Adding Items to the Print Dialogs

The Print Manager dialog was originally designed to be as generic as possible so that applications could print without being device-specific. There are times, however, when an application may need additional information that is not part of the Print Manager interface before it can print. The following example describes how to add items to the existing style and job dialogs.

Apple has guidelines to keep in mind before attempting to modify the print dialogs.

- Do not change the position of items in the current dialogs. This means don't delete items from the existing item list or add items in the middle. Only add items to the end of the list.
- Don't depend on items retaining their positions in the list.
- Don't use more than half the screen height for your items. Apple reserves the right to expand the items in the standard print dialogs to fill the top half of the screen.
- If you are adding a lot of items to the dialogs, you should consider having a separate dialog in addition to the standard print dialogs.

When your application calls **PrStlDialog** and **PrJobDialog**, the printer driver calls **PrDlgMain**. **PrDlgMain** calls the pDlgInit routine (its second parameter) to set up the dialog, dialog hook and dialog event filter. After the TPrDlg record has been setup, **PrDlgMain** calls **ShowWindow** (since the window is initially invisible), then it calls **ModalDialog**, using the dialog event filter pointed to by the pFltrProc field. When an item is hit, the pFltrProc's procedure is called and the items are handled appropriately. When the OK button is hit, the print record is validated. The print record is not validated if the Cancel button is hit.

To modify the print dialogs, you need to change the PTrDlg record before the dialog is drawn. You can add items to the list and/or replace the standard dialog hook and event filters.

The following example is long, yet complete. The Rez source is included, but if you only have ResEdit, note that the ".rsrc" file consists of a single resource of type 'DITL' with an ID of 256. The size of the two check boxes can be found in the Rez source code.

