

DDCS FORMS (Delphi XE10)

To install the DDCSFramework from source load the project DDCSFramework.dproj and then in the project manager you can right click the package name "DDCSFramework.bpl" and select install.

To load from DDCSFramework.bpl you can select the "Component" option "Install Packages" then select ADD and navigate to the .bpl and open.

Be sure to add the DDCSForm directory to your Library. Do this by going to "Tools" "Options" and under "Delphi Options" select "Library". Once here go to "Library path" and add the full path (including DDCSForm) by using the "..." button to the right, then the folder icon to select the path and "Add".

You should now have the category "DDCSForm" in your tool palette with TDDCSForm as an entry.

Creating a DDCS Form (as a CPRS Note Extension) DLL

1. In Delphi (XE10) create a New Project - Delphi Projects - Dynamic-link Library.
2. Right click the project name (Project1.dll) from the project manager and select "Add New" - "VCL Form". I changed the name of the form unit to frmMain. The source should now look like this...

```
library Project1;

{ Important note about DLL memory management: ShareMem must be the
  first unit in your library's USES clause AND your project's (select
  Project-View Source) USES clause if your DLL exports any procedures or
  functions that pass strings as parameters or function results. This
  applies to all strings passed to and from your DLL--even those that
  are nested in records and classes. ShareMem is the interface unit to
  the BORLNDMM.DLL shared memory manager, which must be deployed along
  with your DLL. To avoid using BORLNDMM.DLL, pass string information
  using PChar or ShortString parameters. }

uses
  System.SysUtils,
  System.Classes,
  frmMain in 'frmMain.pas' {Form1};

{$R *.res}

begin
end.
```

3. In your VCL Form unit go to your tool pallet and look for TDDCSForm (it's in the DDCSForm category) and add it to your form. Make sure you add at least one page (right click the body of the component and select "New Page") before you add anything else. All components must be placed on a page in order to be picked up by DDCS.
4. Now lets go back to the source and make it look like the following...

```
library Project1;

{$R *.dres}

uses
  Winapi.Windows,
  Vcl.Forms,
  uExtndComBroker,
  frmMain in 'frmMain.pas' {Form1};

{$R *.res}

function Launch(const CPRSBroker: PCPRSCoMBroker; out Return: WideString): WordBo
var
  oldHandle: HWND;
begin
  Result := False;

  oldHandle := Application.Handle;
  Application.Handle := GetActiveWindow;

  RPCBrokerV := CPRSBroker^;
  Form1 := TForm1.Create(nil);
  try
    RPCBrokerV.DisabledWindow := DisableTaskWindows(0);
    Form1.ShowModal;
    if Form1.DDCSForm1.Validated then
    begin
      Result := True;
      Return := Form1.DDCSForm1.TmpStrList.Text;
    end;
  finally
    Form1.Free;
    RPCBrokerV := nil;
    Application.Handle := oldHandle;
  end;
end;

exports
  Launch;

begin
end.
```



Adding Themes

Add the Windows10 theme to your project. This step is optional and at the time of this writing the component's option to change the theme was disabled but if you wish to use it your project will also have to have the appropriate themes added as a resource.

Select the "Project" option and under it select "Resources and Images...". Then select "Add..." and navigate to the theme .vsf file resource you wish to add (mine were located at C:\Users\Public\Documents\Embarcadero\Studio\17.0\Styles). Once added, set the type to VCLSTYLE and set the identifier to Resource_1 for the first one and just increment the number for each additional entry.

Resource Theme

Adding Components to the Form

Now that we have the base form built we'll want to add some components to it - make sure to add components only to a TDDCSForm TTabSheet.

Supported Components are those that inherit from...

- TORDateBox
- TORCheckBox
- TORComboBox
- TORListBox
- TORListView
- TStaticText
- TDateTimePicker
- TCustomMemo
- TCustomEdit
- TCheckBox
- TRadioButton
- TRadioGroup
- TCustomCombo
- TCustomListBox
- TListView
- TStringGrid

Form with Components

Form with Components

When you add a component to the TabSheet it will be added to your DDCSForm ReportCollection (the ReportCollection is used to build the note).

ReportCollection

ReportCollection

For the ReportItem for your newly added component you can set the following fields and this is what they do/are for.

DialogReturn

When the component is set up to list Dialogs (forms stored in the DDCSDialogs.dll) the return of the dialog must be stored in another component that's inherited from a TCustomMemo.

DoNotRestoreV

This value must be updated to VistA in order for it to be acted on. When the VistA configuration record is updated each time the form is created it will not recall any previously saved information other than configuration values.

DoNotSave

At the GUI level this component will not send any information back to VistA.

DoNotSpace

For the generation of the note there will not be a space added to the output of this component.

HideFromNote

This component will not output to the note.

IdentifyingName

If validation fails (if this component is required but not valued) then this is the name that is displayed to the user.

Order

When the note is generated the note is built as a list of components and it is done so in this order. In order to change the order use the up and down arrows within in the ReportCollection in the project structure.

OwningObject

This is the component that this ReportItem extends.

Prefix

When generating the note this value is printed before the component value. If the component inherits from a TCustomMemo each line will be prefixed.

Required

If TRUE then when the user Finishes the form this component must be valued.

SayOnFocus

This is the string that JAWS will say when the component gains focus before carrying on with any of its defaulted behavior.

Suffix

When generating a note this value is print after the component value. If the component inherits from TCustomMemo each line will be suffixed.

Title

When generating the note this string will be printed above the component value.

Form Properties

The TDDCSForm inherits from TPageControl so anything you can do with a PageControl you can also do with a DDCS form with the addition to the following...

However, do **NOT** change the following properties...

Property	Value
Align	alClient
TabStop	False
TabOrder	0
MultiLine	tsButtons
Style	tpTop
OwnerDraw	True

And do **NOT** set the following methods...

OnDrawTab = DrawCheckTab

Published Properties

AutoTimer

If you wish to override the existing timer that handles the auto save of ReportItems to VistA then you can select a new TTimer.

DisableSplash

When the form is displayed a splash screen is shown, if this is TRUE then it will not be shown.

ReportCollection

This is the form's collection of components that will be reported on. Only this components will be

checked for validity (if marked as TRUE for the required filed) and will be added to the generated note. Furthermore, only these components are saved to VistA as discreet data.

```
procedure DeleteNoteItem(Value: TWinControl);
```

Pass in the component to delete the ReportItem.

```
function GetNoteItemAddifNil(Value: TWinControl): TDDCSNoteItem;
```

Pass in the component to return the ReportItem and if it doesn't exist then it will create it and return it.

```
function GetNoteItem(Value: TWinControl): TDDCSNoteItem;
```

Pass in the component to return the ReportItem or nil if it doesn't exist.

```
function GetAControl(Value: string): TWinControl;
```

Pass in the name of the component to return that component.

```
function Add: TDDCSNoteItem; overload;
```

Don't use this.

```
function Insert**(Index: Integer): TDDCSNoteItem;
```

Don't use this.

```
property Items[Index: Integer]: TDDCSNoteItem read GetItem write SetItem;
```

Don't use this.

VitalsPage

Assigning a TTabSheet as the VitalsPage means to embed that tab with the TDDCSVitals frame. The embedded TabSheet must not have any other component on it and only one VitalsPage can exists for any one DDCS Form. If the value is changed the previous page has this Vitals component removed.

Published Events

OnAccept

After "Finish" is clicked and the preview is shown and closed and the ReportItems are saved this event will be called.

OnFinish

This event is called when the "Finish" button is clicked before the preview is shown.

OnOverrideNote

When the note is shown either through "Preview" or "Finish" and after the the ReportItems have been validated, if validation passes then this event will be called to populate the note rather than using the ReportItems to populate the note.

OnRestore

Currently not active.

OnSave

This event is called when the user clicks the "Save" button and when the auto save triggers.

