DOCUMENT STORAGE SYSTEMS, INC.

DSIO – CPRS

Modification Guide

Theodore Fontana

4/23/2014

Date	Revision	Description	Author
Not Independently Released			

Purpose

The purpose of this document is to identify, track, and provide reasoning to changes make to CPRS.

The changes fall into three categories:

1. SMART
2. TIU
3. Discreet Data

SMART

1. TFrmNotes.mnuActDeleteClick procedure change

Purpose:
Without Notifications.Clear processing a SMART alert would cause all other notes for that patient to be created with the SMART alert note values.

Unit: fNotes.pas

Notifications.Clear added to the if bATRval statement.

```
  //ATR
  //If the TIU Note being deleted is associated with a TAR Alert record,
  // then delete the TIU Note IEN from the TAR Alert record
  if bATRAval then
  begin
    slATRPtList := TStringList.Create;
    CallV('ORATR PATIENT',[Patient.DFN]);
    FastAssign(RPCBrokerV.Results, slATRPtList);
    for i := 0 to slATRPtList.Count - 1 do
    begin
      thisATRTIUNoteIEN := Piece(Piece(slATRPtList[i],'^',8),';',8);
      if thisATRTIUNoteIEN = intToStr(SavedDocIEN) then
      begin
        thisATRien := Piece(Piece(slATRPtList[i],'^',8),';',7);
        CallV('ORATR SET TIU NOTE',[thisATRien, '@']);
        CallV('ORATR SET ATR PROCESSING LEVEL',[thisATRien, 1]);
      end;
    end;
    slATRPtList.Free;
    //TFF - added Notifications.Clear because processing a SMART alert would
cause all other
    // notes for that patient to be created with the SMART alert note values.
    Notifications.Clear;
  end;
  //ATR end
```

2. TFrmDrawers.InsertText procedure change

Purpose:
There was no reason to support to have a new InsertText procedure for ATR when the only change it offered was providing a different input value to use when creating the TTemplate for ATR.

Unit: fDrawers.pas

```
procedure TfrmDrawers.InsertText;
var
  BeforeLine, AfterTop: integer;
  txt, DocInfo: string;
```

```
  Template: TTemplate;

begin
  DocInfo := '';

  //ATR - TFF
  if InsertOK(TRUE) then
  begin
    if ((bATRAval) and (Notifications.Active) and (Notifications.FollowUp =
NF_ATR)) then
    begin
      //Template := TTemplate.Create(GetATRTemplate(ATRType));
      //TFF - the input parameter wasn't being used
      Template := TTemplate.Create(GetATRTemplate);
    end else
    if dmodShared.TemplateOK(tvTemplates.Selected.Data) then
    begin
      //if InsertOK(TRUE) and
(dmodShared.TemplateOK(tvTemplates.Selected.Data)) then
      //begin
      Template := TTemplate(tvTemplates.Selected.Data);
    end;
    if Template.ID <> '' then
    begin
    //ATR - TFF
      Template.TemplatePreviewMode := FALSE;
      if Template.IsReminderDialog then
        Template.ExecuteReminderDialog(TForm(Owner))
      else
      begin
        if Template.IsCOMObject then
          txt := Template.COMObjectText('', DocInfo)
        else
          txt := Template.Text;
        if(txt <> '') then
        begin
          CheckBoilerplate4Fields(txt, 'Template: ' + Template.PrintName);
          if txt <> '' then
          begin
            BeforeLine := SendMessage(FRichEditControl.Handle,
EM_EXLINEFROMCHAR, 0, FRichEditControl.SelStart);
            FRichEditControl.SelText := txt;
            FRichEditControl.SetFocus;
            SendMessage(FRichEditControl.Handle, EM_SCROLLCARET, 0, 0);
            AfterTop := SendMessage(FRichEditControl.Handle,
EM_GETFIRSTVISIBLELINE, 0, 0);
            SendMessage(FRichEditControl.Handle, EM_LINESCROLL, 0, -1 *
(AfterTop - BeforeLine));
            SpeakTextInserted;
          end;
        end;
      end;
    end;
  end;
end;
```

3. GetATRTemplate function change

Purpose:
The input parameter wasn't being used.

