReportRow section and use the Table>Insert Row menu option to add a new row to the table. In the Section Properties, change the name of this section to **HeadReportRow1**.

9. In the Layout Workspace, highlight the Head Report Reportwriter section and click the Associate Existing button.



- [✓] Head Report
- [ ] Head Page
- [✓] Detail
- [ ] Foot Page
- [ ] Foot Report

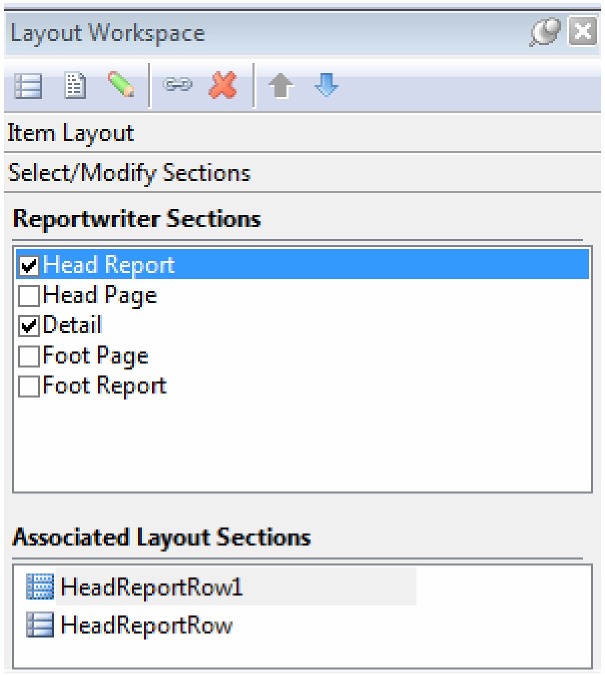
The Select New Association dialog opens.



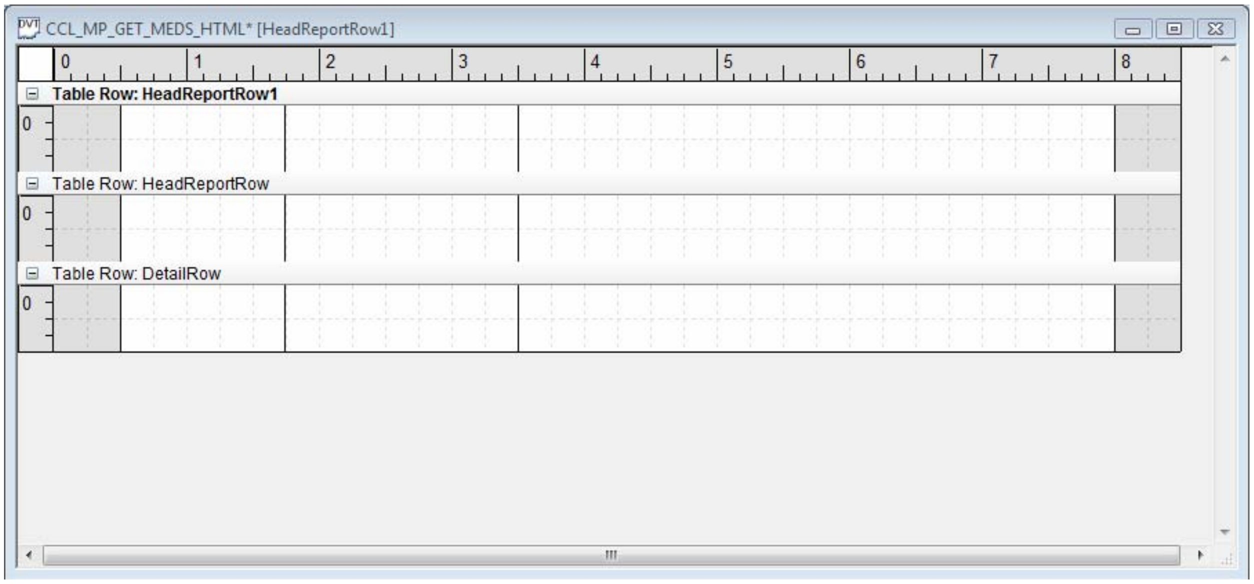
10. Select HeadReportRow1 and click the Add button to associate the HeadReportRow1 table row section with the Head Report Reportwriter section.

11. Click the Up Arrow in the Layout Workspace and move the HeadReportRow1 table row above the HeadReportRow.

Your Layout Workspace should look like the following example:



Your layout should now look similar to the following example:

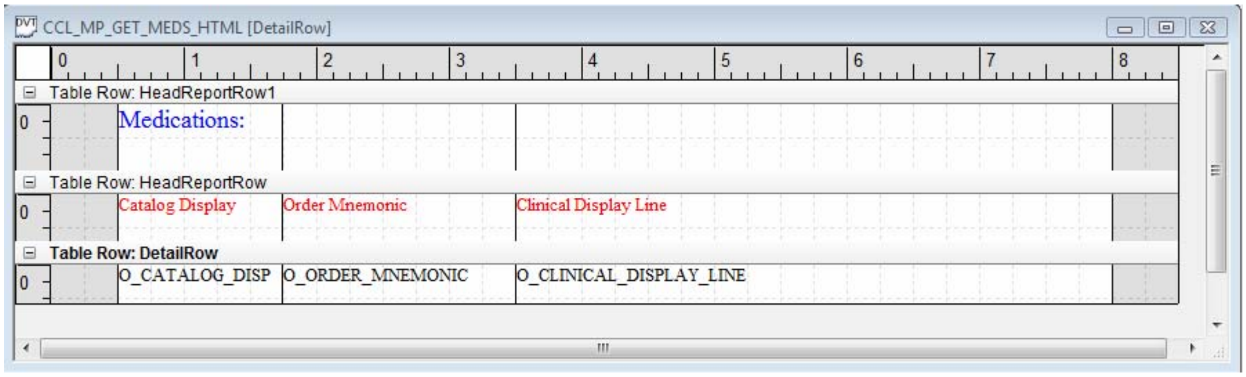

CCL_MP_GET_MEDS_HTML* [HeadReportRow1]
Table Row: HeadReportRow1
Table Row: HeadReportRow
Table Row: DetailRow

12. Set the Source property for the first cell in the HeadReportRow1 section to **Medications:**. Use a slightly larger font and change the font color to make this label stand out.