Public Properties

MultilInterface

This property (which is also a field in the file #19641.42), when set to YES each page in the configuration will be understood treated as an individual program so that each page can be shown alone (other pages will be hidden). However, multiples pages can still be shown simintaneously.

TmpStrList

This property is the TStringList that is populated and sent from TDDCSForm back to the COMObject to be sent over to CPRS. Populate this property only for use of creating an OnOverrideNote method.

Validated

This property is used when the OnOverrideNote method is used, set this property if you are doing your own validation.

VitalsControl

This gives you direct access to the TDDCSVitals component that is embedded into a TTabSheet when the VitalsPage property is set.

ScreenReader

This procedure allows you to associate a sting to a control so that when the control gains focus JAWS will say that string. Use this procedure in the FormCreate event.

CurrentStyle

This property would be used by the menu option via the command menu to change the form style which is currently set to "Default". When this property is set it will pass the style over to DisplayDialog so that

the program is visually uniform. However, that dialog must then have that as a resource - see "Adding Themes".

Configuration

This property is another TOwnedCollection like ReportCollection except it is designed to hold large records imported from VistA that can be populated into many controls on the form and then later updated upon completion of the form and passed back to VistA.

Public Procedures

```
procedure cbAutoWidth(Sender: TObject);
```

This is a helper method that can be assigned to TComboBox.OnDropDown. It will widen the drop down pane to the longest entry in the list.

```
procedure RadioGroupEnter(Sender: TObject);
```

This is a helper method that can be assigned to TRadioGroup.OnEnter. It will focus the first TRadiobutton on tab entry.

```
procedure GetPatientAllergies(var oText: TStringList);
```

This method will call VistA to return a text table to report the patient's allergies.

```
procedure GetPatientActiveProblems(var oText: TStringList);
```

This method will call VistA to return a text table to report the patient's active problems.

```
procedure GetPatientActiveMedications(var oText: TStringList);
```

This method will call VistA to return a text table to report the patient's active medications.

```
procedure LoadDialogs;
```

This method will reload the DDCSDialogs.dll which is also accomplished by the user via the command menu.

Public Functions


```
function HasSecurityKey(const KeyName: string): Boolean;
```

Pass in the security key name and VistA will return True/False whether or not the user running the program has said key.

Method Pointers

```
DisplayDialog: TDisplayDialog;  
  
    TDisplayDialog = function(const Broker: PCPRComBroker; dlgName: WideString;  
                                out rSave, rConfig, rText: Boolean) Boolean;  
  
    if Assigned(DisplayDialog) then  
        if DisplayDialog(@RPCBrokerV, sBuild, False, CurrentStyle, wSave, wConfig, rText) then
```

DisplayDialog can be called as follows but is used internally for components configured with dialog entries.

```
GetDialogComponents: TGetDialogComponents;  
  
    TGetDialogComponents = procedure(dlgName: WideString; out Return: WideString;  
                                     out rSave, rConfig, rText: Boolean) Boolean;  
  
    if Assigned(FDDCSForm.GetDialogComponents) then  
        FDDCSForm.GetDialogComponents(lvDialog.Items[lvDialog.ItemIndex].Caption, wSave, wConfig, rText)
```

GetDialogComponents can be called as follows but is used internally for the GUI side configuration.

VitalsPage

The TDDCSVitals frame is embedded in a TTabSheet of a TDDCSForm component. It will show one to three pages and is itself a PageControl.

The following are its pages...

1. **Vitals**

Vitals Page One

2. **Estimated Delivery Date**

Vitals Page Two

This page is only visible for female patients.

3. Menstrual History

Vitals Page Three

This page is only visible for female patients.

Public Procedures

```
procedure Save;
```

This procedure is called when the user clicks the save button located on the command bar or when the form is closed.

```
procedure GetPatientVitals(var oText: TStringList);
```

This procedure is called when the note for preview is generated to display the first page of the vitals component.

```
procedure GetEDDNote(var oText: TStringList);
```

This procedure is called when the note for preview is generated to display the second page of the vitals component (only female patients).

```
procedure GetLMPNote(var oText: TStringList);
```

This procedure is called when the note for preview is generated to display the third page of the vitals component (only female patients).

Public Functions

```
function GetTextforFocus(Value: TWinControl): string;
```

When the ActiveControl changes (the user has changed focus from one TWinControl to another) this function is called and if it comes back with a string and the ScreenReader (JAWS COMObject) is not nil then the ScreenReader Say method is called with it.

Public Properties

```
property Vitals: TDDCSVitals ...
```

Direct access to the vitals component if it has been assigned to a TTabSheet.

DDCSDialogs.DLL

The DDCSDialogs.dll is a collection of forms that inherit from TDDCSDialog and they can range from the very simple to the very complex.

Adding a new dialog to this DLL is as simple as...

1. Add a new VCL Form to the project.
2. Name your unit with a prefix of "udlg" and your form with a prefix of "dlg".
3. Right Click the project "DDCSDialogs.dll" and select "View Source". In here add your new form to the RegisterDialogs procedure as follows.

```
RegisterClass(TdlgMyNewDialog); sl.Add('TdlgMyNewDialog^udlgMyNewDialog');
```

If a dialog is not within this RegisterClass procedure then it will not be recognized by TDDCSForm.

Public Procedures

SayOnFocus(wControl: TWinControl; tSay: string);

This procedure allows you to associate a sting to a control so that when the control gains focus JAWS will say that string. Use this procedure in the FormCreate event.

Example Use

```
procedure TdlgMyNewDialog.FormCreate(Sender: TObject);
begin
    SayOnFocus(cbSOBY, 'Shortness of breath?');
    SayOnFocus(cbSOBN, 'Shortness of breath?');
end;
```

When the user focuses on cbSOBY JAWS will say "Shortness of breath?".

BuildSaveList(var oText: TStringList);

This method is used collect all the values of all the ReportItems (which is all the components for the dialog unless you added code to delete some) which is called in the DisplayDialog method when the

dialog is closed and the ModalResult is mrOK to send back to the caller of DisplayDialog as the rSave WideString which (if called by the methods to display dialogs from component lists) is saved in VistA as the components discreet data for that dialog for that TDDCSForm for that note.

This TStringList is constructed with each line representing a component and if that component has multiple values then it would be duplicated in the list with the value piece updated to reflect the additional entries.

[0]=COMPONENT_NAME^INDEX^VALUE *If no INDEX then that piece is just left empty. If the component is a TCheckbox or TChecklistbox then the INDEX may also contain the literal TRUE if checked.*

```
| +-- _Note
|   +-- _TDDCSForm
|       +-- Component - Data
|       +-- Component - Data
|       +-- _Dialog
|           +-- Component - Data
|           +-- Component - Data
```

Public Properties

TmpStrList

This property is really the whole point to the dialog. The point of the dialog is to share a common form among DDCS Forms which produce a block of text to be added to the DDCS Form note.

Example Use

```
procedure TdlgMyNewDialog.bbbtnOKClick(Sender: TObject);
begin
    if leOnset.Text <> '' then
        TmpStrList.Add(' Onset: ' + leOnset.Text);
    if cbSOBY.Checked then
        TmpStrList.Add(' Shortness of breath?: Yes');
    if cbSOBN.Checked then
        TmpStrList.Add(' Shortness of breath?: No');
    if leDur.Text <> '' then
        TmpStrList.Add(' Duration: ' + leDur.Text);
    if leAssocSym.Text <> '' then
        begin
            TmpStrList.Add(' Associated Symptoms:');
            TmpStrList.Add(' ' + leAssocSym.Text);
        end;

    if TmpStrList.Count > 0 then
```

```
    TmpStrList.Insert(0, 'MyNewDialog:');  
end;
```

Configuration

This property is another TOwnedCollection like ReportCollection except it is designed to hold large records imported from VistA that can be populated into many controls on the form and then later updated upon completion of the form and passed back to VistA.

