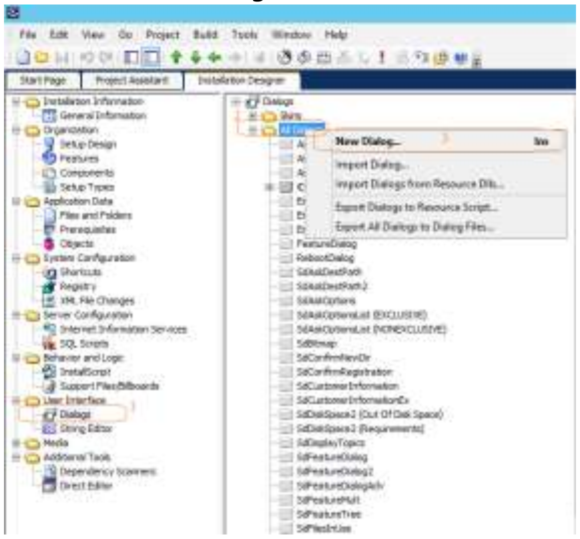
InstallShield Custom dialog

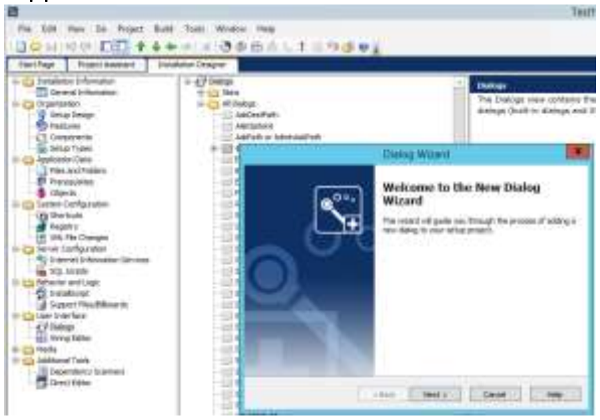
This document explains the steps, how to create/add a custom dialog to your installer.

Steps:-

1. Expand ***User Interface*** section from the left panel, select ***Dialogs***.
2. Right click on ***All Dialogs*** → select ***New Dialog…***



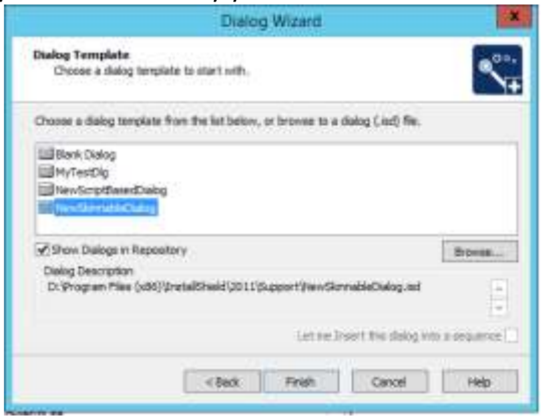
- ***New Dialog Wizard*** will appear. Click Next.



Welcome to the New Dialog Wizard

The wizard will guide you through the process of adding a new dialog to your setup project.

3. Select ***NewSkinnableDialog*** template. Blank Dialog is also fine, but selecting a dialog with skin got its advantage in case you wish to modify your UI with installer skins.



Dialog Template

Choose a dialog template to start with.

Choose a dialog template from the list below, or browse to a dialog (.isd) file.