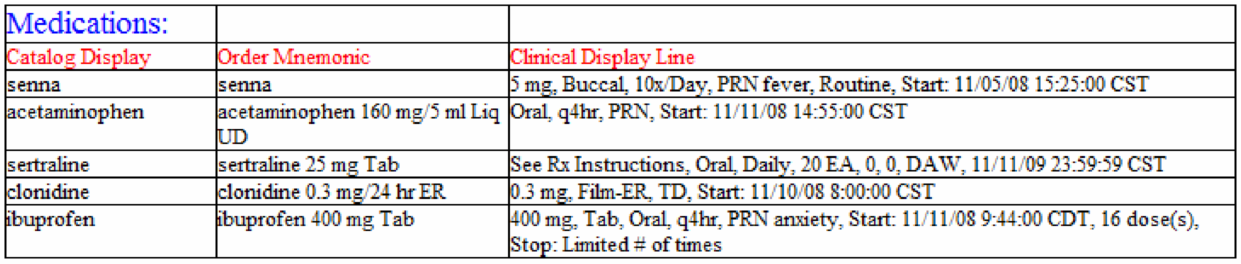
13. Set the Source property of the three cells in the HeadReportRow section to Catalog Display, Order Mnemonic, and Clinical Display Line. Use a different color font to make these labels stand out.

14. Set the Source property of the three cells in the DetailRow section to, O_CATALOG_DISP, O.ORDER_MNEMONIC, and O.CLINICAL_DISPLAY_LINE. You can use the Drop Items functionality to drag and drop these fields from the Select Fields list directly onto the layout.

Your layout should look similar to the following example:



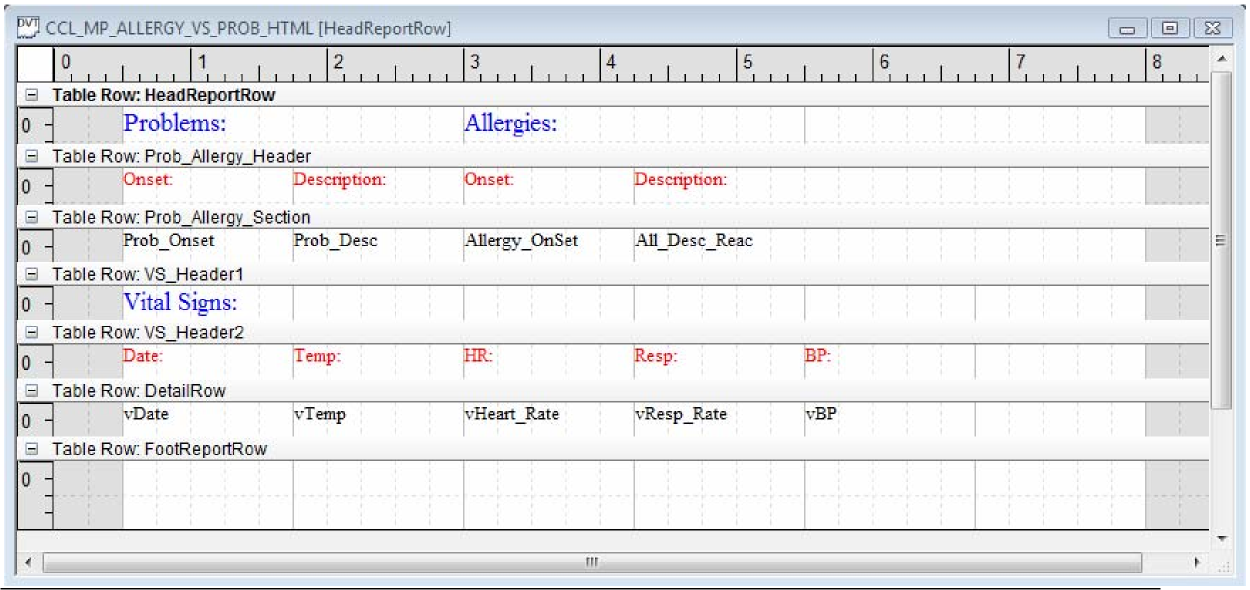
The output from executing the *1_initials*_GET_MEDS_HTML layout program should look something like the following example:



Medications:		
Catalog Display	Order Mnemonic	Clinical Display Line
senna	senna	5 mg, Buccal, 10x/Day, PRN fever, Routine, Start: 11/05/08 15:25:00 CST
acetaminophen	acetaminophen 160 mg/5 ml Liq UD	Oral, q4hr, PRN, Start: 11/11/08 14:55:00 CST
sertraline	sertraline 25 mg Tab	See Rx Instructions, Oral, Daily, 20 EA, 0, 0, DAW, 11/11/09 23:59:59 CST
clonidine	clonidine 0.3 mg/24 hr ER	0.3 mg, Film-ER, TD, Start: 11/10/08 8:00:00 CST
ibuprofen	ibuprofen 400 mg Tab	400 mg, Tab, Oral, q4hr, PRN anxiety, Start: 11/11/08 9:44:00 CDT, 16 dose(s), Stop: Limited # of times

Now that we have created the example medications program, we want to add a link to display in the output of the *1_initials*_ALLERGY_VS_PROB_HTML program. This link will execute the *1_initials*_GET_MEDS_HTML program to display the medications for the person whose allergies, vital signs, and problems are currently being displayed.

15. Open the *1_initials*_ALLERGY_VS_PROB_HTML layout program that you created earlier. The layout should look something like:



CCL_MP_ALLERGY_VS_PROB_HTML [HeadReportRow]

Table Row: HeadReportRow
Problems: | Allergies:

Table Row: Prob_Allergy_Header
Onset: | Description: | Onset: | Description:

Table Row: Prob_Allergy_Section
Prob_Onset | Prob_Desc | Allergy_OnSet | All_Desc_Reac

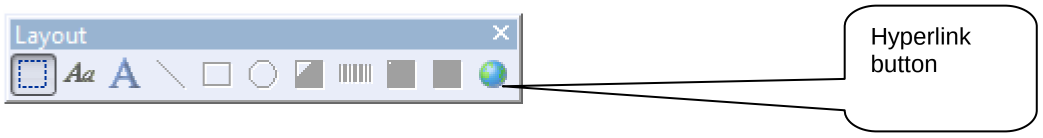
Table Row: VS_Header1
Vital Signs:

Table Row: VS_Header2
Date: | Temp: | HR: | Resp: | BP:

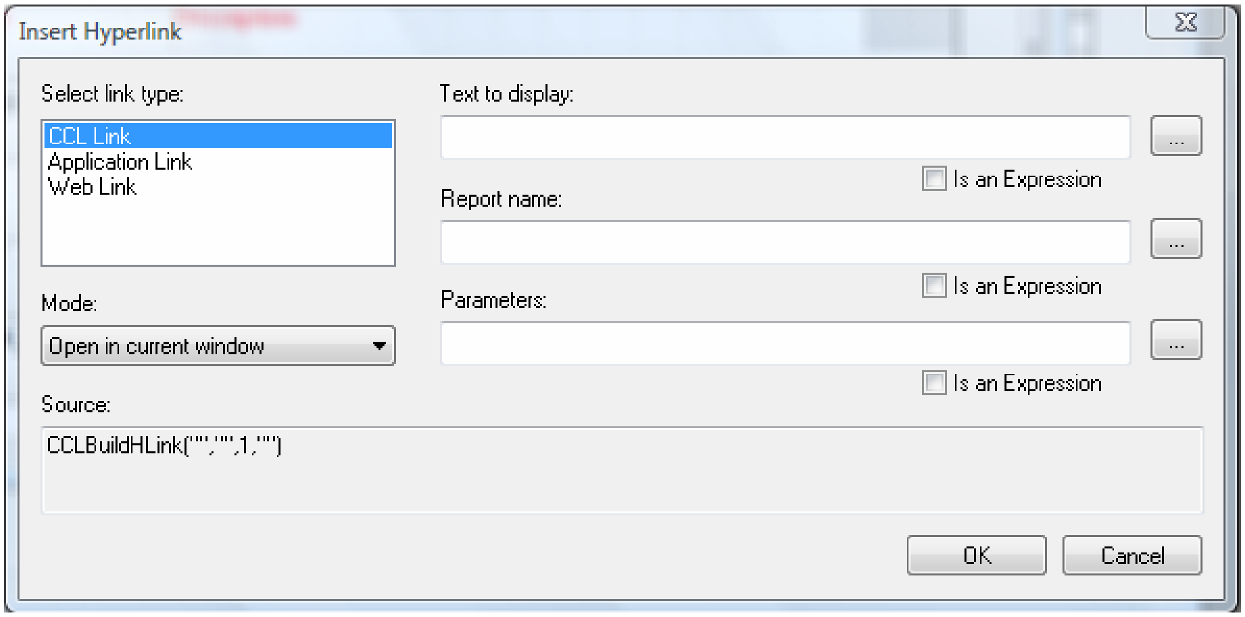
Table Row: DetailRow
vDate | vTemp | vHeart_Rate | vResp_Rate | vBP

Table Row: FootReportRow

16. Currently there is an empty cell on the right side of the HeadReportRow section. Click in that cell and then click the Hyperlink button on the Layout Toolbar.



Clicking the Hyperlink button opens the Insert Hyperlink dialog which can be used to create a hyperlink in the output of the report.



Source:

CCLBuildHLink("","",1,"")

17. Use the following to set the properties of the Insert Hyperlink dialog:

Text to display:
Display Medications:

Report Name:
*1_initials*_GET_MEDS_HTML

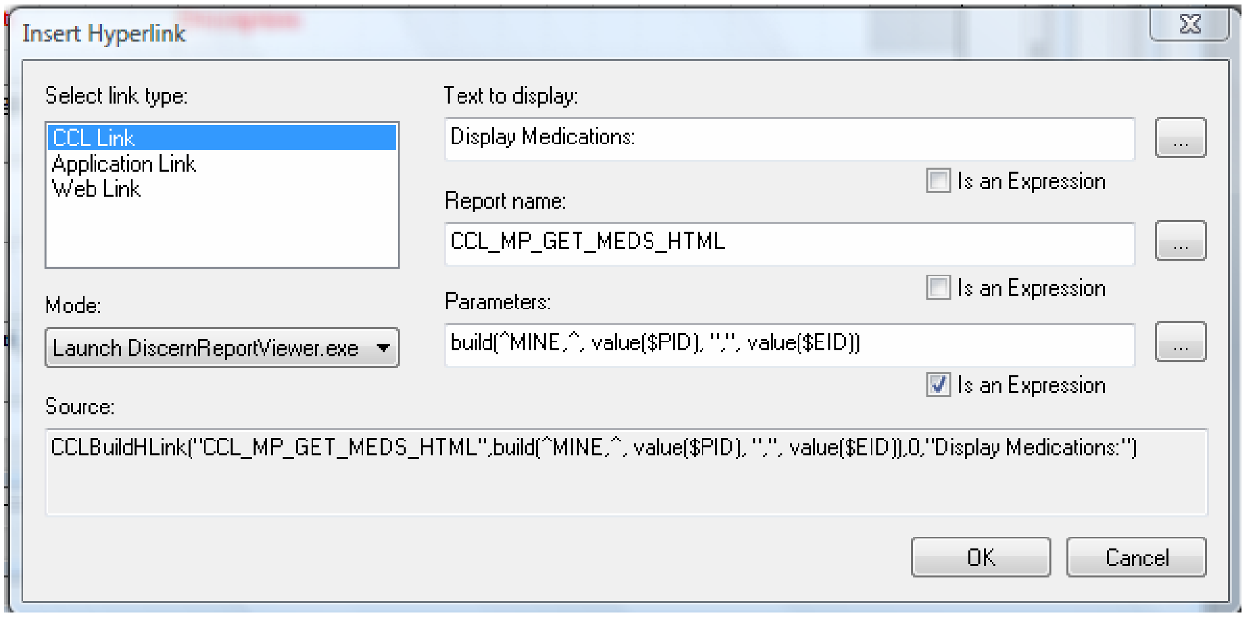
Note if your *1_initials*_GET_MEDS_HTML is a group1 object you will need to add ":group1" to the end of the report name. For example
*1_initials*_GET_MEDS_HTML:group1

Parameters:
build(^MINE,^, value($PID), ",", value($EID))