Example

The TdlgPregHist utilizes this collection so that it can build many pages of historical pregnancy data. The user is also able to change and add to this data which means that this collection needs to be updated as well.

```
procedure TdlgPregHist.FormShow(Sender: TObject);  
var  
    I,G,J,L: Integer;  
    vPreg: TfPreg;  
    vPregInfo: TfPregInfo;  
    vPregChild: TfChild;  
    cItem: TConfigItem;  
    tmp,btmp,sLkup: string;  
begin  
  
    //  1) L  
    //  2) IEN  
    //  3) DATE RECORDED  
    //  4) EDC  
    //  5) DFN|PATIENT  
    //  6) STATUS  
    //  7) FOF|(IEN OR IDENTIFIER)  
    //  8) EDD  
    //  9) PREGNANCY END  
    // 10) OB IEN|OB  
    // 11) FACILITY IEN|FACILITY  
    // 12) UPDATED BY IEN|UPDATED BY  
    // 13) GESTATIONAL AGE  
    // 14) LENGTH OF LABOR  
    // 15) TYPE OF DELIVERY  
    // 16) ANESTHESIA  
    // 17) PRETERM DELIVERY  
    // 18) BIRTH TYPE  
    // 19) IEN;NUMBER;NAME;GENDER;BIRTH WEIGHT;STILLBORN;APGAR1;APGAR2;STATUS;NICU|  
    // 20) OUTCOME  
    // 21) HIGH RISK FLAG  
    // 22) DAYS IN HOSPITAL  
    //  
    // C^IEN^COMMENT  
    // B^IEN|BABY|^COMMENT
```

```

if Configuration.Count > 0 then
begin
  for I := 0 to Configuration.Count - 1 do
  begin
    cItem := Configuration.Items[I];

    if cItem.ID[1] = 'L' then
    begin
      // ---- Add the Pregnancy Tab -----
      tmp := Uppercase(cItem.Piece[20]);    // Outcome
      if tmp = 'ECTOPIC' then
        edtEctopic.Value := edtEctopic.Value + 1
      else if tmp = 'TERMINATION' then
        edtAbInduced.Value := edtAbInduced.Value + 1
      else if tmp = 'SPONTANEOUS ABORTION' then
        edtAbSpont.Value := edtAbSpont.Value + 1
      else
        edtTotPreg.Value := edtTotPreg.Value + 1;
      // -----

      // ---- Get the Pregnancy Info Form -----
      if pgPregnancy.Pages[pgPregnancy.PageCount - 1].ControlCount > 0 then
        if pgPregnancy.Pages[pgPregnancy.PageCount - 1].Controls[0] is TfPreg then
          begin
            vPreg := TfPreg(pgPregnancy.Pages[pgPregnancy.PageCount - 1].Controls[0]);

            if AnsiContainsText(cItem.Piece[2], '+') then
              vPreg.Added := True;
            vPreg.PregnancyIEN := StrToIntDef(cItem.Piece[2], 0);

            vPregInfo := vPreg.GetPregInfo;
            if vPregInfo <> nil then
              begin
                if cItem.Piece[6] = 'CURRENT' then
                  begin
                    vPregInfo.lbStatus.Visible := True;
                    if vPregInfo.lbStatus.Caption <> '' then
                      vPregInfo.lbStatus.Caption := vPregInfo.lbStatus.Caption + ' (C)'
                    else
                      vPregInfo.lbStatus.Caption := 'CURRENT';

                    vPregInfo.Disable := True;
                    vPreg.btnDelete.Enabled := False;
                  end;

                vPregInfo.dtDelivery.Text := cItem.Piece[9];

                tmp := cItem.Piece[11];
                if tmp <> '' then
                  begin
                    if vPregInfo.cbDeliveryPlace.Items.IndexOf(Piece(tmp,'|',2)) <> -1 then
                      vPregInfo.cbDeliveryPlace.ItemIndex := vPregInfo.cbDeliveryPlace.ItemIndex + 1;
                  end;
                end;
              end;
            end;
          end;
        end;
      end;
    end;
  end;
end;

```

```

        else
            vPregInfo.cbDeliveryPlace.Text := Piece(tmp, '|', 2);
        end;

tmp := cItem.Piece[13];
vPregInfo.spnGAWeeks.Value := StrToIntDef(Piece(tmp, 'W', 1), 0);
vPregInfo.spnGADays.Value := StrToIntDef(Piece(Piece(tmp, 'D', 1), 'W', 2), 0);

vPregInfo.spnLaborLength.Value := StrToIntDef(cItem.Piece[14], 0);

tmp := cItem.Piece[15];
if tmp = 'V' then
    vPregInfo.rgTypeDelivery.ItemIndex := 0
else if tmp = 'C' then
    vPregInfo.rgTypeDelivery.ItemIndex := 1;

tmp := cItem.Piece[16];
if tmp <> '' then
begin
    if vPregInfo.cbAnesthesia.Items.IndexOf(tmp) = -1 then
        vPregInfo.cbAnesthesia.Items.Add(tmp);
    vPregInfo.cbAnesthesia.ItemIndex := vPregInfo.cbAnesthesia.Items.IndexOf(tmp);
end;

vPregInfo.rgPretermDelivery.ItemIndex := StrToIntDef(cItem.Piece[17], 0);

if vPregInfo.cbOutcome.Enabled then
begin
    tmp := cItem.Piece[20];
    if tmp <> '' then
    begin
        if vPregInfo.cbOutcome.Items.IndexOf(tmp) = -1 then
            vPregInfo.cbOutcome.Items.Add(tmp);
        vPregInfo.cbOutcome.ItemIndex := vPregInfo.cbOutcome.Items.IndexOf(tmp);
    end;
end;

vPregInfo.edtDeliveryAt.Value := StrToIntDef(cItem.Piece[22], 0);

// IEN;NUMBER;NAME;GENDER;BIRTH WEIGHT;STILLBORN;APGAR1;APGAR2;STATUS;NI
tmp := cItem.Piece[19];
if tmp <> '' then
begin
    G := SubCount(tmp, '|') + 1;
    for J := 1 to G do
    begin
        btmp := Piece(tmp, '|', J);
        // ---- Add the Baby Tab -----
        vPregInfo.spnBirthCount.Value := vPregInfo.spnBirthCount.Value + 1;
        // -----

        // ---- Get the Baby Info Form -----

```

```

if vPreg.pgPreg.PageCount > 1 then
    if vPreg.pgPreg.Pages[vPreg.pgPreg.PageCount - 1].ControlCount > 0
        if vPreg.pgPreg.Pages[vPreg.pgPreg.PageCount - 1].Controls[0] is
begin
            vPregChild := TfChild(vPreg.pgPreg.Pages[vPreg.pgPreg.PageCoun

// IEN
vPregChild.BabyIEN := Piece(btmp, ';', 1);

// Baby #
vPregChild.BabyNumber := Piece(btmp, ';', 2);

// Sex
if Piece(btmp, ';', 4) = 'M' then
    vPregChild.rgSex.ItemIndex := 0
else if Piece(btmp, ';', 4) = 'F' then
    vPregChild.rgSex.ItemIndex := 1
else if Piece(btmp, ';', 4) = 'U' then
    vPregChild.rgSex.ItemIndex := 2;

// Weight
vPregChild.spnG.Value := StrToIntDef(Piece(btmp, ';', 5), 0);

// APGAR1
vPregChild.edAPGARone.Text := Piece(btmp, ';', 7);

// APGAR2
vPregChild.edAPGARfive.Text := Piece(btmp, ';', 8);

// NICU
vPregChild.ckNICU.Checked := (Piece(btmp, ';', 10) = '1');

// Baby Notes
if vPreg.Added then
    sLkup := '+' + IntToStr(vPreg.PregnancyIEN)
else
    sLkup := IntToStr(vPreg.PregnancyIEN);
sLkup := sLkup + '|' + vPregChild.BabyIEN + '|' + vPregChild.B

cItem := Configuration.LookUp('B', sLkup, '');
if cItem <> nil then
    for L := 0 to cItem.Data.Count - 1 do
        vPregChild.meComplications.Lines.Add(Pieces(cItem.Data[L],
end;
end;
end;

// Delivery Notes
if vPreg.Added then
    sLkup := '+' + IntToStr(vPreg.PregnancyIEN)
else
    sLkup := IntToStr(vPreg.PregnancyIEN);

