

```

//*****
//      MACRO: PROMPTS.WCM
//      PURPOSE:   Creates and edits the prompts inside of an automated template.
//                Use when editing or creating an automated template.
//*****
Application (A1; "WordPerfect"; Default; "EN")

UserLink := DLLLoad ("User32")
AllDone := False
CancelMacro := False

//*****
//      Program Begin
//*****
SmartQuotesWereActivated = False

If( ?SingleSmartQuote = 1 ) //Single Smart Quotes are ON
    // Ensure that Single Smart Quotes are off for apostrophes from AdressBook
    SingleSmartQuote (Off!)
    SmartQuotesWereActivated = True
Endif

vQuote = ?SingleSmartQuote

If( vQuote = 0)
    SingleSmartQuote (State: Off!)
EndIf

If (?BlockActive)
    SelectMode (Off!)           // make sure nothing is selected
EndIf
Call (IsDefault@)             // Warns user and QUIT if in default template.
Call (InitVariables@)
Call (FindObject@)           // Locate _autotmp.wpt.

// define the dialogs
DialogDefine (DlgMain; 50; 50; 182; 146; 8 + 16; "Prompt Builder")
DialogDefine (DlgEdit; 50; 50; 174; 89; 16; "")
DialogDefine (DlgPers; 50; 50; 174; 103; 8 + 16; "Personal Fields")
WaitMsgInit ()

WaitMsgDisplay ("Preparing template...")
Call (DefineDialogs@)
Call (ParseAbbr@)
WaitMsgHide ()

```

Display (On!)  
InhibitInput (Off!)  
DialogShow (DlgMain; "PromptList"; CBMain@)

ButtonPressed := ""

Repeat

    Switch (ButtonPressed)

        Caseof AddMBtn:

            DialogDismiss (DlgMain; AddMBtn)

            Call (Add@)

            DialogShow (DlgMain; AddMBtn; CBMain@)

            ButtonPressed := ""

        Caseof EditMBtn:

            DialogDismiss (DlgMain; EditMBtn)

            Call (Edit@)

            DialogShow (DlgMain; EditMBtn; CBMain@)

            ButtonPressed := ""

        Caseof DeleteMBtn:

            Call (Delete@)

            ButtonPressed := ""

        Caseof PersonalMBtn:

            DialogDismiss (DlgMain; PersonalMBtn)

            Call (Personal@)

            DialogShow (DlgMain; PersonalMBtn; CBMain@)

            ButtonPressed := ""

        Default:

    EndSwitch

Until (AllDone)

DialogDismiss (DlgMain; OKMBtn)

InhibitInput (On!)

Display (Off!)

If (Not CancelMacro)

    QuickMarkExists := CheckQuickMark ()

    WaitMsgDisplay ("Preparing template...")

    Call (MakePromptAbbr@)

    Call (BuildBookmarks@)

    Call (MakeBookmarkAbbr@)

    Assoc := TemplateGetAssociation (PostNew!; Macro!)

    If (Assoc = "")

        TemplateCopyObject (ObjectFile; Macro!; ObjectName)

        TemplateSetAssociation (PostNew!; Macro!; ObjectName)

    EndIf

    If (Not QuickMarkExists)

```

        PosDocVeryTop ()
        QuickMarkSet ()
    EndIf
EndIf

// exit cleanup
Label (Stop@)
DllFree (UserLink)
WaitMsgHide ()
WaitMsgDestroy ()
Call (DestroyDialogs@)
Go (End@)
//*****

//      Main Program End
//*****

//*****
//      ROUTINE NAME: IsDefault
//      INPUT VARIABLES: None
//      OUTPUT VARIABLES: None
//      USES:
//      DESCRIPTION: Warns user if playing macro in default template.
//*****
Label (IsDefault@)
If (ToUpper (?TemplateFile) = ToUpper (?CurrentTemplate))
    MsgBox (; "Default Template"; "You cannot add prompts to your default template.
    To add prompts to a different template: edit the template then choose Build Prompts
    again."; IconStop!)
    Go(End@)
EndIf
Return // IsDefault
//*****

//*****
//      ROUTINE NAME: FindObject
//      INPUT VARIABLES: ObjectFilename
//      OUTPUT VARIABLES: ObjectFile
//      USES: FindFile
//      DESCRIPTION: Locates _autotmp.wpt.
//*****
Label (FindObject@)
ObjectFile := FindFile (ObjectFilename)
If (ObjectFile = "")
    MsgBox (; ObjectFilename + " Not Found"; "The template named ^0 was not found
    in either of your template directories. Make sure that ^0 is in your template or
    supplemental template directory, then choose Build Prompts again."; IconStop! |