Blank Dialog
MyTestDlg
NewScriptBasedDialog
NewSkinnableDialog

☑ Show Dialogs in Repository

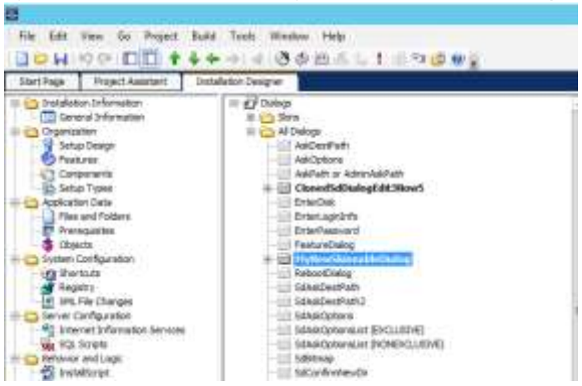
Dialog Description

D:\Program Files (x86)\InstallShield\2011\Support\NewSkinnableDialog.isd

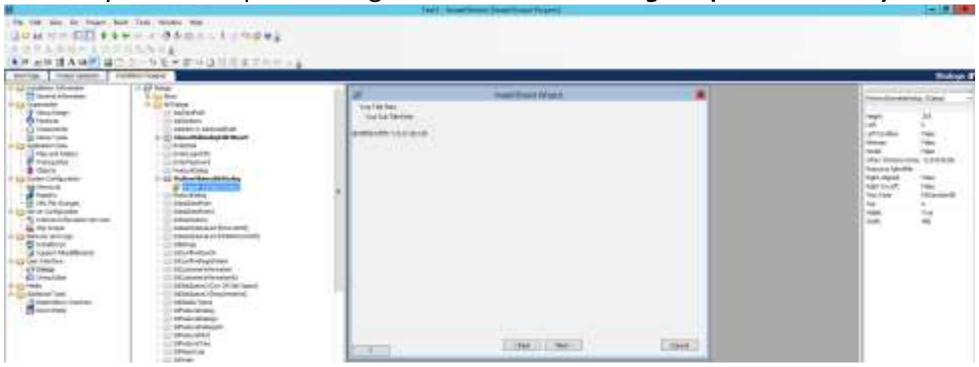
Let me Insert this dialog into a sequence ☐

InstallShield Custom dialog

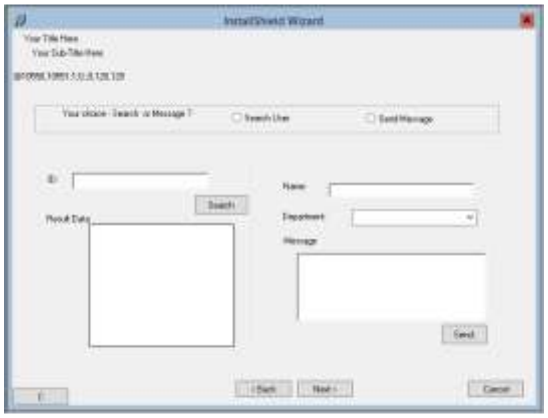
4. Name your dialog as you prefer. I named it as ***MyNewSkinnableDialog***.



- Click the **+** symbol to expand dialog content and select ***English (United States)*** as in screenshot.