18. Select *the Is an Expression* option for the Parameters:

19. Change the Mode: from *Open in current window* to *Launch DiscernReportViewer.exe*.

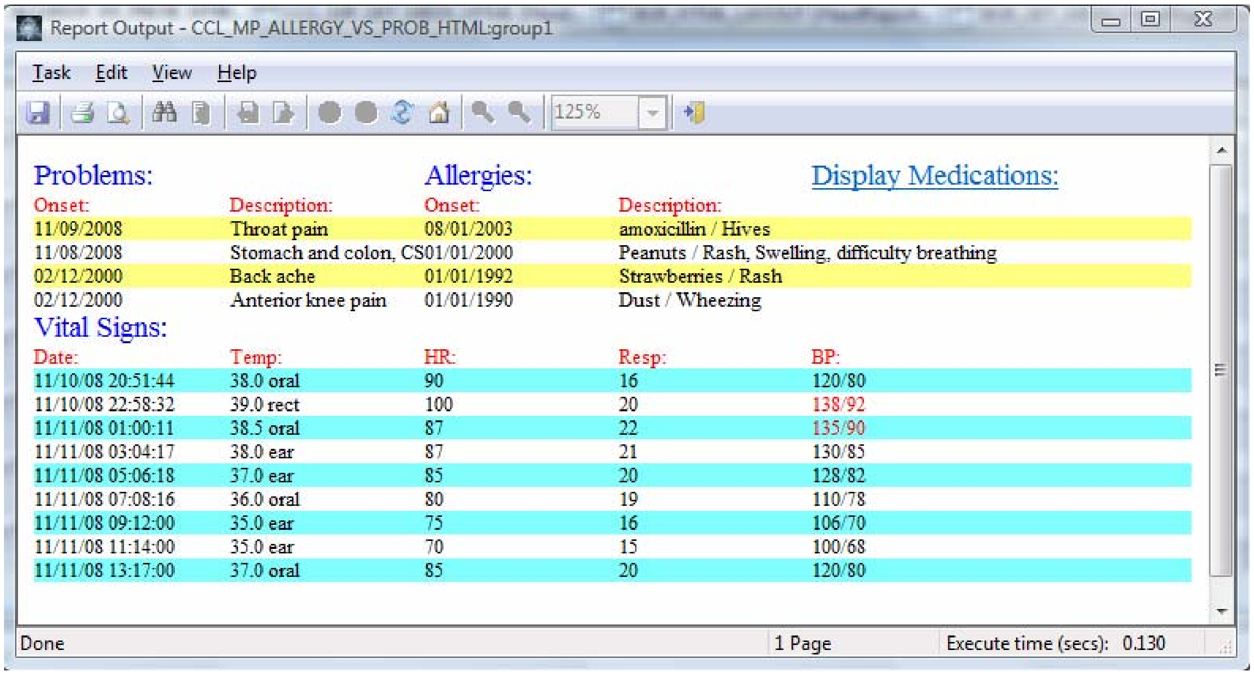
Your Insert Hyperlink dialog should look similar to the following example:



Text to display: Display Medications:
☐ Is an Expression
Report name: CCL_MP_GET_MEDS_HTML
☐ Is an Expression
Parameters: build(^MINE,^, value($PID), ",", value($EID))
☑ Is an Expression
Mode: Launch DiscernReportViewer.exe
Source:

```
CCLBuildHLink("CCL_MP_GET_MEDS_HTML",build(^MINE,^, value($PID), ",", value($EID)),0,"Display Medications:")
```

20. Save and execute your *1_initials*_ALLERGY_VS_PROB_HTML layout program. The output should look similar to the following example:


Report Output - CCL_MP_ALLERGY_VS_PROB_HTML:group1
Task
Edit
View
Help
125%
Done
1 Page
Execute time (secs): 0.130

Problems:		Allergies:		Display Medications:
Onset:	Description:	Onset:	Description:
11/09/2008	Throat pain	08/01/2003	amoxicillin / Hives
11/08/2008	Stomach and colon, CS	01/01/2000	Peanuts / Rash, Swelling, difficulty breathing
02/12/2000	Back ache	01/01/1992	Strawberries / Rash
02/12/2000	Anterior knee pain	01/01/1990	Dust / Wheezing

Vital Signs:

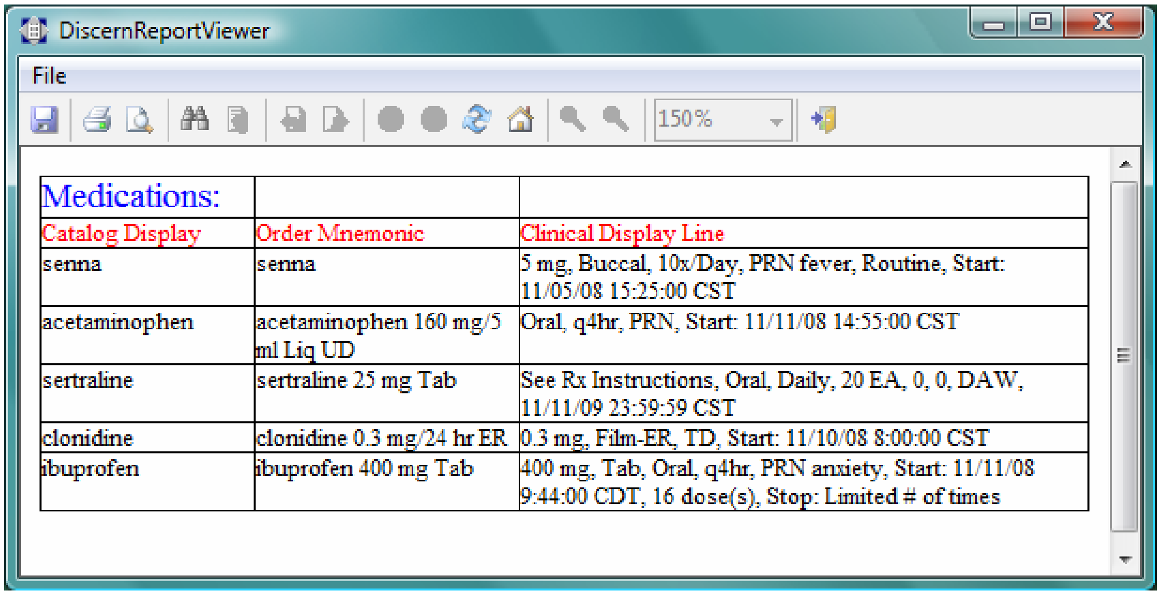
Date:	Temp:	HR:	Resp:	BP:
11/10/08 20:51:44	38.0 oral	90	16	120/80
11/10/08 22:58:32	39.0 rect	100	20	138/92
11/11/08 01:00:11	38.5 oral	87	22	135/90
11/11/08 03:04:17	38.0 ear	87	21	130/85
11/11/08 05:06:18	37.0 ear	85	20	128/82
11/11/08 07:08:16	36.0 oral	80	19	110/78
11/11/08 09:12:00	35.0 ear	75	16	106/70
11/11/08 11:14:00	35.0 ear	70	15	100/68
11/11/08 13:17:00	37.0 oral	85	20	120/80

21. Click on the Display Medications link to execute your *1_initials*_GET_MEDS_HTML program.

The *1_initials_ALLERGY_VS_PROB_HTML* program uses prompts named $PID and $EID to get a person and encounter ID from the user. The *1_initials*_GET_MEDS_HTML program prompts for the same values. When the Display Medications: link was added, the Parameters property was used to pass the values that were entered at the *1_initials*_ALLERGY_VS_PROB_HTML program to the *1_initials*_GET_MEDS_HTML program, so that the medications displayed are for the same person and encounter.