```

```

        HasParameters!; ObjectFilename)
    Go (End@)
EndIf
Return // FindObject
//*****

//*****
//    ROUTINE NAME: ParseAbbr
//    INPUT VARIABLES: PromptAbbr
//    OUTPUT VARIABLES: None
//    USES: Nothing
//    DESCRIPTION: Extracts prompts from abbreviation and stores in listbox.
//*****

Label (ParseAbbr@)
NumOfPrompts := 0
GetData (NumOfAbbrs; Abbreviation!; Count!; 0)
If (NumOfAbbrs = 0)
    Return
EndIf
// For the number of abbreviations,
For (CurrAbbrNum; 1; CurrAbbrNum <= NumOfAbbrs; CurrAbbrNum + 1)
    GetData (CurrAbbrName; Abbreviation!; Name!; 0; CurrAbbrNum)
    If (CurrAbbrName = PromptAbbr)
        // Get contents of Abbreviation
        GetData (PullString; Abbreviation!; Data!; 0; CurrAbbrNum)
        If (PullString = "")
            Go (StopParse@)
        Else
            Go (ParseLoop@)
        EndIf
    EndIf
EndFor
Go (StopParse@)
Label (ParseLoop@)
DelPos := StrPos (PullString; "|")
// Add the field to the listbox
NumOfPrompts := NumOfPrompts + 1
NewField := SubStr (PullString; 1; DelPos-1)
LBInsertWPString (UserLink; hwndList; NumOfPrompts-1; NewField)
// Cut assigned field from main string
Len := StrLen (PullString)
PullString := SubStr (PullString; DelPos + 1; Len-DelPos)
// Assign the link number to the listbox
DelPos := StrPos (PullString; "|")
If (DelPos = 0) // If last field to parse,
    LBSetItemData (UserLink; hwndList; NumOfPrompts-1; LookupLink (PullString))

```

```

        // highlight first prompt
        Go (StopParse@)
Else
    LBSetItemData (UserLink; hwndList; NumOfPrompts-1; LookupLink (SubStr
        (PullString; 1; DelPos-1)))
    Len := StrLen (PullString)
    PullString := SubStr (PullString; DelPos + 1; Len - DelPos)
    Go (ParseLoop@)
EndIf
Label (StopParse@)
Discard (NumOfAbbrs; CurrAbbrNum; CurrAbbrName; PullString; DelPos; Len)
Return
//*****

//*****
//    ROUTINE NAME: DefineDialogs@
//    INPUT VARIABLES: NumOfPrompts; NewPrompt; MaxChars; NewLink; ArrayCount;
PersFld
//    OUTPUT VARIABLES: hwndList
//    USES: Nothing
//    DESCRIPTION: Defines all macro dialogs.
//*****

Label (DefineDialogs@)
// Main Dialog
DialogAddPushButton (DlgMain; OKMBtn; 124; 8; 50; 14; 0; "OK")
DialogAddPushButton (DlgMain; CancelMBtn; 124; 26; 50; 14; CancelBbtn!; "Cancel")
DialogAddPushButton (DlgMain; AddMBtn; 124; 48; 50; 14; 1; "&Add...")
DialogAddPushButton (DlgMain; PasteMBtn; 124; 66; 50; 14; 0; "&Paste")
DialogAddPushButton (DlgMain; EditMBtn; 124; 84; 50; 14; 0; "&Edit...")
DialogAddPushButton (DlgMain; DeleteMBtn; 124; 102; 50; 14; 0; "&Delete")
DialogAddPushButton (DlgMain; PersonalMBtn; 124; 124; 50; 14; 0; "Pe&rsonal...")
DialogAddText (DlgMain; "T1"; 8; 8; 100; 10; 1; "&Template prompts: ")
DialogAddListBox (DlgMain; PromptlistMListBx; 8; 19; 108; 54; 32; CurrPrompt)
DialogAddPushButton (DlgMain; MoveUpMBtn; 8; 81; 50; 14; 0; "Move &up")
DialogAddPushButton (DlgMain; MoveDnMBtn; 66; 81; 50; 14; 0; "Move do&wn")
DialogAddText (DlgMain; HelpMBtn; 8; 103; 108; 35; 1; "Add the prompts your template will
display. Paste the prompts where you want the responses to appear.")
DialogLoad (DlgMain)
hwndList := DialogHandle (DlgMain; PromptlistMListBx)
hwndEdit := DialogHandle (DlgMain; EditMBtn)
hwndPaste := DialogHandle (DlgMain; PasteMBtn)
hwndDelete := DialogHandle (DlgMain; DeleteMBtn)
hwndMoveUp := DialogHandle (DlgMain; MoveUpMBtn)
hwndMoveDn := DialogHandle (DlgMain; MoveDnMBtn)
hwndDlgMain := DialogHandle (DlgMain)