Unit: rTemplates.pas

For all cases of GetATRTemplate the function was change to have no input parameters. This includes:

- rTemplates declaration
- The function itself
- fDrawers.pas TfrmDrawers.InsertText Template := TTemplate.Create(GetATRTemplate);

4. GetATRTemplate function change

Purpose:
Replaced unnecessary code using CallV with sCallV.

Unit: rTemplates.pas

```
//ATR
//function GetATRTemplate(ATRName: string): string;
//TFF - the input parameter wasn't being used
function GetATRTemplate: string;
var
 I: Integer;
 thisTemplateIEN: string;
 DlgIEN: string;
begin
  if bATRAval then
  begin
    CallV('ORATR PATIENT',[Patient.DFN]);
    for I := 0 to RPCBrokerV.Results.Count - 1 do
    begin
      if uATR.ATR_ien = Piece(Piece(RPCBrokerV.Results[I],'^',8),';',7) then
      begin
        thisTemplateIEN := Piece(Piece(RPCBrokerV.Results[I],'^',8),';',5);
        Break;
      end;
    end;

    DlgIEN := Piece(sCallV('ORATR GET NOTE',[thisTemplateIEN]),'^',3);
    //DlgIEN := Piece(RPCBrokerV.Results[0],'^',3); //339 or 423   //TFF

    //examp. output from ORATR GET TIU TEMPLATE
    //Result - 2049^T^A^ATR BI-RADS FOLLOW-UP^0^^^0^0^0^0^0^0^0^0^339^DSSVET-
ATR BI-RAD SELECTOR^0^^^22;TIU(8925.1,^^^^0
    Result := sCallV('ORATR GET TIU TEMPLATE',[DlgIEN]);
    //TFF - switched the broker call above to sCallV instead of CallV
    //result := RPCBrokerV.Results[0];
```

```
  end;
end;
//ATR
```

5. TfrmRemDlg.btnFinishClick procedure change

Purpose:
The ATR block was moved from the top to the procedure to be within the if Process statement to only process SMART on successful processing of the Reminder Dialog.

Unit: fReminderDialog.pas

```
        if (Process) then
        begin

        //TFF - moved ATR block from the start to occur only if sccessful
processing
        //ATR
        uATR.TARHealthFactors := ''; //ATR - init - See procedure
TPCEData.SetHealthFactors()

        //Mark the TAR alert as 'Complete'?
          if bATRAval then
            begin
            if (bNotATRPt = false) then    //<=================== ORIG
              begin
              if not cbTAR.Checked then
                begin
                //MessageDlg('An exception has occurred in GetATRPatient()',
mtError, [mbOk], 0);
                TARResponse := MessageDlg('The checkbox, ''System for
Mammogram Results Tracking (SMART) Alert is complete'' is not checked.' +
#13#10 +
                                          'Click ''Yes'' to mark this alert
as complete.  Otherwise, click ''No''', mtConfirmation, [mbYes,mbNo], 0);
                if TARResponse = mrYes then
                  cbTAR.Checked := true;
                end;
              end;

            if (bNotATRPt = false) then
              begin
              if cbTar.Checked then
                begin
                CallV('ORATR SET TIU NOTE',[uATR.ATR_ien,
RemForm.PCEObj.NoteIEN]);
                CallV('ORATR SET ATR PROCESSING LEVEL',[uATR.ATR_ien, 2]);
//Complete - Ready for transmit
                end
              else
                CallV('ORATR SET ATR PROCESSING LEVEL',[uATR.ATR_ien, 1]);
//Incomplete - Some action taken
              end;
```

```
        end;
    //ATR
```

6. uATR.pas change

Purpose:
When the TAR.dll was not present slPtList would cause an error – having the unit create the expected TStringlist (and free it) corrected the error.