When the *1_initials*_ALLERGY_VS_PROB_HTML is set up to execute as an MPage, the context sensitive variables, $PAT_PersonID$ and $VIS_EncntrID$, are passed to its prompts from *PowerChart*. Those values are subsequently passed to the *1_initials*_GET_MEDS_HTML program when the Display Medications link is used.

The output of your *1_initials*_GET_MEDS_HTML program should look similar to the following example:

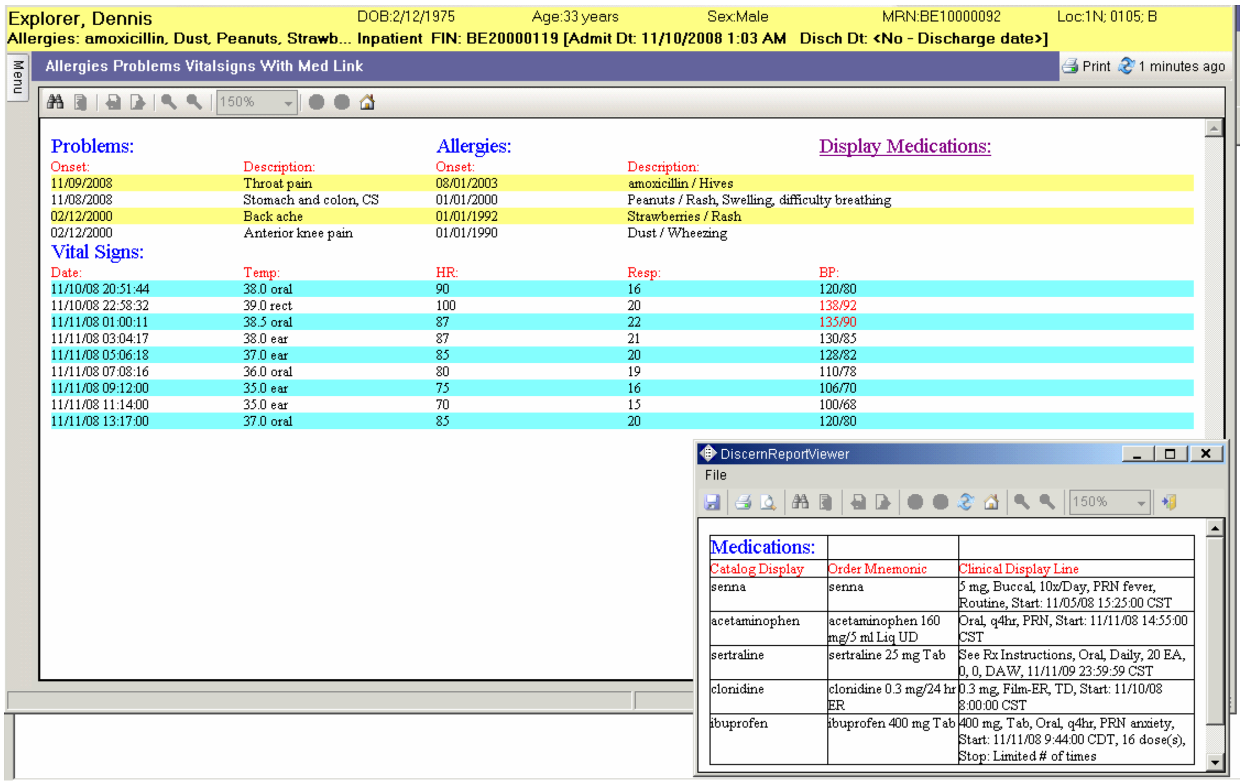


Medications:		
Catalog Display	Order Mnemonic	Clinical Display Line
senna	senna	5 mg, Buccal, 10x/Day, PRN fever, Routine, Start: 11/05/08 15:25:00 CST
acetaminophen	acetaminophen 160 mg/5 ml Liq UD	Oral, q4hr, PRN, Start: 11/11/08 14:55:00 CST
sertraline	sertraline 25 mg Tab	See Rx Instructions, Oral, Daily, 20 EA, 0, 0, DAW, 11/11/09 23:59:59 CST
clonidine	clonidine 0.3 mg/24 hr ER	0.3 mg, Film-ER, TD, Start: 11/10/08 8:00:00 CST
ibuprofen	ibuprofen 400 mg Tab	400 mg, Tab, Oral, q4hr, PRN anxiety, Start: 11/11/08 9:44:00 CDT, 16 dose(s), Stop: Limited # of times

When the hyperlink was created using the Insert Hyperlink dialog, the Mode: property was set to *Launch DiscernReportViewer.exe*. This causes the output of the *1_initials*_GET_MEDS_HTML program to be opened in a separate, maximized window. Double-clicking the title bar of that window restores it down and allows it to be resized so the output of both the *1_initials*_GET_MEDS_HTML and the *1_initials*_ALLERGY_VS_PROB_HTML programs can be viewed on the same screen.

The *1_initials*_ALLERGY_VS_PROB_HTML program created in the above example can be set up to execute as an MPage from *PowerChart* by using Preference Maintenance (PrefMaint.exe) and entering ***1_initials*_ALLERGY_VS_PROB_HTML** as the REPORT_NAME preference and **"MINE", $PAT_PersonID$, $VIS_EncntrID$** as the REPORT_PARAM preference.