```

### // Edit Dialog

```
DialogAddText (DlgEdit; "T1"; 8; 8; 100; 10; 1; "&Prompt:")
DialogAddEditBox (DlgEdit; PromptTxtMEditBx; 8; 19; 100; 13; 32; NewPrompt; MaxChars)
DialogAddText (DlgEdit; "T2"; 8; 38; 100; 10; 1; "&Link to Address Book field:")
DialogAddComboBox (DlgEdit; ABkLinkMCombo; 8; 49; 100; 100; DropDown! | Sorted!;
NewLink; 255)
For (Count; 1; Count <= ArrayCount; Count + 1)
    DialogAddListItem (DlgEdit; ABkLinkMCombo; Array (Count; 1))
EndFor
DialogAddCheckBox (DlgEdit; ShowAllCheckBx; 8; 70; 100; 10; "&Show all available fields";
ShowAllFields)
DialogAddPushButton (DlgEdit; OKMBtn; 116; 8; 50; 14; DefaultBtn!; "OK")
DialogAddPushButton (DlgEdit; CancelMBtn; 116; 26; 50; 14; CancelBtn!; "Cancel")
DialogLoad (DlgEdit)
hwndDlgEdit := DialogHandle (DlgEdit)
```

### // Personal Dialog

```
DialogAddListBox (DlgPers; PersDataListBx; 8; 8; 100; 72; Sorted!; PersFld)
For (Count; 2; Count <= ArrayCount; Count + 1) // skip the "< none>" entry
    DialogAddListItem (DlgPers; PersDataListBx; Array (Count; 1))
EndFor
DialogAddCheckBox (DlgPers; ShowAllCheckBx; 8; 84; 100; 10; "&Show all available fields";
ShowAllFields)
DialogAddPushButton (DlgPers; PasteMBtn; 116; 8; 50; 14; DefaultBtn!; "&Paste")
DialogAddPushButton (DlgPers; CloseMBtn; 116; 24; 50; 14; CancelBtn!; "&Close")
DialogLoad (DlgPers)
hwndPersList := DialogHandle (DlgPers; PersDataListBx)
hwndPersPaste := DialogHandle (DlgPers; PasteMBtn)
Return
```

```
//*****
```

```
//*****
```

```
// ROUTINE NAME: CBMain@
// INPUT VARIABLES: None
// OUTPUT VARIABLES: AllDone; CancelMacro
// USES: LB... Functions
// DESCRIPTION: Callback routine for main dialog.
```

```
//*****
```

```
Label (CBMain@)
```

```
Switch (CBMain[3])
```

```
CaseOf "":
```

```
Switch (CBMain[5])
```

```
CaseOf 6: // WM_ACTIVATE message
```

```
CurSel := LBGetCurSel (UserLink; hwndList)
```

```
NumItems := LBGetCount (UserLink; hwndList)
```

```
If ((CurSel = -1) And (NumItems>0))
```

```

        LBSetCurSel (UserLink; hwndList; 0)
    EndIf
    EnableWindow (UserLink; hwndEdit; NumItems>0)
    EnableWindow (UserLink; hwndDelete; NumItems>0)
    EnableWindow (UserLink; hwndPaste; NumItems>0)
    EnableWindow (UserLink; hwndMoveUp; NumItems>1)
    EnableWindow (UserLink; hwndMoveDn; NumItems>1)
    CaseOf 274: // WM_SYSCOMMAND
        AllDone := True
        CancelMacro := True
    EndSwitch

    CaseOf AddMBtn:
        ButtonPressed := AddMBtn

    CaseOf EditMBtn:
        ButtonPressed := EditMBtn

    CaseOf DeleteMBtn:
        Call (Delete@)

    CaseOf PasteMBtn:
        Call (Paste@)

    CaseOf PersonalMBtn:
        ButtonPressed := PersonalMBtn

    CaseOf OKMBtn:
        AllDone := True

    CaseOf CancelMBtn:
        AllDone := True
        CancelMacro := True

    CaseOf MoveDnMBtn:
        MoveItemDown (UserLink; hwndList)

    CaseOf MoveUpMBtn:
        MoveItemUp (UserLink; hwndList)

    Default: Return
EndSwitch
Return
//*****
//*****