```



```

        cItem := Configuration.LookUp('C', sLkup, '');
        if cItem <> nil then
            for J := 0 to cItem.Data.Count - 1 do
                vPregInfo.meDeliveryNotes.Lines.Add(Pieces(cItem.Data[J],U,3,999));
            end;
        end;
    end;
end;

if pgPregnancy.PageCount > 0 then
begin
    for I := pgPregnancy.PageCount - 1 downto 0 do
        if pgPregnancy.Pages[I].ControlCount > 0 then
            if pgPregnancy.Pages[I].Controls[0] is TfPreg then
                begin
                    vPreg := TfPreg(pgPregnancy.Pages[I].Controls[0]);
                    if vPreg.pgPreg.PageCount > 0 then
                        vPreg.pgPreg.ActivePageIndex := 0;
                    end;
                end;

                pgPregnancy.ActivePageIndex := 0;
            end;
        end;
    end;

procedure TdlgPregHist.btnOKClick(Sender: TObject);
var
    sl: TStringList;
    I,nPreg: Integer;
    vPreg: TfPreg;
    PregID: string;
    cItem: TConfigItem;
begin
    sl := TStringList.Create;
    try
        TmpStrList.Add('Pregnancy History: ');
        TmpStrList.Add('  Total Pregnancies: ' + edtTotPreg.Text);
        TmpStrList.Add('  Induced Abortion: ' + edtAbInduced.Text);
        TmpStrList.Add('  Spontaneous Abortion: ' + edtAbSpont.Text);
        TmpStrList.Add('  Ectopic: ' + edtEctopic.Text);

        nPreg := 0;
        for I := 0 to pgPregnancy.PageCount - 1 do
            if pgPregnancy.Pages[I].ControlCount > 0 then
                if pgPregnancy.Pages[I].Controls[0] is TfPreg then
                    begin
                        vPreg := TfPreg(pgPregnancy.Pages[I].Controls[0]);

                        vPreg.GetText(sl);
                        if sl.Count > 0 then
                            TmpStrList.AddStrings(sl);
                        end;
                    end;
                end;
            end;
        end;
    except
        sl.Free;
    end;
end;

```

```

sl.Clear;

inc(nPreg);
if vPreg.PregnancyIEN < 1 then
    PregID := '+' + IntToStr(nPreg)
else begin
    if vPreg.Added then
        PregID := '+' + IntToStr(vPreg.PregnancyIEN)
    else
        PregID := IntToStr(vPreg.PregnancyIEN);
end;

// Pregnancy Info
cItem := Configuration.LookUp('L', PregID, '');
if cItem = nil then
begin
    cItem := TConfigItem.Create(Configuration);
    cItem.ID[1] := 'L';
    cItem.ID[2] := PregID;
    cItem.Data.Add('');
end;
cItem.Data[0] := vPreg.GetSavePregInfo(PregID);

// Pregnancy Comments
cItem := Configuration.LookUp('C', PregID, '');
if cItem = nil then
begin
    cItem := TConfigItem.Create(Configuration);
    cItem.ID[1] := 'C';
    cItem.ID[2] := PregID;
end;
cItem.Data.Clear;
vPreg.GetSavePregComments(PregID, sl);
if sl.Count > 0 then
    cItem.Data.AddStrings(sl);
sl.Clear;

// Baby Comments
vPreg.GetSaveChildComments(PregID);
end;
finally
    sl.Free;
end;
end;

```

ScreenReader

If you need direct access to the screen reader then you have access to it through TDDCSDialog.

ScreenReader: IJawsApi

```

IJawsApi = interface(IDispatch)
['{123DEDB4-2CF6-429C-A2AB-CC809E5516CE}']
    function RunScript(const ScriptName: WideString): WordBool; safecall;
    function SayString(const StringToSpeak: WideString; bFlush: WordBool): WordBool; safecall;
    procedure StopSpeech; safecall;
    function Enable(vbNoDDIHooks: WordBool): WordBool; safecall;
    function Disable: WordBool; safecall;
    function RunFunction(const FunctionName: WideString): WordBool; safecall;
end;

```

Only SayString is recommend for use - all other methods should be avoided.

ReportCollection

This is just like the ReportCollection owned by TDDCSForm. When a dialog is created all components are added to the ReportCollection and they are saved to VistA in the same manner as well, so when the dialog is running you may manipulate these ReportItems as you see fit.

Configuring your Form

Before you are able to run your project you will first have to do some setup to link it to VistA.

Parameter

The DSIO DDCS LOCATION parameter is required as it and the configuration file needs to be able to build the absolute path that will allow COM to access and open the form dll. In this case if I am accessing the TIU Note Title that is under DDCS control (NURSE POSTPARTUM – MATERNAL) then the oCNT_NursePostpartumMaternal.dll must then be located where the parameter indicates so:
C:\Users\USERNAME\Desktop\DDCS_output\oCNT_NursePostpartumMaternal.dll

Set the LOCATION parameter to the location of the dlls.

```

Select PARAMETER DEFINITION NAME: DSIO DDCS LOCATION      DSIO DDCS LOCATION

----- Setting DSIO DDCS LOCATION for System: ... -----
LOCATION: C:\Users\USERNAME\Desktop\DDCS\_output\

```

Register COM Object

In Windows, open the command prompt as an administrator and navigate to the COM Object that is located on the workstation that is accessing the DDCS Form Templates/oCNTs. Once your terminal is pointing to the same directory in which your COM resides run the command “regsvr32

DDCSFormBuilder.dll" without quotes.

Image

If this is your first time using COM Object with CPRS then you may have to also register CPRS. To do this repeat the steps above but navigate to the location of the CPRS executable and run the command "CPRSChart /register" without quotes.

Both of the above commands just need to be run once on each of the machines accessing COM Objects for CPRS. If CPRS is on a server then it needs to be registered there; however, the DDCSFormBuilder.dll needs to be on each machine and registered there.

For more information check out the [VA Software Document Library](#).

Create CONTROL Entry

DSIO DDCS CONTROL FILE (#19641.4)

FILE STRUCTURE

FIELD NUMBER	FIELD NAME
.01	CONTROL OBJECT (RF), [0;1]
.02	INACTIVE (S), [0;2]
.03	DESTINATION FILE (P1'), [0;3]
.04	PATIENT ORIENTED (S), [0;4]
.05	PATIENT FIELD (FX), [0;5]
.06	SHARE (S), [0;6]
.07	FORM (P19641.42), [0;7]
1	PATIENT LOOKUP CODE (K), [1;E1,245]
2	TRIGGER LOGIC (F), [2;E1,245]
3	PRE PUSH (K), [3;E1,245]
4	POST PUSH (K), [4;E1,245]
99.1	PACKAGE (P9.4'), [99;1]

In order for the COMObject to display a DDCS Form it must have the CONTROL OBJECT in this file - we will be using TIU DOCUMENT DEFINITION (8925.1) for our example but it is not limited to TIU Note Titles.

In FileMan use the "ENTER OR EDIT FILE ENTRIES" option to edit the "DSIO DDCS CONTROL" (#19641.4) file.

VA FileMan 22.0

Select OPTION: 1 ENTER OR EDIT FILE ENTRIES

INPUT TO WHAT FILE: 19641.4 DSIO DDCS CONTROL
EDIT WHICH FIELD: ALL//

CONTROL OBJECT: 167;TIU(8925.1,//

This is the "object" you wish to have extended by this package (DSIO DDCS). Basically, what it comes down to is that you are collecting information and associating it to a record in VistA.