Unit: uATR.pas

```
//TFF - added initialization and finalization of slPtList - was causing an
error when the dll
// was not present within fPtSel procedure TfrmPtSel.cmdOKClick(Sender:
TObject);
initialization
  slPtList := TStringList.Create;

finalization
  slPtList.Free;

end.
```

TIU

1. TFrmNotes, TfrmDCSum, TfrmConsults, and TfrmSurgery Class Changes

Purpose:
Run a template not linked to a TIU Document Title.

Unit: fNotes.pas, fDCSumm.pas, fConsults.pas, fSurgery.pas

Property EditingIndex moved from private to public.
FEditNote moved from private to public.

2. uTemplates References Added

Purpose:
Run a template not linked to a TIU Document Title.

Unit: uTemplates.pas

The following references were added to the implementation uses clause:
fFrame, fNotes, fConsults, fDCSumm, fSurgery

3. TTemplate.COMObjectText function Change

Purpose:
Run a template not linked to a TIU Document Title.

Unit: uTemplates.pas

```
//TFF - DSIO

//function TTemplate.COMObjectText(const DefText: string = ''; DocInfo:
string = ''): string;
//function TTemplate.COMObjectText(DefText: string; var DocInfo: string):
string;
//var
//  p2: string;
//
//begin
//  Result := '';
//  if (FCOMObject > 0) then
//  begin
//    p2 := '';
//    if (LinkType <> ltNone) and (LinkIEN <> '') then
//      p2 := LinkPassCode[LinkType] + '=' + LinkIEN;
//    Result := DefText;
//    GetCOMObjectText(FCOMObject, p2, FCOMParam, Result, DocInfo);
```

```
//  end;
//end;

function TTemplate.COMObjectText(DefText: string; var DocInfo: string):
string;
var
  p2: string;
  tmpPtr : integer;
  iCurTabIndex : integer;
  TIUNoteIEN : string;
begin
  Result := '';
  if (FCOMObject > 0) then
  begin
    // MBH TF JAH LM  add DocInfo pass for unlinked COM objects start change
    if DocInfo = '' then
    begin
      iCurTabIndex := frmFrame.tabPage.TabIndex;
      if (icurTabIndex = frmframe.PageIDToTab(CT_NOTES)) then
      begin
        tmpPtr := frmNotes.lstNotes.ItemIndex;
        TIUNoteIEN := Piece(frmNotes.lstNotes.Items[tmpPtr],U,1);
        DocInfo := MakeXMLParamTIU(TIUnoteIEN, frmnotes.FEditNote);
      end
      else if (icurTabIndex = frmframe.PageIDToTab(CT_CONSULTS)) then
      begin
        tmpPtr := frmConsults.lstNotes.ItemIndex;
        TIUNoteIEN := Piece(frmConsults.lstNotes.Items[tmpPtr],U,1);
        DocInfo := MakeXMLParamTIU(TIUnoteIEN, frmConsults.FEditNote);
      end
      else if (icurTabIndex = frmframe.PageIDToTab(CT_DCSUMM)) then
      begin
        tmpPtr := frmDCSumm.lstSumms.ItemIndex;
        TIUNoteIEN := Piece(frmDCSumm.lstSumms.Items[tmpPtr],U,1);
        DocInfo := MakeXMLParamTIU(TIUnoteIEN, frmDCSumm.FEditDCSumm);
      end
      else if (icurTabIndex = frmframe.PageIDToTab(CT_SURGERY)) then
      begin
        tmpPtr := frmSurgery.lstNotes.ItemIndex;
        TIUNoteIEN := Piece(frmSurgery.lstNotes.Items[tmpPtr],U,1);
        DocInfo := MakeXMLParamTIU(TIUnoteIEN, frmSurgery.FEditNote);
      end;
    end;
    // MBH TF JAH LM end change
    p2 := '';
    if (LinkType <> ltNone) and (LinkIEN <> '') then
      p2 := LinkPassCode[LinkType] + '=' + LinkIEN;
    Result := DefText;
    GetCOMObjectText(FCOMObject, p2, FCOMParam, Result, DocInfo);
  end;
end;
//TFF - DSIO
```

Discreet Data

1. TfrmRemDlg.btnFinishClick procedure change

Purpose:
This checks if the Reminder Dialog being processed is being controlled (within the DSIO NOTE CONTROL file) and if so submits the reminder to the SubmitDiscreetData procedure.