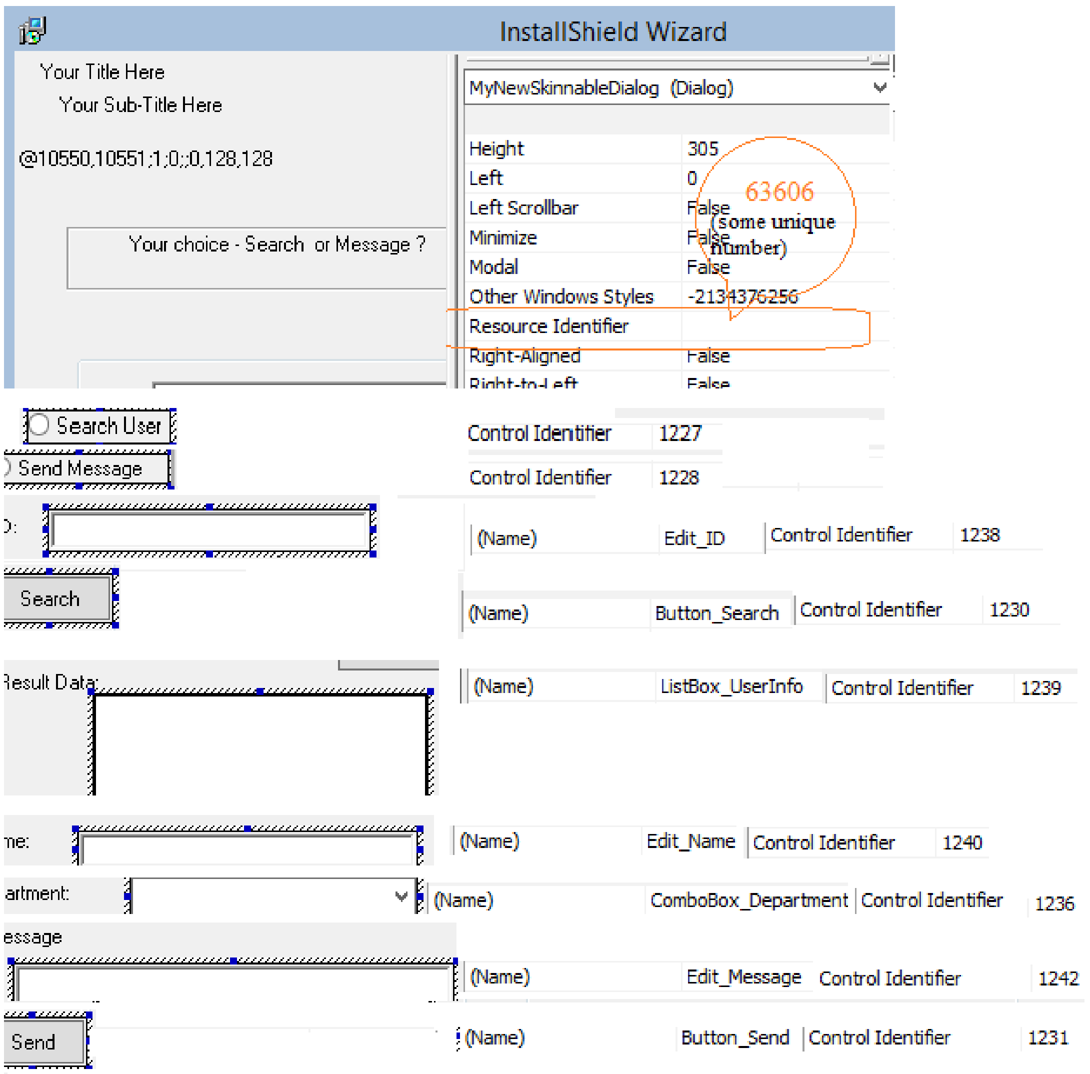


5. Add controls in your dialog as you wish.
   Two things to be noted are the *name* of your control and its *identifier*.
   My dialog looks as below.



InstallShield Custom dialog

- For the sake of clarity, displayed are my Dialog's Identifiers and name of control. Yours will be different.



InstallShield Wizard

Your Title Here

Your Sub-Title Here

@10550,10551;1;0;;0,128,128

Your choice - Search or Message ?

MyNewSkinnableDialog (Dialog)

Height	305
Left	0
Left Scrollbar	False
Minimize	False
Modal	False
Other Windows Styles	-2134376256
Resource Identifier	
Right-Aligned	False
Right-to-Left	False

63606 (some unique number)

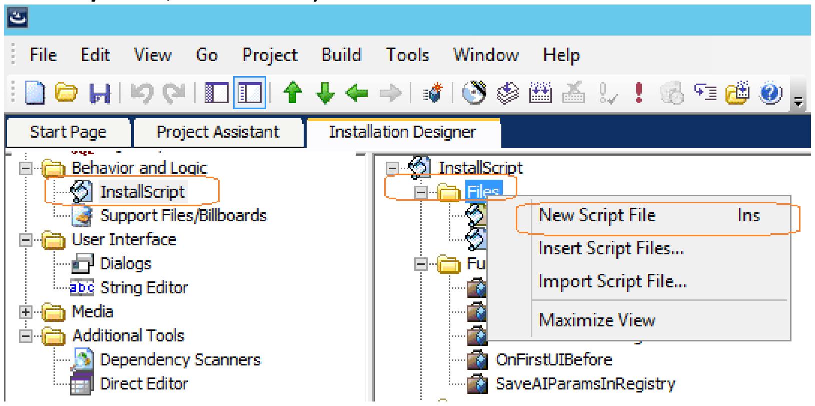
Search User — Control Identifier 1227

Send Message — Control Identifier 1228

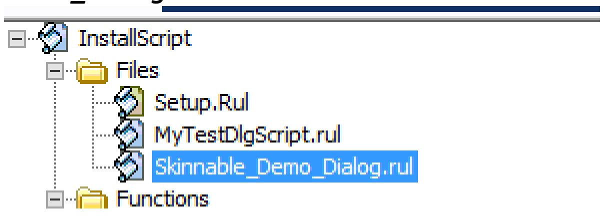
Control	(Name)		Control Identifier	
D:	(Name)	Edit_ID	Control Identifier	1238
Search	(Name)	Button_Search	Control Identifier	1230
Result Data:	(Name)	ListBox_UserInfo	Control Identifier	1239
me:	(Name)	Edit_Name	Control Identifier	1240
artment:	(Name)	ComboBox_Department	Control Identifier	1236
essage	(Name)	Edit_Message	Control Identifier	1242
Send	(Name)	Button_Send	Control Identifier	1231