```

```

//      ROUTINE NAME: CBPers@
//      INPUT VARIABLES: None
//      OUTPUT VARIABLES: DonePasting
//      USES: LBGetCurSel; LBGetText
//      DESCRIPTION: Callback routine for personal dialog.
//*****
Label (CBPers@)
Switch (CBPers[3])
  CaseOf CloseMBtn:
    DonePasting := True
  CaseOf PasteMBtn:
    FieldName := LBGetText (UserLink; hwndPersList; LBGetCurSel (UserLink;
    hwndPersList))
    LookupLink (FieldName)
    Type ("<" + FieldName + ">")
  CaseOf ShowAllCheckBox:
    RegionEnableWindow (DlgPers; Disable!)
    If (ShowAllFields = 0)
      AddAllFields (DlgPers + "." + PersDataListBx)
    Else
      RemoveExtraFields (DlgPers + "." + PersDataListBx)
    EndIf
    RegionEnableWindow (DlgPers; Enable!)
    RegionSetFocus (DlgPers)
EndSwitch
Return
//*****

```

```

//*****
//      ROUTINE NAME: CBEdit@
//      INPUT VARIABLES: None
//      OUTPUT VARIABLES: None
//      USES: Nothing
//      DESCRIPTION: Callback routine for Add/Edit dialog.
//*****
Label (CBEdit@)
Switch (CBEdit[3])
  CaseOf CancelMBtn:
    DoneEditing := True
    DialogDismiss (DlgEdit; CancelMBtn)
  CaseOf OKMBtn:
    DoneEditing := True
    DialogDismiss (DlgEdit; OKMBtn)
  CaseOf ShowAllCheckBox:
    RegionEnableWindow (DlgEdit; Disable!)
    If (ShowAllFields = 0)

```



```

                AddAllFields (DlgEdit + "." + ABkLinkMCombo)
            Else
                RemoveExtraFields (DlgEdit + "." + ABkLinkMCombo)
            EndIf
            RegionEnableWindow (DlgEdit; Enable!)
            RegionSetFocus (DlgEdit)
        EndSwitch
    Return
    /*******
    /*******
    //    ROUTINE NAME: Add@
    //    INPUT VARIABLES: MaxPrompts
    //    OUTPUT VARIABLES: NewPrompt
    //    USES: ErrorChecks@; LBGetCount
    //    DESCRIPTION: Adds new prompt to prompt list.
    /*******
    Label (Add@)
    If (LBGetCount (UserLink; hwndList)>= MaxPrompts)
        MessageBox (; "Too Many Prompts"; "You cannot add more than ^0 prompts.";
        IconExclamation! | HasParameters!; MaxPrompts)
        Return
    EndIf
    If (?BlockActive)
        NewPrompt := ?SelectedText
    Else
        NewPrompt := ""
    EndIf
    NewLink := Array (1; 1)    // Set initial link to "none"
    SetWindowText (UserLink; hwndDlgEdit; "Add Template Prompt")

    Label (DispAdd@)
    DoneEditing := False
    DialogShow (DlgEdit; PromptTxtMEditBx; CBEedit@)
    Repeat
    Until (DoneEditing)
    If (MacroDialogResult = OKMBtn)
        Call (ErrorChecks@)
        If (ErrorExists)
            Go (DispAdd@)
        EndIf
        If (?BlockActive)
            Type ("[" + NewPrompt + "]")
        EndIf
        Index := LBGetCount (UserLink; hwndList)
        LBInsertWPString (UserLink; hwndList; Index; NewPrompt)