INACTIVE:

Set this field to "YES" if you wish to keep the entry but cease controlling it.

DESTINATION FILE: TIU DOCUMENT//

When you control an "object" you need to identify where the data is being stored so in the case of TIU that's 8925 so the data collected for this control will be associated to a variable pointer that will be "IEN;TIU(8925," where the IEN is created and passed over to DDCS via the GUI.

PATIENT ORIENTED: YES//

This will assist any back end code in retrieving the DFN if needed. In most cases it is and may not work if it cannot obtain a valid DFN. However, if the DESTINATION FILE is not patient oriented there is still an opportunity to provide M code that can obtain it.

PATIENT FIELD: .02//

This is paired up with the DESTINATION FILE if it is PATIENT ORIENTED.

SHARE:

If there is a need to accumulate data then SHARE would be set to "YES" to save all collected data to a patient centric file rather than the record generic one. Setting this field to "LIMITED" will allow some of the collected data to be stored directly under the patient which is handled by the configuration of the Form (#19641.42) or Dialog (#19641.49). An example of this action is having the field set to LIMITED and using the OB Flow Sheet dialog which shows all entries ever collected every time the dialog is opened no matter what note it was collected in.

FORM: NURSE POSTPARTUM - MATERNAL//

This is the form configuration for this controlled object. This entry will also state what program that this control shall execute.

PATIENT LOOKUP CODE:

If this controlled object's DESTINATION FILE is not PATIENT ORIENTED then you would have to supply the M code here that could be executed to retrieve the appropriate DFN.

TRIGGER LOGIC: I \$\$\$GET1^DIQ(DDCSFLE,SIEN_",",.05)="COMPLETED"

DSIO DDCS has an option that can be invoked (DSIO DDCS CHECK STATUS) that will check any entries that have not been "PUSHED" (which is a X-REF in #19641.41 that only exists if an entry has not been "PUSHED") and come back to it's CONTROL entry here and execute this M code to determine if it can "PUSH" the collected data (the collected data is not destroyed upon "PUSH").

PRE PUSH: N LNK S LNK=\$\$SPG^DSIO3(\$G(DFN),SIEN,\$\$PG^DSIO4(\$G(DFN)))

M code that is executed if the TRIGGER LOGIC passes but before any further action is taken to "PUSH" the collected data.

POST PUSH:

M code that is executed after the collected data is "PUSHED".

PACKAGE: TEXT INTEGRATION UTILITIES//

Used to identify the package this record is enhancing - also helpful when screening entries for transport.

*At the programmer prompt you can **D ^DSIO61** to be assisted in building a CONTROL entry.*

At the **FORM** you would have created an new entry in DSIO DDCCS FORM CONFIGURATION (#19641.42) which now you'll need to edit before you can run your program.

FILE STRUCTURE

FIELD NUMBER	FIELD NAME
-----------------	---------------

.01	INTERFACE (RF), [0;1]
.02	ACTION (F), [0;2]
.03	SHARED (S), [0;3]
.04	PUSH (AS A WHOLE) (P19641.401'), [0;4]
.05	FILENAME (F), [0;5]
.06	RESTORE AS (P19641.42'), [0;6]
.07	MULTI-INTERFACE (S), [0;7]
1	PAGES (Multiple-19641.421), [1;0]
.01	PAGE (MNJ2,0X), [0;1]
.02	PAGE NAME (F), [0;2]
.03	HIDE (S), [0;3]
.04	VITALS (SX), [0;4]
1	CONTROLS (Multiple-19641.4211), [1;0]
.01	CONTROL (MFX), [0;1]
.02	CLASS (P19641.425), [0;2]
.03	PUSH (P19641.401'), [0;3]
.04	IDENTIFIER (F), [0;4]
.05	OBSERVATION (P19641.122'), [0;5]
2	CONFIGURATION VALUE(S) (Multiple-19641.42112), [2;0]
.01	VALUE (Wx), [0;1]
3	DIALOGS (Multiple-19641.42113), [3;0]
.01	DIALOGS (MP19641.49'), [0;1]
4	RUN ROUTINE (K), [4;E1,245]
9	REPORT ITEM (Multiple-19641.42119), [9;0]
.01	PROPERTY (FX), [0;1]
1	VALUE (F), [1;1]
2	PROPERTIES (Multiple-19641.422), [2;0]
.01	PROPERTY (MFX), [0;1]

	1	VALUE (F), [1;1]
3.1		CONFIG ID PIECES (F), [3;1]
3.2		CONFIG DELIMITER (S), [3;2]
3.3		CONFIG ROUTINE (F), [3;3]
99.1		PACKAGE (P9.4'), [99;1]

For now, we'll just start with the basics. Upon selecting our entries we need to identify the name of the program we need to run and that's in the FILENAME field. Set this field with the name of your program including the extension.

VA FileMan 22.0

Select OPTION: 1 ENTER OR EDIT FILE ENTRIES

INPUT TO WHAT FILE: 19641.42 DSIO DDCCS FORM CONFIGURATION

EDIT WHICH FIELD: ALL// FILENAME

THEN EDIT FIELD:

Select DSIO DDCCS FORM CONFIGURATION INTERFACE: NURSE POSTPARTUM - MATERNAL
TEXT INTEGRATION UTILITIES

FILENAME: DDCCSNursePostpartumMaternity.dll

If you're setting up DDCCS for a TIU Note you can either link it to a title or to be launch as a COMObject. If the latter then you would be launching the DDCCSFormBuilder.dll COMObject and then selecting the TIU Note Title you identified in DSIO DDCCS CONTROL (#19641.4).

Before running your program you need to first assign yourself the **DSIO DDCCS CONFIG** security key so that you can run the GUI side configuration tools.

- Allocation of Security Keys
- De-allocation of Security Keys
- Enter/Edit of Security Keys
- All the Keys a User Needs
- Allocate/De-Allocate Exclusive Key(s)
- Change user's allocated keys to delegated keys
- Delegate keys
- Keys For a Given Menu Tree
- List users holding a certain key
- Remove delegated keys
- Show the keys of a particular user

Select Key Management Option: ALLOCATION of Security Keys

Allocate key: DSIO DDCCS CONFIG

Another key:

Holder of key: TF

Another holder:

You've selected the following keys:

DSIO DDCS CONFIG

You've selected the following holders:

TF

You are allocating keys. Do you wish to proceed? YES//

Now, you should be able to launch your program and when you do the first thing you need to do is navigate to the Command Menu select the "Edit Configuration" option and run "Update All". Follow the images below...

Command Menu Button Location

Command Menu Options

Configuration & Reporting

What you just did was to populate your DSIO DDCS FORM CONFIGURATION (#19641.42) entry will all the components controlled by TDDCSForm and the properties set for their associated ReportItem.

Configuration of Components

Form components can be configured in VistA to populate when the form is created. For example, a TCheckBoxList can have the DIALOGS multiple (#19641.42113) populated so that when the component is built it will include a list of all added dialog (#19641.49) display names so that when they are double clicked they will launch the dialog. Likewise, the CONFIGURATION VALUE(S) multiple (#19641.42112) can be populated to just show a list of items and if you needed to get information that wasn't static you can set the RUN ROUTINE field (#4) to run a routine to set an array to populate the component.

Configuration Order...