InstallShield Custom dialog

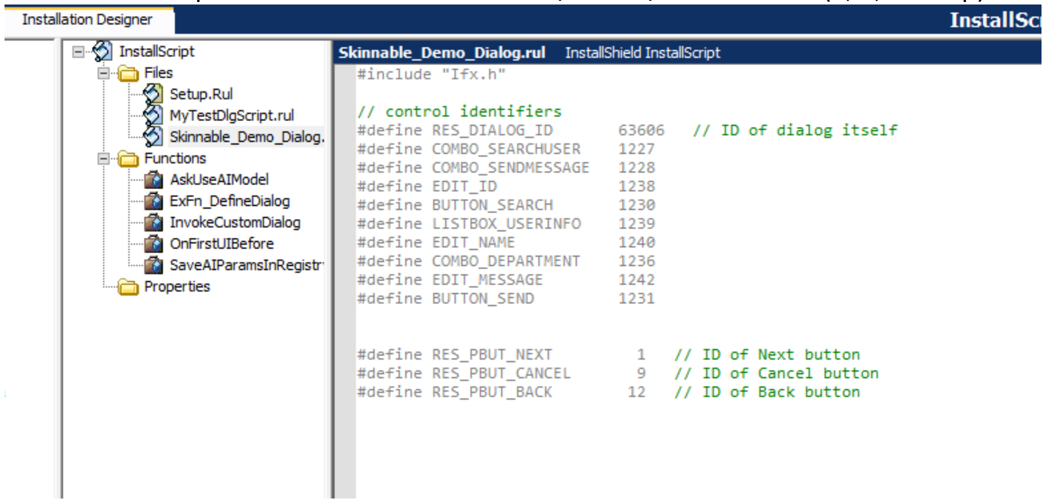
6. Attach the dialog to your installer.
   Create a ***New Script File***, name it as you like.



- Mine is ***Skinnable_Demo_Dialog***.