The output from running the *1_initials*_ALLERGY_VS_PROB_HTML as an MPage, then clicking the Display Medications link, and resizing the new window, may look something like the following example:



Explorer, Dennis DOB:2/12/1975 Age:33 years Sex:Male MRN:BE10000092 Loc:1N; 0105; B
Allergies: amoxicillin, Dust, Peanuts, Strawb... Inpatient FIN: BE20000119 [Admit Dt: 11/10/2008 1:03 AM Disch Dt: <No - Discharge date>]

Allergies Problems Vitalsigns With Med Link

Problems:

Onset:	Description:
11/09/2008	Throat pain
11/08/2008	Stomach and colon, CS
02/12/2000	Back ache
02/12/2000	Anterior knee pain

Allergies:

Onset:	Description:
08/01/2003	amoxicillin / Hives
01/01/2000	Peanuts / Rash, Swelling, difficulty breathing
01/01/1992	Strawberries / Rash
01/01/1990	Dust / Wheezing

Display Medications:

Vital Signs:

Date:	Temp:	HR:	Resp:	BP:
11/10/08 20:51:44	38.0 oral	90	16	120/80
11/10/08 22:58:32	39.0 rect	100	20	138/92
11/11/08 01:00:11	38.5 oral	87	22	135/90
11/11/08 03:04:17	38.0 ear	87	21	130/85
11/11/08 05:06:18	37.0 ear	85	20	128/82
11/11/08 07:08:16	36.0 oral	80	19	110/78
11/11/08 09:12:00	35.0 ear	75	16	106/70
11/11/08 11:14:00	35.0 ear	70	15	100/68
11/11/08 13:17:00	37.0 oral	85	20	120/80

DiscernReportViewer

Medications:		
Catalog Display	Order Mnemonic	Clinical Display Line
senna	senna	5 mg, Buccal, 10x/Day, PRN fever, Routine, Start: 11/05/08 15:25:00 CST
acetaminophen	acetaminophen 160 mg/5 ml Liq UD	Oral, q4hr, PRN, Start: 11/11/08 14:55:00 CST
sertraline	sertraline 25 mg Tab	See Rx Instructions, Oral, Daily, 20 EA, 0, 0, DAW, 11/11/09 23:59:59 CST
clonidine	clonidine 0.3 mg/24 hr ER	0.3 mg, Film-ER, TD, Start: 11/10/08 8:00:00 CST
ibuprofen	ibuprofen 400 mg Tab	400 mg, Tab, Oral, q4hr, PRN anxiety, Start: 11/11/08 9:44:00 CDT, 16 dose(s), Stop: Limited # of times

Example Patient Summary Report with HTML Output Created Manually in the Report Writer

HTML tags can be embedded in the output of a *Discern Explorer* select command. The following source code example shows how to create HTML output by manually embedding the HTML tags in the report writer sections of a select command.

Note: This topic is not intended to teach the creation and use of HTML tags. It is intended to show the person who already has a knowledge of HTML how it can be created in the report writer sections of a Discern Explorer Select command.

The following source code example creates a program named CCL_MP_MANUAL_CODED_HTML. It also includes a file named CCL_MP_GET_ALLERGY_VS_PROB.INC. When you completed the, Example Patient Summary Report with PostScript/PDF output Created in Layout Builder section, you created a file named, *1_initials*_GET_ALLERGY_VS_PROB.INC and saved it in a standard location. If you do not have a *1_initials*_GET_ALLERGY_VS_PROB.INC file saved in a standard location, go back to the Example Patient Summary Report with PostScript/PDF output Created in Layout Builder section and follow the steps for creating the *1_initials*_GET_ALLERGY_VS_PROB.INC file and saving it in a standard back-end location. If you are working in a multi-node AIX or HP_UX environment, the .inc file must be saved on all nodes or in a directory that is accessible from all nodes.

```
drop program ccl_mp_manual_coded_html go
create program ccl_mp_manual_coded_html

prompt
  "Output to File/Printer/MINE" = "MINE"
  , "Enter a Person_ID" = 0
  , "Enter an Encntr_ID" = 0
with OUTDEV, PID, EID

%i CCL_MP_GET_ALLERGY_VS_PROB.INC

SELECT into $outdev
  vDATE = substring(1, 30, vs_rec->vlist[d1.seq].date)
  , vTEMP = concat(trim(vs_rec->vlist[d1.seq].temp), " ",
      trim(vs_rec->vlist[d1.seq].temp_type))
  , vHEART_RATE = vs_rec->vlist[d1.seq].heart_rate
  , vRESP_RATE = vs_rec->vlist[d1.seq].resp_rate
  , vBP = if(vs_rec->vlist[d1.seq].sys_bp != "-")
      concat(trim(vs_rec->vlist[d1.seq].sys_bp), "/", trim(vs_rec->vlist[d1.seq].dia_bp))
    else  "-"
    endif
FROM
  (DUMMYT   D1  WITH SEQ = VALUE(SIZE(VS_REC->vlist, 5)))

PLAN D1

head report
  row +1 ^<html>^

  row +1 ^<head> <style type='text/css'>^,
  row +1 ^.myblue {color: blue; font: bold 20pt times;}^
  row +1 ^.myred {color: red}^
  row +1 ^</style> </head>^

  row +1 ^<body>^
  row +1 ^<table border='0' width='100%'>^
  row +1 call print(^<tr class='myblue'>^)
  /********************* Display Problems and Allergies *********************/
  row +1 call print(concat(^<td colspan='2'>^,  "Problems:", ^</td>^))
  row +1 call print(concat(^<td>^, "Allergies:", ^</td>^))
  row +1 call print(^</tr>^)
  row +1 call print(^<tr class='myred'>^)
  row +1 call print(concat(^<td>^,"Onset:", ^</td>^))
  row +1 call print(concat(^<td>^,"Description:" , ^</td>^))
  row +1 call print(concat(^<td>^,"Onset:", ^</td>^))
  row +1 call print(concat(^<td>^,"Description:" , ^</td>^))
  row +1 call print(^</tr>^)
  /*
  Make the allergy->list and the problem->list the same size.
  This will allow a single for loop to be used to display all the allergies and problems
  */
  if(problem->problem_cnt > allergy->allergy_cnt ) ;more problems than allergies
```

```
     lcnt = problem->problem_cnt
     stat = alterlist(allergy->list ,lcnt) ;pad the allergy list
  elseif( problem->problem_cnt < allergy->allergy_cnt) ;more allergies than problems
     lcnt = allergy->allergy_cnt
     stat = alterlist(problem->list, lcnt) ;pad the problem list
  else  ;equal number of allergies and problems
     lcnt = allergy->allergy_cnt
  endif
  ;start the for loop
  for(lcnt2 = 1 to lcnt)
     if(allergy->list[lcnt2].description > " ")
        reac_string = concat(allergy->list[lcnt2].description, " / ", allergy->list[lcnt2].reaction)
     else
        reac_string = " "
     endif
     row +1 call print(^<tr>^)
     row +1 call print(concat(^<td>^, problem->list[lcnt2].onset, ^</td>^))
     row +1 call print(concat(^<td>^, problem->list[lcnt2].description , ^</td>^))
     row +1 call print(concat(^<td>^, allergy->list[lcnt2].onset   , ^</td>^))
     row +1 call print(concat(^<td>^, reac_string, ^</td>^))
     row +1 call print(^</tr>^)
  endfor
/********************* Vital Signs Headers *********************/
  row +1 call print(concat(^<tr class='myblue'> <td>^, "Vital Signs:"  , ^</td> </tr>^))
  row +1 call print(^<tr class='myred'>^)
  row +1 call print(concat(^<td>^, "DATE:", ^</td>^))
  row +1 call print(concat(^<td>^, "TEMP:", ^</td>^))
  row +1 call print(concat(^<td>^, "HR:"  , ^</td>^))
  row +1 call print(concat(^<td>^, "RESP:", ^</td>^))
  row +1 call print(concat(^<td>^, "BP:"  , ^</td>^))
  row +1 call print(^</tr>^)

/********************* Vital Sign Details *********************/
detail
  row +1 call print(^<tr>^)
  row +1 call print(concat(^<td>^, vDATE, ^</td>^))
  row +1 call print(concat(^<td>^, vTEMP, ^</td>^))
  row +1 call print(concat(^<td>^, vHEART_RATE, ^</td>^))
  row +1 call print(concat(^<td>^, vRESP_RATE, ^</td>^))
  row +1 call print(concat(^<td>^, vBP, ^</td>^))
  row +1 call print(^</tr>^)
foot report
row +1 ^</table>^
row +1 ^</body>^
row +1 ^</html>^

WITH NOCOUNTER, SEPARATOR=" ", FORMAT

end
go
```