Example

```
/* Example of how to add items to a Print Dialog Box */
/* Rez source for required resource is included */
/* MacHeaders is used for this example */

/* Rez source */
#if 0
/* use preprocessor to avoid nested comments - not allowed by C */
    {   /* array DITLarray: 2 elements */
        /* [1] */
```

```
        {8, 0, 24, 112},
        CheckBox {
            enabled,
            "First Box"
        };
        /* [2] */
        {8,175, 24, 287},
        CheckBox {
            enabled,
            "Second Box"
        }
    }
};
#endif

#include <PrintTraps.h>
#define nil 0L
#define MyDITL 256  /* resource ID of our DITL to be spliced on to job dialog */

TPPrDlg PrtJobDialog;
pascal void MyJobItems();                   /* our modal item handler */
THPrint hPrintRec;                          /* handle to print record */
short FirstBoxValue = 0;                    /* value of first additional box */
short SecondBoxValue = 0;                   /* value of our second box */
long prFirstItem;       /* save first item here */
long prPItemProc;                           /* store the old itemProc here */
WindowPtr MyWindow;
OSErr err;
Str255 MyStr;

/* prototypes */
pascal TPPrDlg MyJobDlgInit(THPrint);
pascal Boolean PrDlgMain();                 /* print managers dialog handler */
pascal TPPrDlg PrJobInit();                 /* get standard print job dialog */
pascal TPPrDlg MyJobDlgInit();              /* our extension to PrJobInit */
OSErr Print(void);
short AppendDITL(DialogPtr, short);
pascal void MyJobItems(TPPrDlg, short);

main()
{
    Rect myWRect;

    InitGraf(&thePort);
    InitFonts();
    InitWindows();
    InitMenus();
    InitDialogs(nil);
    InitCursor();
    SetRect(&myWRect, 50, 260, 350, 340);

    /* call the routine that does printing */
    PrOpen();
    err = Print();

    PrClose();
```

```
}

OSErr Print()
{
    /* call PrJobInit to get pointer to the invisible job dialog */
    hPrintRec = (THPrint)(NewHandle(sizeof(TPrint)));
    PrintDefault(hPrintRec);
    PrValidate(hPrintRec);
    if(PrError() != noErr)
        return PrError();

    PrtJobDialog = PrJobInit(hPrintRec);
    if(PrError() != noErr)
        return PrError();

    if(!PrDlgMain(hPrintRec, &MyJobDlgInit)) /* this line does all */
        return Cancel;

    if(PrError() != noErr)
        return PrError();
}

pascal TPPrDlg MyJobDlgInit(THPrint hPrint)
{
    /*  this routine appends items to the standard job dialog and
        sets up user fields of the printing dialog record TPRDlg
    */

    short firstItem;
    short itemType;
    Handle itemH;
    Rect itemBox;

    firstItem = AppendDITL((DialogPtr)PrtJobDialog, MyDITL);

    prFirstItem = firstItem;                    /* save so MyJobItems can find it */
    /* set up DITL items - the "First Box" */
    GetDItem(PrtJobDialog, firstItem, &itemType, &itemH, &itemBox);
    SetCtlValue(itemH, FirstBoxValue);

    /* set up second of the DITL items - the "Second Box" */
    GetDItem(PrtJobDialog, firstItem, &itemType, &itemH, &itemBox);
    SetCtlValue(itemH, FirstBoxValue);

    /* now patch in the item handler. Save the old item handler
        address, so it can be called if one of the standard items
        is hit, and put our item handler's address in pItemProc field
        of the TPrDlg struct
    */
    prPItemProc = (long)PrtJobDialog->pItemProc;

    /* now tell the modal dialog handler where our routine is */
    PrtJobDialog->pItemProc = (ProcPtr)MyJobItems;

    /* PrDlgMain expects a pointer to the modifier dialog to be returned */
    return PrtJobDialog;
```

```
}

pascal void MyJobItems(TPPrDlg theDialog, short itemNo)
{
    short myItem;
    short firstItem;
    short itemType;                         /* for GetDItem/SetDItem calls */
    Handle itemH;
    Rect itemBox;

    firstItem = prFirstItem;                /* saved in myJobDlgInit */
    myItem = itemNo - firstItem + 1;        /* "localize" item number */
    if(myItem > 0)    /* if localized item > 0, it is our item */
        {
        GetDItem(theDialog, itemNo, &itemType, &itemH, &itemBox);
        switch(myItem)
            {
            case 1:
                /* invert value of FirstBoxValue and redraw it */
                FirstBoxValue ^= 1;
                SetCtlValue(itemH, FirstBoxValue);
                break;
            case 2:
                /* invert value of SecondBoxValue and redraw it */
                SecondBoxValue ^= 1;
                SetCtlValue(itemH, SecondBoxValue);
                break;
            default:
                Debugger();                 /* error state */
            }
        }

        else
            /* chain to standard item handler, whose addr is in prPItemProc */
            CallPascal(theDialog, itemNo, prPItemProc);
}

short AppendDITL(DialogPtr theDialog, short theDITLID)
{
    typedef struct {
        Handle itmHndl;
        Rect itmRect;
        char itmType;
        char itmData[];
    } DITLItem, *pDITLItem, **hDITLItem;

    typedef struct {
        short dlgMaxIndex;
        DITLItem DITLItems[];
    } ItemList, *pItemList, **hItemList;

    short *IntPtr;
    Point offset;
    Rect maxRect;
    hItemList hDITL;
    pDITLItem pItem;
```

```
hItemList hItems;
short firstItem;
short newItems, dataSize, i;
long sizeHandle;

maxRect = ((DialogPeek)theDialog)->window.port.portRect;
offset.v = maxRect.bottom;
offset.h = 0;
maxRect.bottom -= 5;
maxRect.right -= 5;
hItems = (hItemList)(((DialogPeek)theDialog)->items);
firstItem = (**hItems).dlgMaxIndex + 2;
hDITL = (hItemList)(GetResource('DITL', theDITLID));
HLock(hDITL);
newItems = (**hDITL).dlgMaxIndex + 1;

pItem = (DITLItem *)&((**hDITL).DITLItems);

for(i = 1; i <= newItems; i++)
    {
    OffsetRect(&(pItem->itmRect), offset.h, offset.v);
    UnionRect(&(pItem->itmRect), &maxRect, &maxRect);

    switch((pItem->itmType) & 0x7F){
        case userItem:
            pItem->itmHndl = nil;
            break;

        case ctrlItem + btnCtrl:
        case ctrlItem + chkCtrl:
        case ctrlItem + radCtrl:
            pItem->itmHndl =
                (Handle)NewControl(theDialog, &pItem->itmRect,
                 pItem->itmData, true, 0, 0, 1, pItem->itmType & 0x3, 0);
            break;
        case ctrlItem + resCtrl:
            pItem->itmHndl =
                (Handle)GetNewControl(pItem->itmData[1], theDialog);

            (*(ControlHandle)(pItem->itmHndl))->contrlRect =
                pItem->itmRect;
            break;
        case statText:
        case editText:
            err = PtrToHand(pItem->itmData,
                    pItem->itmHndl, pItem->itmData[0]);
            break;
        case iconItem:
            pItem->itmHndl = GetIcon(*pItem->itmData);
            break;
        case picItem:
            pItem->itmHndl = (Handle)GetPicture(pItem->itmData[1]);
            break;
        default:
            pItem->itmHndl = nil;
        }
```

```
        dataSize = (pItem->itmData[0] + 1) & 0xFFFE;
        pItem  = (DITLItem *)(((char *)pItem) +  dataSize +
sizeof(DITLItem));
        }

    err = PtrAndHand((**hDITL).DITLItems, hItems,
                        GetHandleSize(hDITL));
    (**hItems).dlgMaxIndex = (**hItems).dlgMaxIndex + newItems;
    HUnlock(hDITL);
    ReleaseResource(hDITL);
    maxRect.bottom += 5;
    maxRect.right += 5;
    SizeWindow(theDialog, maxRect.right, maxRect.bottom, true);

    return firstItem;
}
```