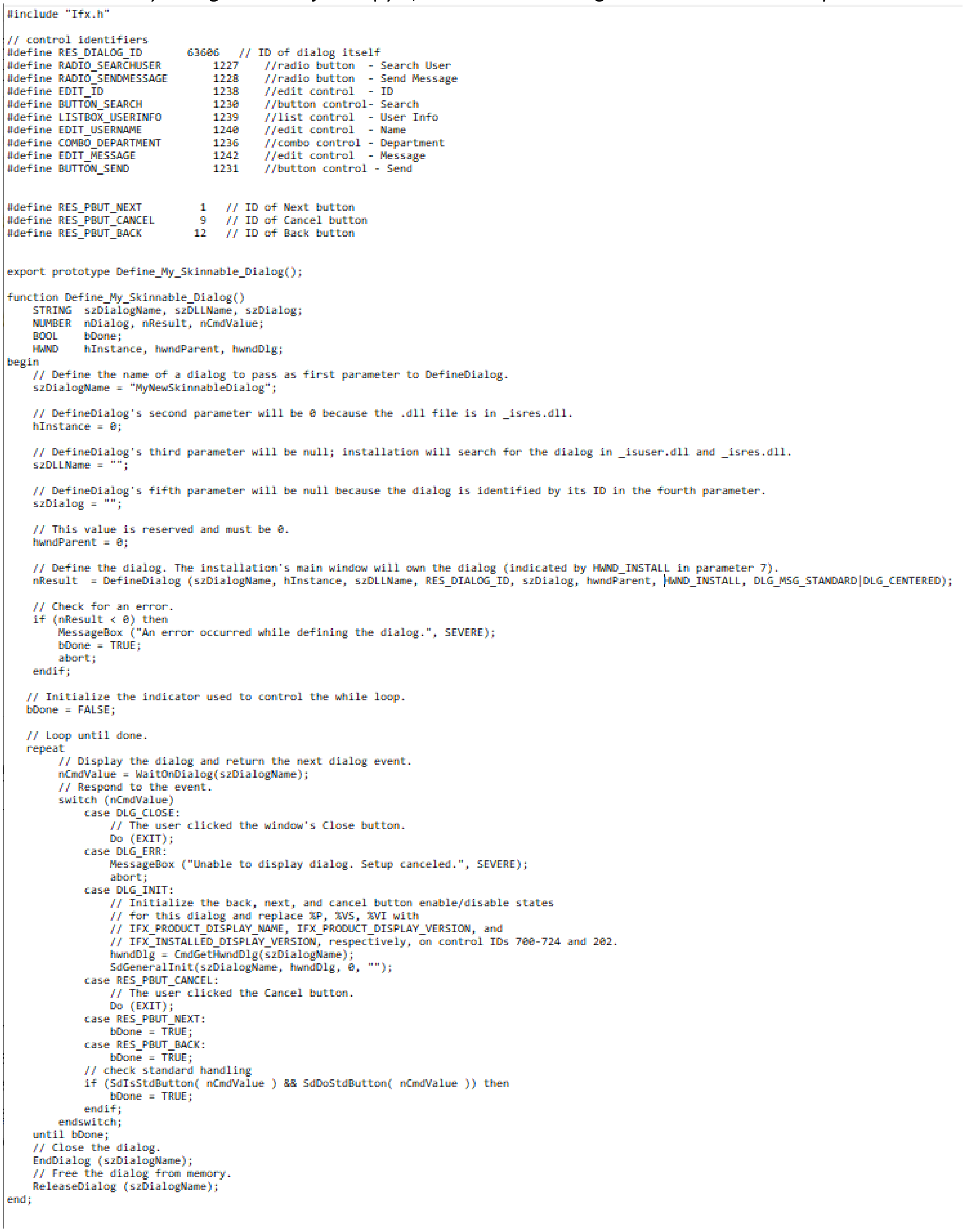
7. ***#define*** your identifiers on those you need control over
   InstallShield has predefined values for the Next/Cancel/Back buttons (1, 9, 12 resp)



```
#include "Ifx.h"

// control identifiers
#define RES_DIALOG_ID        63606   // ID of dialog itself
#define COMBO_SEARCHUSER     1227
#define COMBO_SENDMESSAGE    1228
#define EDIT_ID              1238
#define BUTTON_SEARCH        1230
#define LISTBOX_USERINFO     1239
#define EDIT_NAME            1240
#define COMBO_DEPARTMENT     1236
#define EDIT_MESSAGE         1242
#define BUTTON_SEND          1231


#define RES_PBUT_NEXT          1   // ID of Next button
#define RES_PBUT_CANCEL        9   // ID of Cancel button
#define RES_PBUT_BACK         12   // ID of Back button
```

InstallShield Custom dialog

- You need to script the basic execution code (Message Processor) in this file. This code is the same for any dialog. You can just copy it, remember to change the function name as yours.