1. RUN ROUTINE
2. Dialogs
3. Free Text

The routine (or sub-routine) being called needs to set @RET @DDCSCT where DDCSCT is incremented. In the example below LN is the item to populate in the component.


```
S @RET@(DDCSCT)=LN,DDCSCT=DDCSCT+1
```

Saving Data

When the TDDCSForm saves data it saves it down to DSIO DDCS DATA (#19641.41) under the soft variable pointer that's built from the DSIO DDCS CONTROL (#19641.4) DESTINATION FILE field and the IEN of the entry passed in via the program. All components on your form that have a ReportItem and are not set to DoNotSave = TRUE will exist here as pointers to a record of the object in DSIO DDCS ELEMENT (#19641.45). When the note is saved again these entries are updated.

```
| +-- _Note
|   +-- _TDDCSForm
|       +-- Component - Data
|       +-- Component - Data
|       +-- _Dialog
|           +-- Component - Data
|           +-- Component - Data
```

DSIO DDCS DATA FILE (#19641.41)
FILE STRUCTURE

FIELD NUMBER	FIELD NAME
.01	RECORD (RF), [0;1]
.02	CONTROL (P19641.4'), [0;2]
.03	PUSHED (D), [0;3]
.04	PUSH START (D), [0;4]
1	INTERFACE (Multiple-19641.411), [1;0]
	.01 INTERFACE (MF), [0;1]
	.02 SHARED (S), [0;2]
	1 DATA (Multiple-19641.4111), [1;0]
	.01 DATA (MP19641.45), [0;1]

DSIO DDCS ELEMENT FILE (#19641.45)
FILE STRUCTURE

FIELD NUMBER	FIELD NAME
.01	CONTROL (RFIX), [0;1]
.02	CLASS (P19641.425), [0;2]
1	VALUE (Multiple-19641.451), [1;0]
	.01 VALUE (Wx), [0;1]

Example

```
*DSIO DDCS DATA FILE (#19641.41)*
VAH>ZW ^DSIO(19641.41,46)
^DSIO(19641.41,46,0)="8054;TIU(8925,^11"
^DSIO(19641.41,46,1,0)="^19641.411^1^1"
^DSIO(19641.41,46,1,1,0)="5;DSIO(19641.42,^1"
^DSIO(19641.41,46,1,1,1,0)="^19641.4111P^23^23"
^DSIO(19641.41,46,1,1,1,1,0)=34844
^DSIO(19641.41,46,1,1,1,2,0)=34845
^DSIO(19641.41,46,1,1,1,3,0)=34846
^DSIO(19641.41,46,1,1,1,4,0)=34847
^DSIO(19641.41,46,1,1,1,5,0)=34848
^DSIO(19641.41,46,1,1,1,6,0)=34849
^DSIO(19641.41,46,1,1,1,7,0)=34850
^DSIO(19641.41,46,1,1,1,8,0)=34851
^DSIO(19641.41,46,1,1,1,9,0)=34852
^DSIO(19641.41,46,1,1,1,10,0)=34853
^DSIO(19641.41,46,1,1,1,11,0)=34854
^DSIO(19641.41,46,1,1,1,12,0)=34855
^DSIO(19641.41,46,1,1,1,13,0)=34856
^DSIO(19641.41,46,1,1,1,14,0)=34857
^DSIO(19641.41,46,1,1,1,15,0)=34858
^DSIO(19641.41,46,1,1,1,16,0)=34859
^DSIO(19641.41,46,1,1,1,17,0)=34860
^DSIO(19641.41,46,1,1,1,18,0)=34861
^DSIO(19641.41,46,1,1,1,19,0)=34862
^DSIO(19641.41,46,1,1,1,20,0)=34863
^DSIO(19641.41,46,1,1,1,21,0)=34864
^DSIO(19641.41,46,1,1,1,22,0)=34865
^DSIO(19641.41,46,1,1,1,23,0)=34866
^DSIO(19641.41,46,1,1,1,"B",34844,1)=""
^DSIO(19641.41,46,1,1,1,"B",34845,2)=""
^DSIO(19641.41,46,1,1,1,"B",34846,3)=""
^DSIO(19641.41,46,1,1,1,"B",34847,4)=""
^DSIO(19641.41,46,1,1,1,"B",34848,5)=""
^DSIO(19641.41,46,1,1,1,"B",34849,6)=""
^DSIO(19641.41,46,1,1,1,"B",34850,7)=""
^DSIO(19641.41,46,1,1,1,"B",34851,8)=""
^DSIO(19641.41,46,1,1,1,"B",34852,9)=""
^DSIO(19641.41,46,1,1,1,"B",34853,10)=""
^DSIO(19641.41,46,1,1,1,"B",34854,11)=""
^DSIO(19641.41,46,1,1,1,"B",34855,12)=""
^DSIO(19641.41,46,1,1,1,"B",34856,13)=""
^DSIO(19641.41,46,1,1,1,"B",34857,14)=""
^DSIO(19641.41,46,1,1,1,"B",34858,15)=""
^DSIO(19641.41,46,1,1,1,"B",34859,16)=""
^DSIO(19641.41,46,1,1,1,"B",34860,17)=""
^DSIO(19641.41,46,1,1,1,"B",34861,18)=""
^DSIO(19641.41,46,1,1,1,"B",34862,19)=""
^DSIO(19641.41,46,1,1,1,"B",34863,20)=""
```

```

^DSIO(19641.41,46,1,1,1,"B",34864,21)="
^DSIO(19641.41,46,1,1,1,"B",34865,22)="
^DSIO(19641.41,46,1,1,1,"B",34866,23)="
^DSIO(19641.41,46,1,1,1,"C","CKACEPIDURAL",1)="
^DSIO(19641.41,46,1,1,1,"C","CKACGENERAL",2)="
^DSIO(19641.41,46,1,1,1,"C","CKACOTHER",3)="
^DSIO(19641.41,46,1,1,1,"C","CKACSPINAL",4)="
^DSIO(19641.41,46,1,1,1,"C","CKALLERGYLATEX",5)="
^DSIO(19641.41,46,1,1,1,"C","CKBLOODTRANSFUSION",6)="
^DSIO(19641.41,46,1,1,1,"C","CKLSTPROBLEMS",7)="
^DSIO(19641.41,46,1,1,1,"C","CKPLANNEDANESTHESIA",8)="
^DSIO(19641.41,46,1,1,1,"C","LBSUMMARY",9)="
^DSIO(19641.41,46,1,1,1,"C","LISTBOXCOMPLAINTS",10)="
^DSIO(19641.41,46,1,1,1,"C","LISTBOXFAMILYHIST",11)="
^DSIO(19641.41,46,1,1,1,"C","LISTBOXMEDICALHIST",12)="
^DSIO(19641.41,46,1,1,1,"C","LISTBOXSOCIALHIST",13)="
^DSIO(19641.41,46,1,1,1,"C","MEMCHIEF",14)="
^DSIO(19641.41,46,1,1,1,"C","MEMOCOMPLAINTS",15)="
^DSIO(19641.41,46,1,1,1,"C","MEMOHISTORY",16)="
^DSIO(19641.41,46,1,1,1,"C","MEMOMEDICATIONS",17)="
^DSIO(19641.41,46,1,1,1,"C","MEMOOBEXAM",18)="
^DSIO(19641.41,46,1,1,1,"C","MEMOPHYSICAL",19)="
^DSIO(19641.41,46,1,1,1,"C","MEMOPRENATAL",20)="
^DSIO(19641.41,46,1,1,1,"C","MEMOPROBLEMSNAR",21)="
^DSIO(19641.41,46,1,1,1,"C","MEMOROS",22)="
^DSIO(19641.41,46,1,1,1,"C","RADIOGROUP3",23)="
^DSIO(19641.41,46,1,"B",5;DSIO(19641.42,"",1)="

*DSIO DDCS ELEMENT FILE (#19641.45)*
VAH>ZW ^DSIO(19641.45,34866)
^DSIO(19641.45,34866,0)="RADIOGROUP3^3"
^DSIO(19641.45,34866,1,0)="^^1^1^3160923"
^DSIO(19641.45,34866,1,1,0)="0^Established Patient"

```

You can also view the data via a menu option...