Unit: fReminderDialog.pas

Within the if Process statement…

```
        //TFF - DSIO save off discreet data
        try
          if sCallV('DSIO IS IT CONTROLLED?', [Rem.IEN + ';PXRMD(801.41,']) =
'1' then
          begin
            if Rem.Processing and assigned(Rem.Elements) then
            SubmitDiscreetData(Rem, RemForm.NoteList.ItemIEN);
          end;
        except
        end;
```

2. uReminders.pas additions

Purpose:
To process all components of a Reminder Dialog and build an array to be sent over to VistA using the RPC – DSIO OCNT STORE to be stored discreetly with the DSIO NOTE DATA, DSIO NOTE S, DSIO NOTE M, and DSIO NOTE WP files.

Unit: uReminders.pas

```
//TFF - DSIO Discreet Data ------------------------------------------------
---

function SetRemPrompt(AObject: TRemPrompt): TStringList;
var
  Fld: string;

  function BuildDiscreetData(Sender: TObject; Fld,Seg: string): TStringList;
  var
    I,x: Integer;
  begin
    Result := TStringList.Create;
    Result.Clear;

    try
      if Sender is TCPRSDialogStaticLabel then
      begin
```

```
      end
      else if Sender is TCPRSDialogParentCheckBox then
      begin
        if TCPRSDialogParentCheckBox(Sender).Checked then
        Result.Add('S^' + Fld + TCPRSDialogParentCheckBox(Sender).Caption + U
+ Seg)
        else if not TCPRSDialogParentCheckBox(Sender).Checked then
        Result.Add('S^' + Fld + U + Seg);
      end
      else if Sender is TCPRSDialogFieldEdit then
      begin
        Result.Add('S^' + Fld + TCPRSDialogFieldEdit(Sender).Text + U + Seg);
      end
      else if Sender is TCPRSDialogComboBox then
      begin
        Result.Add('S^' + Fld + TCPRSDialogComboBox(Sender).Text + U + Seg);
      end
      else if Sender is TCPRSDialogCheckBox then
      begin
        if TCPRSDialogCheckBox(Sender).Checked then
        Result.Add('S^' + Fld + TCPRSDialogCheckBox(Sender).Caption + U +
Seg)
        else if not TCPRSDialogCheckBox(Sender).Checked then
        Result.Add('S^' + Fld + U + Seg);
      end
      else if Sender is TCPRSDialogButton then
      begin
      end
      else if Sender is TCPRSDialogYearEdit then
      begin
      end
      else if Sender is TCPRSDialogDateCombo then
      begin
        Result.Add('S^' + Fld + TCPRSDialogDateCombo(Sender).Text + U + Seg);
      end
      else if Sender is TCPRSDialogDateBox then
      begin
      end
      else if Sender is TCPRSNumberField then
      begin
        Result.Add('S^' + Fld + TCPRSNumberField(Sender).Text + U + Seg);
      end
      else if Sender is TCPRSDialogNumber then
      begin
      end
      else if Sender is TCPRSTemplateFieldLabel then
      begin
      end
      else if Sender is TCPRSDialogHyperlinkLabel then
      begin
      end
      else if Sender is TCPRSDialogRichEdit then
      begin
        for x := 0 to TCPRSDialogRichEdit(Sender).Lines.Count - 1 do
        Result.Add('WP^' + Fld +
TCPRSDialogRichEdit(Sender).Lines.Strings[x]);
```

```
        Result.Add('WP^' + Fld + TCPRSDialogRichEdit(Sender).Lines.Strings[x]
+ U + Seg);
      end
      else if Sender is TDlgFieldPanel then
      begin
      end