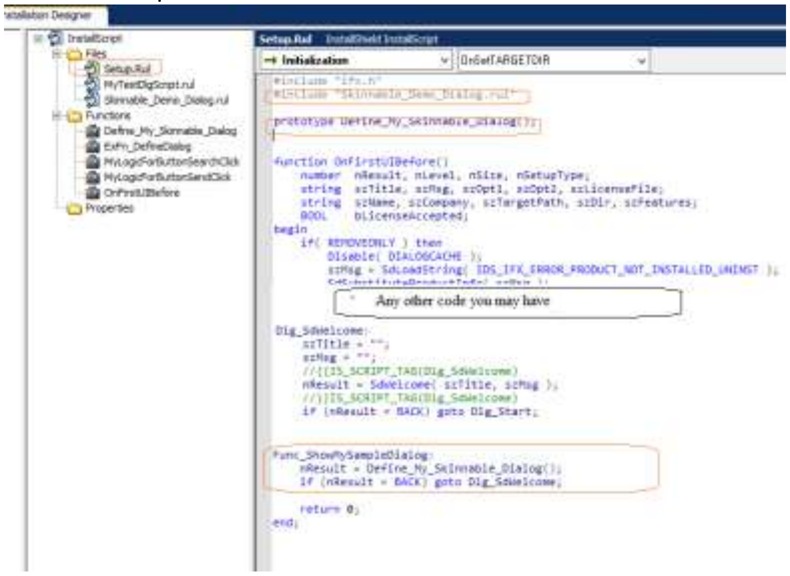


```
#include "Ifx.h"

// control identifiers
#define RES_DIALOG_ID       63606   // ID of dialog itself
#define RADIO_SEARCHUSER        1227    //radio button  - Search User
#define RADIO_SENDMESSAGE       1228    //radio button  - Send Message
#define EDIT_ID                 1238    //edit control  - ID
#define BUTTON_SEARCH           1230    //button control- Search
#define LISTBOX_USERINFO        1239    //list control  - User Info
#define EDIT_USERNAME           1240    //edit control  - Name
#define COMBO_DEPARTMENT        1236    //combo control - Department
#define EDIT_MESSAGE            1242    //edit control  - Message
#define BUTTON_SEND             1231    //button control - Send


#define RES_PBUT_NEXT         1   // ID of Next button
#define RES_PBUT_CANCEL       9   // ID of Cancel button
#define RES_PBUT_BACK        12   // ID of Back button


export prototype Define_My_Skinnable_Dialog();

function Define_My_Skinnable_Dialog()
    STRING  szDialogName, szDLLName, szDialog;
    NUMBER  nDialog, nResult, nCmdValue;
    BOOL    bDone;
    HWND    hInstance, hwndParent, hwndDlg;
begin
    // Define the name of a dialog to pass as first parameter to DefineDialog.
    szDialogName = "MyNewSkinnableDialog";

    // DefineDialog's second parameter will be 0 because the .dll file is in _isres.dll.
    hInstance = 0;

    // DefineDialog's third parameter will be null; installation will search for the dialog in _isuser.dll and _isres.dll.
    szDLLName = "";

    // DefineDialog's fifth parameter will be null because the dialog is identified by its ID in the fourth parameter.
    szDialog = "";

    // This value is reserved and must be 0.
    hwndParent = 0;

    // Define the dialog. The installation's main window will own the dialog (indicated by HWND_INSTALL in parameter 7).
    nResult  = DefineDialog (szDialogName, hInstance, szDLLName, RES_DIALOG_ID, szDialog, hwndParent, HWND_INSTALL, DLG_MSG_STANDARD|DLG_CENTERED);

    // Check for an error.
    if (nResult < 0) then
        MessageBox ("An error occurred while defining the dialog.", SEVERE);
        bDone = TRUE;
        abort;
    endif;

   // Initialize the indicator used to control the while loop.
   bDone = FALSE;

   // Loop until done.
   repeat
        // Display the dialog and return the next dialog event.
        nCmdValue = WaitOnDialog(szDialogName);
        // Respond to the event.
        switch (nCmdValue)
            case DLG_CLOSE:
                // The user clicked the window's Close button.
                Do (EXIT);
            case DLG_ERR:
                MessageBox ("Unable to display dialog. Setup canceled.", SEVERE);
                abort;
            case DLG_INIT:
                // Initialize the back, next, and cancel button enable/disable states
                // for this dialog and replace %P, %VS, %VI with
                // IFX_PRODUCT_DISPLAY_NAME, IFX_PRODUCT_DISPLAY_VERSION, and
                // IFX_INSTALLED_DISPLAY_VERSION, respectively, on control IDs 700-724 and 202.
                hwndDlg = CmdGetHwndDlg(szDialogName);
                SdGeneralInit(szDialogName, hwndDlg, 0, "");
            case RES_PBUT_CANCEL:
                // The user clicked the Cancel button.
                Do (EXIT);
            case RES_PBUT_NEXT:
                bDone = TRUE;
            case RES_PBUT_BACK:
                bDone = TRUE;
            // check standard handling
            if (SdIsStdButton( nCmdValue ) && SdDoStdButton( nCmdValue )) then
                bDone = TRUE;
            endif;
        endswitch;
    until bDone;
    // Close the dialog.
    EndDialog (szDialogName);
    // Free the dialog from memory.
    ReleaseDialog (szDialogName);
end;
```

InstallShield Custom dialog

8. To make the call to your dialog, remember three things:-
   a. Include the .rul file as header.
   b. Expose the function (in my example Define_My_Skinnable_Dialog).
   c. Call the method.

- You can see all three steps in below screenshot.