Select OPTION NAME: DSIO DDCS MAIN

DDCS MAIN MENU

Create CONTROL objects to capture

Configure DDCS forms

****View Discreet Data****

Run DDCS Control Triggers for PUSH

Remove Push Lock

Select DDCS MAIN MENU Option: VIEW Discreet Data

Select DESTINATION FILE: ****8925**** TIU DOCUMENT

Select TIU DOCUMENT: ****`8054**** OB FOLLOWUP **NOTE** AYS,SHTS 07-18-95 TIT

=====

CONTROLLED BY: 92;TIU(8925.1,

PUSHED:

INTERFACE: 5;DSIO(19641.42,
SHARED: YES

DATA -----

-- CONTROL: CKACEPIDURAL
CLASS: TCHECKBOX

VALUE -----
^Epidural

-- CONTROL: LISTBOXCOMPLAINTS
CLASS: TCHECKLISTBOX

VALUE -----
^ABDOMINAL PAIN AND CRAMPING
^BACK PAIN
^COUGH AND CONGESTION
^HEADACHES

Enter RETURN to continue or '^' to exit:



Option: DSIO DDCS MAIN

Select OPTION NAME: DSIO DDCS MAIN

DDCS MAIN MENU

Create CONTROL objects to capture
Configure DDCS forms
View Discreet Data
Run DDCS Control Triggers for PUSH
Remove Push Lock

Select DDCS MAIN MENU Option:

1. Create CONTROL objects to capture

This option will assist in creating an entry in file 19641.4 which will allow for the capture of discreet data.

2. Configure DDCS forms

This option will assist in creating an entry that will act as the configuration for an interface that can capture discreet data.

3. View Discreet Data

This option will display the components associated with a file and record combination and their values.

4. Run DDCCS Control Triggers for PUSH

This option will run through any entries in the data collected (#19641.41) among records not already PUSHed, check their control entries (#19641.4) by running the trigger logic and if true, run that data record's configuration (#19641.42 and #19641.49) PUSH which will either be field "PUSH (AS A WHOLE)" (#.04) or each PAGE (#1) CONTROLS (#1) PUSH (#.03).

```
| +-- _DSIO DDCCS FORM CONFIGURATION (#19641.42)
|   +-- _PAGE (#1)
|       +-- _CONTROLS (#1)
|           +-- PUSH (#.03)
```

1. Remove Push Lock

If running the PUSH for discreet data causes a hard error all PUSH will be locked until resolved and this option is run. This option will then remove the lock so that PUSH can resume.

Accessing the Component Data in VistA

Each component is represented in VistA as a record in DSIO DDCCS ELEMENT (#19641.45) which holds the name of the component, its class, and its values. The values are stored in a word processing field and note that the data can either be a single line like for a TEdit or many lines like a TMemo. Information about the class can be found in the DSIO DDCCS CONTROLS (#19641.425) file. The expected format of the VALUE field is that each line will have two pieces with the first piece representing the INDEX and the second piece being the value. Not all controls will use INDEX and some like a TCheckBox will use it and also include the word TRUE along with the index to indicate that that entry was checked.

You can configure your DSIO DDCCS FORM CONFIGURATION to do something with your data once it passes the check to trigger in your DSIO DDCCS CONTROL record. This action can be configured to execute a routine or to execute M code for individual components or to generate observations.

APIs

```
$$IEN^DSIO62(NAM)
```

Input: NAM (Name of the component)

Return: IEN (IEN of the component entry in #19641.45)

```
$$GET1^DSIO62(IEN)
```

Input: IEN (IEN of the component entry in #19641.45)

Return: A single line of data including INDEX

```
$$EGET1^DSIO62(NAM)
```

Input: NAM (Name of the component)

Return: A single line of data excluding INDEX

```
$$TGET1^DSIO62(NAM)
```

Input: NAM (Name of the component)

Return: 1 (True) or 0 (False)

```
$$LS^DSIO62
```

Input:

Return:

```
GETS^DSIO62(RET, IEN)
```

Input: IEN (IEN of the component entry in #19641.45)

Return: Array of data including INDEX

```
TXT^DSIO62(RET, IEN)
```

Input: IEN (IEN of the component entry in #19641.45)

Return: Array of data without INDEX

```
WGETS^DSIO62(RET, IEN)
```

Input: IEN (IEN of the component entry in #19641.45)

Return: Array of data without INDEX

```
$$CCLASS^DSIO62(FORM, NAM)
```

Input: FORM (Variable Pointer Format of 19641.42 or 19641.49 record)

Input: NAM (Name of the component)

Return: IEN of DSIO DDCCS CONTROLS (#19641.425) or 0

```
$$LIST^DSIO62(FORM, NAM)
```

Input: FORM (Variable Pointer Format of 19641.42 or 19641.49 record)
Input: NAM (Name of the component)
Return: 1 (True) or 0 (False) to indicate if the class is marked as a list

```
$$CHECK^DSIO62(RET,IEN)
```

Input: FORM (Variable Pointer Format of 19641.42 or 19641.49 record)
Input: NAM (Name of the component)
Return: 1 (True) or 0 (False) to indicate if the class is marked as a check

Triggering to Run Routines

The configuration file points to the DSIO DDCCS REPORT ITEMS (#19641.401) file which can have an item that is used to run code when the whole interface is triggered to be pushed or when a single component is pushed. In the case of single components each component of an interface can have its own push logic or none at all.

```
| +-- _DSIO DDCCS FORM CONFIGURATION (#19641.42)  
|      +-- _PUSH (AS A WHOLE) (#.04) = P19641.401
```

```
#19641.401  
-----  
.01      NAME  
.02      INACTIVE  
1        INPUT TRANSFORM  
2        RUN ROUTINE
```

When writing a routine (or sub routine) for PUSH you can expect the following variables...

Variable	What it is
DDCSC	CONTROL
DDCSR	RECORD
SIEN	DESTINATION IEN
DDCSFLE	DESTINATION FILE
DFN	PATIENT
DDCSFRM	INTERFACE/FORM
DDCSE	ELEMENT IEN

Example PUSH (AS A WHOLE) for a TDDCSForm

```

Select DSIO DDCS REPORT ITEMS NAME: NURSE POSTPARTUM - DELIVERY
ANOTHER ONE:
STANDARD CAPTIONED OUTPUT? Yes// (Yes)
Include COMPUTED fields: (N/Y/R/B): NO// BOTH Computed Fields and Record Numb
(IEN)

NUMBER: 12 NAME: NURSE POSTPARTUM - DELIVERY
RUN ROUTINE: D DELV^DSIO01

```

```

DELV ; NURSE POSTPARTUM - DELIVERY
;
; 1) L
; 2) IEN
; 3) DATE RECORDED
; 4) EDC
; 5) DFN|PATIENT
; 6) STATUS
; 7) FOF|(IEN OR IDENTIFIER)
; 8) EDD
; 9) PREGNANCY END
; 10) OB IEN|OB
; 11) FACILITY IEN|FACILITY
; 12) UPDATED BY IEN|UPDATED BY
; 13) GESTATIONAL AGE
; 14) LENGTH OF LABOR
; 15) TYPE OF DELIVERY (C or V)
; 16) ANESTHESIA
; 17) PRETERM DELIVERY
; 18) BIRTH TYPE
; 19) IEN;NUMBER;NAME;GENDER;BIRTH WEIGHT;STILLBORN;APGAR1;APGAR2;STATUS;NICU|
; 20) OUTCOME
; 21) HIGH RISK FLAG
; 22) DAYS IN HOSPITAL
;
; C^IEN^COMMENT
; B^IEN|BABY|#^COMMENT
;
N ROWS,CT, LN,PREG,DATES,FAC,FA,STR,OBJ,I,BCT,DNOTES,OUT,IFC,IFCF
D GETS^DSIO62(.ROWS,$$IEN^DSIO62("##TCONFIGCOLLECTION##")) Q:'$D(ROWS)
; *** PREGNANCY
; Update Pregnancies and Babies
S CT=0 F S CT=$O(ROWS(CT)) Q:'CT S LN=ROWS(CT) D
. K OLD,PREG,DATES,FAC,FA,STR
. I $P(LN,U)="L" D
. . S OBJ(0)="PG."_$P(LN,U,2)