      else if Sender is TVitalComboBox then
      begin
        Result.Add('S^' + Fld + TVitalComboBox(Sender).Text + U + Seg);
      end
      else if Sender is TVitalEdit then
      begin
        Result.Add('S^' + Fld + TVitalEdit(Sender).Text + U + Seg);
      end
      else if Sender is TVitalComboBox then
      begin
        Result.Add('S^' + Fld + TVitalComboBox(Sender).Text + U + Seg);
      end;
    except
    end;
  end;

begin
  Result := TStringList.Create;
  if not Assigned(AObject.FCurrentControl) then Exit;

  Fld := TWinControl(AObject.FCurrentControl).ClassName + '|' +
         Piece(AObject.FRec4,U,2) + ',' + Piece(AObject.FRec4,U,1) + U +
         Piece(AObject.FRec4,U,8) + U;
  Result := BuildDiscreetData(AObject.FCurrentControl, Fld, AObject.NoteText)
end;

//    FChildren: TList;  // Points to other TRemDlgElement objects
//    FData: TList;      // List of TRemData objects
//    FPrompts: TList;   // list of TRemPrompt objects

function SetRemData(AObject: TRemData): string;
begin
  Result := '';
end;

function SetRemDlgElement(AObject: TRemDlgElement): TStringList;
var
  I: Integer;
  sl: TStringList;
begin
  Result := TStringList.Create;

  if AObject.FChildren <> nil then
  if AObject.FChildren.Count > 0 then
  begin
    for I := 0 to AObject.FChildren.Count - 1 do
    begin
      sl := TStringList.Create;
      try
        sl := SetRemDlgElement(TRemDlgElement(AObject.FChildren[I]));
        Result.AddStrings(sl);
```

```
      finally
        sl.Free;
      end;
    end;
  end;

//  if AObject.FData <> nil then
//  if AObject.FData.Count > 0 then
//  begin
//    for I := 0 to AObject.FData.Count - 1 do
//    Result.Add(SetRemData(TRemData(AObject.FData[I])));
//  end;

  if AObject.FPrompts <> nil then
  if AObject.FPrompts.Count > 0 then
  begin
    for I := 0 to AObject.FPrompts.Count - 1 do
    begin
      sl := TStringList.Create;
      try
        sl := SetRemPrompt(TRemPrompt(AObject.FPrompts[I]));
        Result.AddStrings(sl);
      finally
        sl.Free;
      end;
    end;
  end;
end;

{
 ; EXAMPLE ARRAY:
 ; --------------
 ; S^CONTROL^VISTA LABEL 1^SOME VALUE
 ; S^CONTROL^VISTA LABEL 2^SOME OTHER VALUE
 ; M^CONTROL^VISTA LABEL^VALUE 1^INDEX
 ; M^CONTROL^VISTA LABEL^VALUE 2^INDEX
 ; WP^CONTROL^VISTA LABEL^THIS TYPE IS USED MORE FOR A FIELD THAT
 ; WP^CONTROL^VISTA LABEL^CONTAINS A LOT OF TEXT THAT'S MEANT TO BE READ
 ; WP^CONTROL^VISTA LABEL^TOGETHER.
}

procedure SubmitDiscreetData(Rem: TReminderDialog; NoteIEN: Integer);
var
  I: Integer;
  Note,Build: TStringList;
begin
  Note := TStringList.Create;
  try
    for I := 0 to Rem.Elements.Count - 1 do
    begin
      Build := TStringList.Create;

      if Rem.Elements.Objects[I] is TRemDlgElement then
      Build := SetRemDlgElement(TRemDlgElement(Rem.Elements.Objects[I]));

      if Build.Count > 0 then
      Note.AddStrings(Build);
```

```
      Build.Free;
    end;

    if Note.Count > 0 then
    begin
      try
        CallV('DSIO OCNT STORE', [NoteIEN, Note, 'R']);
      except
      end;
    end;
  finally
    Note.Free;
  end;
end;

//TFF - DSIO Discreet Data ------------------------------------------------
---
```