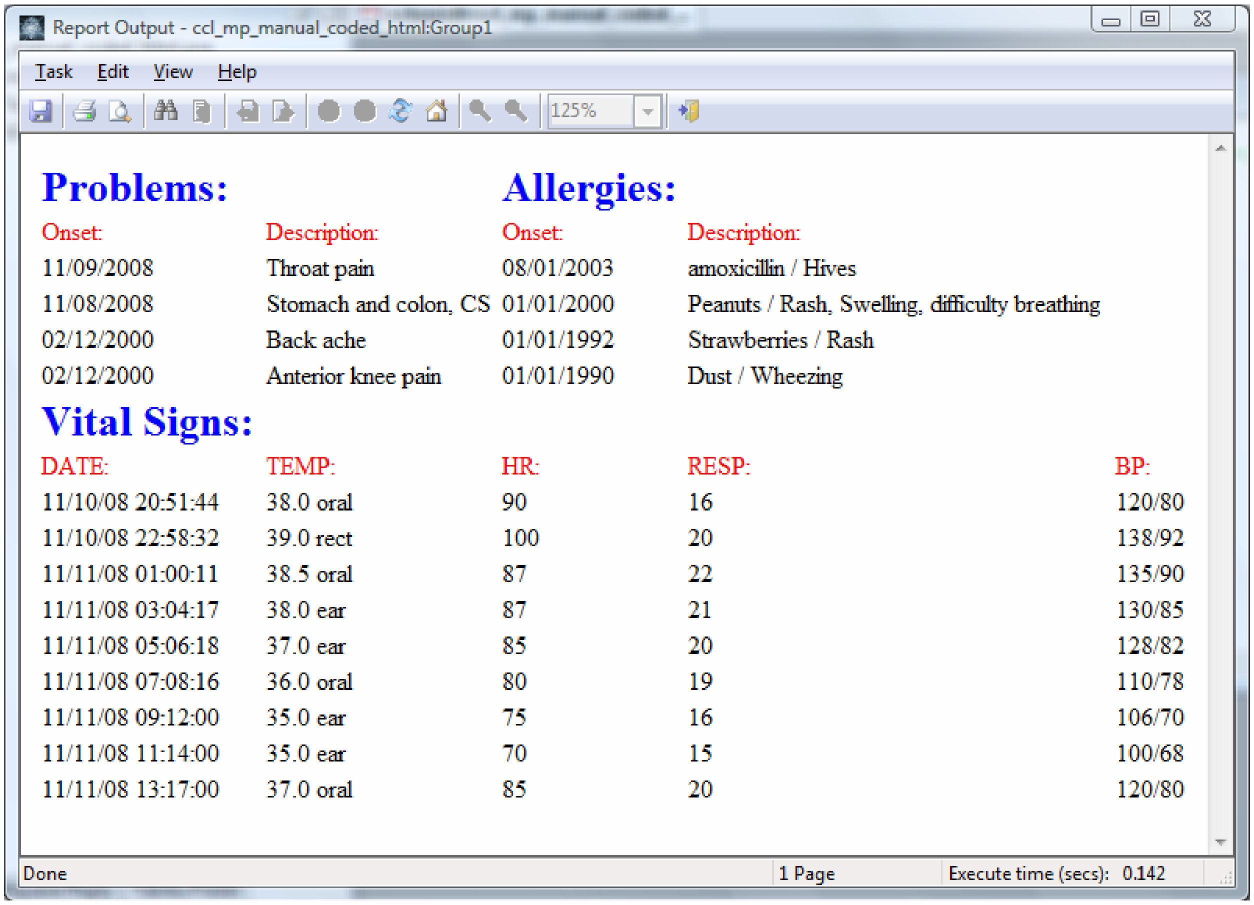
1. Assuming you have a *1_initials*_GET_ALLERGY_VS_PROB.INC file saved in the CCLUSERDIR: directory, you can copy and paste the above source code into a blank file in DVDev (DiscernVisualDeveloper.exe). Save the file as *1_initials*_MANUAL_CODED_HTML.PRG and change the

%i CCL_MP_GET_ALLERGY_VS_PROB.INC

to

%i *1_initials* _GET_ALLERGY_VS_PROB.INC

If you compile and execute the program, the output should look similar to the following example:


Report Output - ccl_mp_manual_coded_html:Group1
Task Edit View Help
125%
Done
1 Page
Execute time (secs): 0.142

Problems:

Onset:	Description:
11/09/2008	Throat pain
11/08/2008	Stomach and colon, CS
02/12/2000	Back ache
02/12/2000	Anterior knee pain

Allergies:

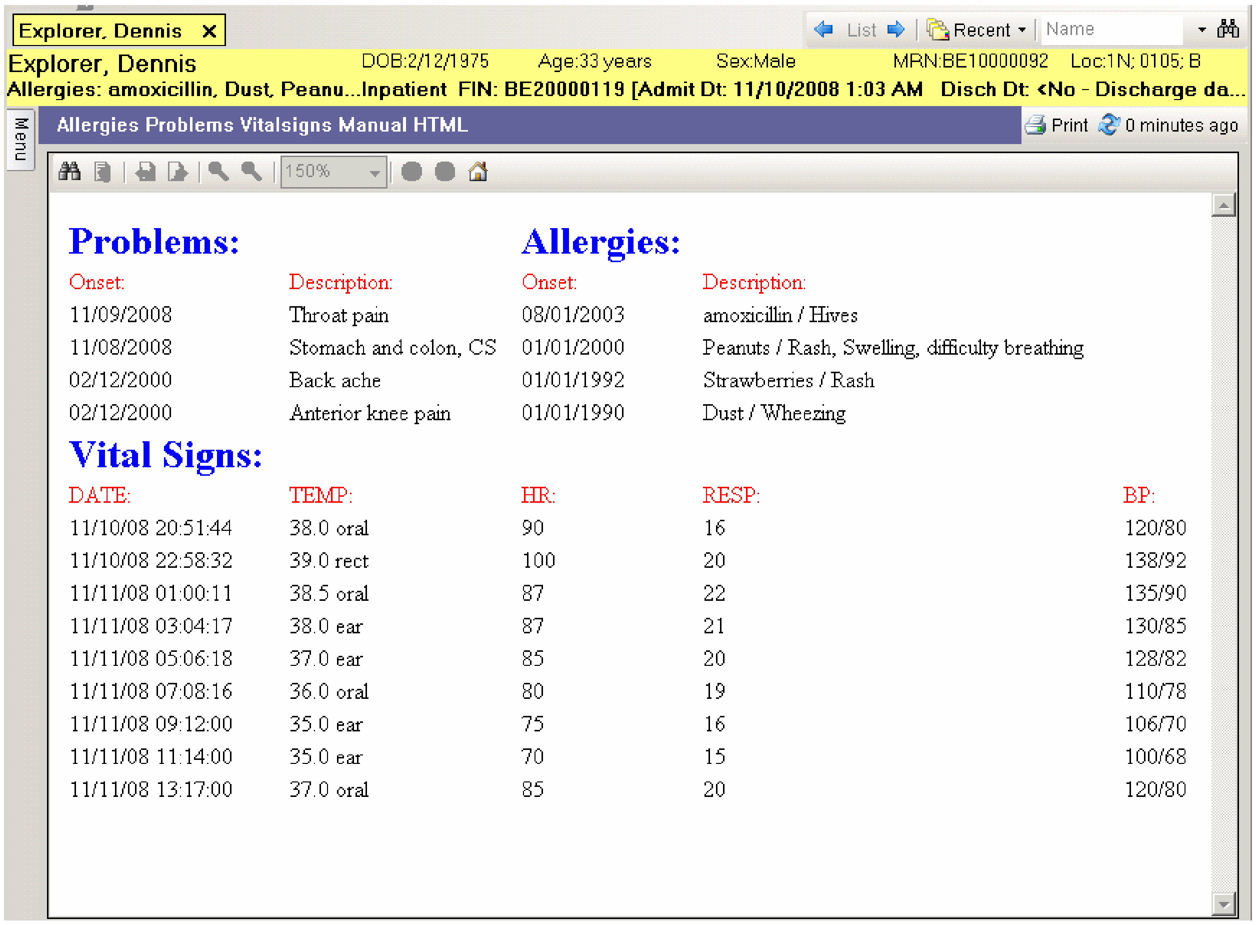
Onset:	Description:
08/01/2003	amoxicillin / Hives
01/01/2000	Peanuts / Rash, Swelling, difficulty breathing
01/01/1992	Strawberries / Rash
01/01/1990	Dust / Wheezing

Vital Signs:

DATE:	TEMP:	HR:	RESP:	BP:
11/10/08 20:51:44	38.0 oral	90	16	120/80
11/10/08 22:58:32	39.0 rect	100	20	138/92
11/11/08 01:00:11	38.5 oral	87	22	135/90
11/11/08 03:04:17	38.0 ear	87	21	130/85
11/11/08 05:06:18	37.0 ear	85	20	128/82
11/11/08 07:08:16	36.0 oral	80	19	110/78
11/11/08 09:12:00	35.0 ear	75	16	106/70
11/11/08 11:14:00	35.0 ear	70	15	100/68
11/11/08 13:17:00	37.0 oral	85	20	120/80

The *1_initials*_MANUAL_CODED_HTML program created in the above example, could be set up to execute as an MPage from *PowerChart* by using Preference Maintenance (PrefMaint.exe) and entering ***1_initials*_MANUAL_CODED_HTML** as the REPORT_NAME preference and **"MINE", $PAT_PersonID$, $VIS_EncntrID$** as the REPORT_PARAM preference.

The output from running the *1_initials*_MANUAL_CODED_HTML as an MPage should look something like the following example:



Explorer, Dennis DOB:2/12/1975 Age:33 years Sex:Male MRN:BE10000092 Loc:1N; 0105; B
Allergies: amoxicillin, Dust, Peanu...Inpatient FIN: BE20000119 [Admit Dt: 11/10/2008 1:03 AM Disch Dt: <No - Discharge da...

Allergies Problems Vitalsigns Manual HTML

Problems:

Onset:	Description:
11/09/2008	Throat pain
11/08/2008	Stomach and colon, CS
02/12/2000	Back ache
02/12/2000	Anterior knee pain

Allergies:

Onset:	Description:
08/01/2003	amoxicillin / Hives
01/01/2000	Peanuts / Rash, Swelling, difficulty breathing
01/01/1992	Strawberries / Rash
01/01/1990	Dust / Wheezing

Vital Signs:

DATE:	TEMP:	HR:	RESP:	BP:
11/10/08 20:51:44	38.0 oral	90	16	120/80
11/10/08 22:58:32	39.0 rect	100	20	138/92
11/11/08 01:00:11	38.5 oral	87	22	135/90
11/11/08 03:04:17	38.0 ear	87	21	130/85
11/11/08 05:06:18	37.0 ear	85	20	128/82
11/11/08 07:08:16	36.0 oral	80	19	110/78
11/11/08 09:12:00	35.0 ear	75	16	106/70
11/11/08 11:14:00	35.0 ear	70	15	100/68
11/11/08 13:17:00	37.0 oral	85	20	120/80

Document Revision History

Revision Number:	**Revision Date:**	**Description:**
001	February 27, 2009	Initial release of this *Cerner Millennium* Support Guide.
002	June 3, 2009	Reformatted Step 15 under Context Variables for greater clarity. No content was changed.