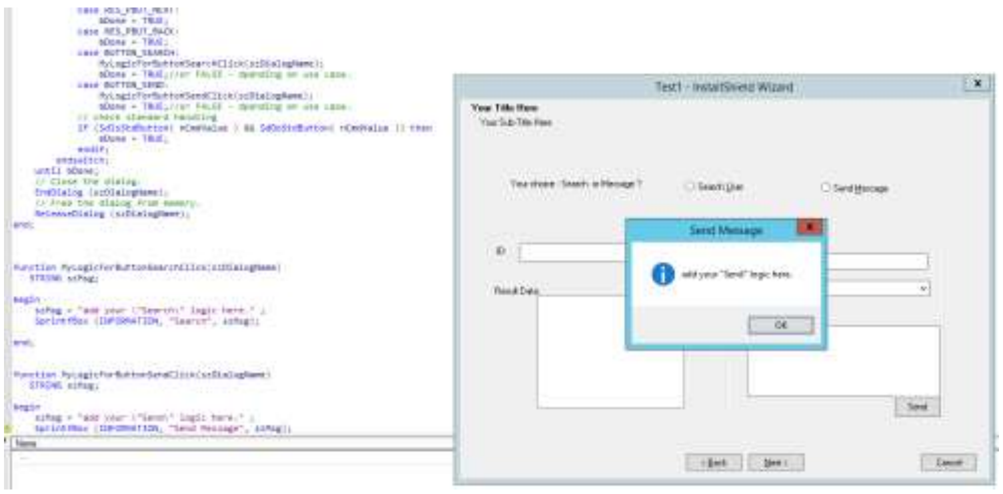


```
#include "ifx.h"
#include "Skinnable_Demo_Dialog.rul"

prototype Define_My_Skinnable_Dialog();

function OnFirstUIBefore()
    number  nResult, nLevel, nSize, nSetupType;
    string  szTitle, szMsg, szOpt1, szOpt2, szLicenseFile;
    string  szName, szCompany, szTargetPath, szDir, szFeatures;
    BOOL    bLicenseAccepted;
begin
    if( REMOVEONLY ) then
        Disable( DIALOGCACHE );
        szMsg = SdLoadString( IDS_IFX_ERROR_PRODUCT_NOT_INSTALLED_UNINST );
        SdSubstituteProductInfo( szMsg );
```

Any other code you may have

```
Dlg_SdWelcome:
    szTitle = "";
    szMsg = "";
    //{{IS_SCRIPT_TAG(Dlg_SdWelcome)
    nResult = SdWelcome( szTitle, szMsg );
    //}}IS_SCRIPT_TAG(Dlg_SdWelcome)
    if (nResult = BACK) goto Dlg_Start;


Func_ShowMySampleDialog:
    nResult = Define_My_Skinnable_Dialog();
    if (nResult = BACK) goto Dlg_SdWelcome;

    return 0;
end;
```

Code in Action!



```
            case RES_PBUT_NEXT:
                bDone = TRUE;
            case RES_PBUT_BACK:
                bDone = TRUE;
            case BUTTON_SEARCH:
                MyLogicForButtonSearchClick(szDialogName);
                bDone = TRUE;//or FALSE - depending on use case.
            case BUTTON_SEND:
                MyLogicForButtonSendClick(szDialogName);
                bDone = TRUE;//or FALSE - depending on use case.
            // check standard handling
            if (SdIsStdButton( nCmdValue ) && SdDoStdButton( nCmdValue )) then
                bDone = TRUE;
            endif;
        endswitch;
    until bDone;
    // Close the dialog.
    EndDialog (szDialogName);
    // Free the dialog from memory.
    ReleaseDialog (szDialogName);
end;


function MyLogicForButtonSearchClick(szDialogName)
   STRING szMsg;

begin
    szMsg = "add your \"Search\" logic here." ;
    SprintfBox (INFORMATION, "Search", szMsg);

end;


function MyLogicForButtonSendClick(szDialogName)
   STRING szMsg;

begin
    szMsg = "add your \"Send\" logic here." ;
    SprintfBox (INFORMATION, "Send Message", szMsg);
```

Script file of this dialog is attached. Code just displays a dialog when one of the button is pressed.

Skinnable_Demo_Dialog.rul_final

Fun with Scripting
Renjith