```



```

. . ; DATES = EDC^END^EDD
. . F I=4,9,8 S DATES=$S($D(DATES):DATES_U,1:"")_$$DT^DSIO2($P(LN,U,I))
. . I $P($P(LN,U,11),"|",2)'="" D K:$G(FAC)'?1"NVA."N FAC
. . . S FAC=$$UP^XLFSTR($P($P(LN,U,11),"|",2))
. . . I $D(^DSIO(19641.1,"F",FAC)) D Q
. . . . S FAC="NVA."_0(^DSIO(19641.1,"F",FAC,""))
. . . D SENT^DSIO1(.FA,,FAC,"F",,,,1) I FA S FAC="NVA."_+FA
. . ; *** OBSERVATIONS -----
. . ;     BABY LEVEL
. . ; Add/Update Babies
. . S BCT=0 F I=1:1:$L($P(LN,U,19),"|") S STR=$P($P(LN,U,19),"|",I) D
. . . D:STR'="" PR2^DSIO04($P(LN,U,2),STR) K OBJ(1)
. . ; *** OBSERVATIONS -----
. . ;     PREGNANCY LEVEL
. . I BCT D
. . . ;     OBS BABY COUNT
. . . D O^DSIO03(DFN,.OBJ,, "Pregnancy", "FetusBabyCount", "OTHER", "Fetus/Baby Count", BC
. . . ;     OBS BIRTH DATE OF FETUS
. . . D O^DSIO03(DFN,.OBJ,, "Outcome", "75092-7", "LOINC", "Birth Date of Fetus", $P(LN,U,
. . ;     OBS GESTATIONAL AGE
. . D O^DSIO03(DFN,.OBJ,, "DeliveryDetails", "412726003", "SNOMED-CT", "Gestational Age a
. . S STR=$$UP^XLFSTR($E($P(LN,U,15),1))
. . ;     OBS DELIVERY TYPE
. . D O^DSIO03(DFN,.OBJ,, "DeliveryDetails", "CesareanDelivery", "OTHER", , $S(STR="C": "Tr
. . D O^DSIO03(DFN,.OBJ,, "DeliveryDetails", "NormalSpontaneousVaginalDelivery", "OTHER"
. . D O^DSIO03(DFN,.OBJ,, "DeliveryDetails", "OtherDelivery", "OTHER", , $S(STR="" : "True",
. . ;     OBS DAYS IN HOSPITAL
. . D O^DSIO03(DFN,.OBJ,, "DeliveryDetails", "DaysInHospital", "OTHER", , $P(LN,U,22))
. . ;     OBS PREGNANCY OUTCOME
. . I $$UP^XLFSTR($P(LN,U,20))="TERMINATION" D
. . . D O^DSIO03(DFN,.OBJ,, "PregnancyTerminationOutcome", "Indication", "OTHER", , "Unkno
. . D O^DSIO03(DFN,.OBJ,, "Outcome", "OutcomeType", "OTHER", "Pregnancy Outcome", $$OT^DSI
. . ; *** PREGNANCY - Close it out
. . K PREG D PREG^DSIO15(.PREG, $P(LN,U,2), DFN, DATES, , , , $G(FAC), , U_ $P(LN,U,14)_^^_ $P
. . ; *** DELIVERY COMMENTS
. I $P(LN,U)="C" D
. . S DNOTES(+ $P(LN,U,2), + $O(DNOTES(+ $P(LN,U,2), ""), -1)+1) = $P(LN,U,3,9999)
;     *** OBS PREGNANCY NOTES
I $D(DNOTES) D
. S CT=0 F S CT=$O(DNOTES(CT)) Q: 'CT D
. . S OBJ(0) = "PG."_CT K OUT M OUT = DNOTES(CT)
. . D O^DSIO03(DFN,.OBJ,, "DeliveryDetails", "Notes", "OTHER", , , , .OUT)
; *** OBS OBSTETRIC DELIVERY METHOD
D D1($S($$TGET1^DSIO62("CKDELIVERYMETHODC"): "Cesarean Delivery", $$TGET1^DSIO62("CKDEL
;     OBS FailedForcepVacuumDelivery
F IFC="EDCPRIMARYFOR", "CBREASONSCPRIMARY", "EDREASONSCOTHPRIMARY", "CBREASONSCSECONDARY
. I $TR($$UP^XLFSTR($$EGET1^DSIO62(IFC)), " ") ["FAILEDFORCEP" S IFCF=1
I $D(IFCF), $$TGET1^DSIO62("CKVAGVACUUM") D O^DSIO03(DFN,.OBJ,, "DeliveryDetails", "Fail
; *** OBS CESAREAN INCISION
S STR=$$EGET1^DSIO62("RGINCISION") I STR'="" D
. D O^DSIO03(DFN,.OBJ,, "DeliveryDetails", "11466000", "SNOMED-CT", "Cesarean Delivery", S
; *** OBS TOTAL PREGNANCIES, STILLBIRTHS, PRETERM, LIVING, TERM BIRTHS

```

```
D CALC
Q
```

Example PUSH (AS A WHOLE) for a TDDCSDialog

```
Select DSIO DDCS REPORT ITEMS NAME: GENETIC SCREENING (D)
ANOTHER ONE:
STANDARD CAPTIONED OUTPUT? Yes// (Yes)
Include COMPUTED fields: (N/Y/R/B): NO// BOTH Computed Fields and Record Number (IEN)
```

```
NUMBER: 13 NAME: GENETIC SCREENING (D)
RUN ROUTINE: D GENETI^DSIO04
```

```
GENETI ; Genetic Screening
N I, LN, VAL, OUT, NAR
; CONTROL_BOOLEAN ; CONTROL_NARRATIVE ; CODE_SYSTEM ; CODE ; DISPLAY NAME
F I=1:1 S LN=$P($T(GENETI+I^DSIO05),";",2) Q:LN="" D
. K OUT, NAR S VAL=$$UP^XLSTR($$GET1^DSIO62($$IEN^DSIO62($P(LN,";")))
. I $P(VAL,U)="TRUE" D
. . S VAL=$$CKVAL($P(VAL,U,2)) Q:VAL="Unknown"!(VAL="")
. . S OUT=$P($$GET1^DSIO62($$IEN^DSIO62($P(LN,";"),2)),U,2) S:OUT'="" NAR(1)=OUT
. . D O^DSIO03(DFN,,"FAMMEMB",,$P(LN,";",4),$P(LN,";",3)_"^1.3.6.1.4.1.19376.1.5.3.1.
Q
```

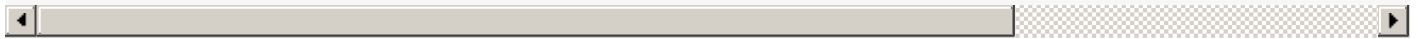
Transferring Data from one form to Restore to Another

The example use of this functionality is while using TIU Notes. You can set up the DSIO DDCS CONTROL (#19641.4) to control three different note titles that each have their own configuration (DSIO DDCS FORM CONFIGURATION #19641.42) which each call up the same FILENAME (#.05). In this case you can have all three records with the same starting values but two entries can restore (RESTORE AS #.06) from the configuration (pointer back to #19641.42) of another.

```
#Note 1 = RESTORE AS = ""
#Note 2 = RESTORE AS = #Note 1
#Note 3 = RESTORE AS = #Note 2
```

Since each note is using the same program and the same configuration this means that it can restore the data elements of its current configuration by pulling in the values saved to the same components using the configuration set as RESTORE AS.

```
#Note 1 = COMPONENT1 = "Test Text" (Saved and Completed)
#Note 2 = COMPONENT1 = "Test Text" (Start Up/Creation - Get values to components mat
#Note 3 = COMPONENT1 = "Test Text" (Start Up/Creation - Get values to components mat
```



If you change the value of COMPONENT1 in #Note 2 save and complete then that value will be restored in #Note 3.