```

```

    LinkIndex := LookupLink (NewLink)
    LBSetItemData (UserLink; hwndList; Index; LinkIndex)
    ArraySet (LinkIndex; NewLink; False)
    LBSetCurSel (UserLink; hwndList; Index)
EndIf
Return
//*****

//*****
//    ROUTINE NAME: Paste@
//    INPUT VARIABLES:
//    OUTPUT VARIABLES: None
//    USES: CheckSelection; LBGetCurSel
//    DESCRIPTION: Pastes highlighted prompt in document.
//*****
Label (Paste@)
If (Not CheckSelection (UserLink; hwndList))
    Return
EndIf
Index := LBGetCurSel (UserLink; hwndList)
Type ("[" + LBGetWPTText (UserLink; hwndList; Index) + "]")
Return
//*****

//*****
//    ROUTINE NAME: Edit@
//    INPUT VARIABLES:
//    OUTPUT VARIABLES:
//    USES: CheckSelection; LBGetCurSel; ErrorChecks@; UpdateDoc; LBDeleteString;
LBInsertWPString
//    DESCRIPTION: Edits the highlighted prompt.
//*****
Label (Edit@)
If (Not CheckSelection (UserLink; hwndList))
    Return
EndIf
Index := LBGetCurSel (UserLink; hwndList)
NewLink := Array (LBGetItemData (UserLink; hwndList; Index); 1)
NewPrompt := LBGetWPTText (UserLink; hwndList; Index)
Label (DispEdit@)
SetWindowText (UserLink; hwndDlgEdit; "Edit Template Prompt")
DoneEditing := False
DialogShow (DlgEdit; PromptTxtMEditBx; CBEdit)
Repeat
Until (DoneEditing)
If (MacroDialogResult = OKMBtn)

```

```

    If (NewPrompt = LBGetWPText (UserLink; hwndList; Index))
        If (NewLink = Array (LBGetItemData (UserLink; hwndList; Index); 1))
            Return
        Else
            Go (ChangeLink@)
        EndIf
    EndIf
    Call (ErrorChecks@)
    If (ErrorExists)
        Go (DispEdit@)
    EndIf
    UpdateDoc (LBGetWPText (UserLink; hwndList; Index); NewPrompt)
    LBDeleteString (UserLink; hwndList; Index)
    LBInsertWPString (UserLink; hwndList; Index; NewPrompt)
    LBSetCurSel (UserLink; hwndList; Index)
    Label (ChangeLink@)
    LinkIndex := LookupLink (NewLink)
    LBSetItemData (UserLink; hwndList; Index; LinkIndex)
    ArraySet (LinkIndex; NewLink; False)
EndIf
Return
//*****

//*****
//    ROUTINE NAME: Delete@
//    INPUT VARIABLES: None
//    OUTPUT VARIABLES: None
//    USES: CheckSelection; UpdateDoc; LBGetCurSel; LBDeleteString
//    DESCRIPTION: Deletes the highlighted prompt.
//*****
Label (Delete@)
If (Not CheckSelection (UserLink; hwndList))
    Return
EndIf
Index := LBGetCurSel (UserLink; hwndList)
MessageBox (MsgBoxResult; "Confirm Deletion"; "Are you sure you want to delete ""^0""?";
IconQuestion! | YesNo! | HasParameters!; LBGetWPText (UserLink; hwndList; Index))
If (MsgBoxResult = 6) //YES
    UpdateDoc (LBGetWPText (UserLink; hwndList; Index); "")
    // Determine which prompt to highlight next
    If (Index = LBGetCount (UserLink; hwndList)-1)
        NewIndex := Index - 1
    Else
        NewIndex := Index
    EndIf
    LBDeleteString (UserLink; hwndList; Index)

```

```

        LBSetCurSel (UserLink; hwndList; NewIndex)
    EndIf
    Return
    /*******

    /*******
    //      ROUTINE NAME: Personal@
    //      INPUT VARIABLES:
    //      OUTPUT VARIABLES: None
    //      USES: Nothing
    //      DESCRIPTION: Displays personal field dialog.
    /*******

    Label (Personal@)
    PersFld := Array (2; 1)
    DonePasting := False
    DialogShow (DlgPers; PersDataListBx; CBPers@)
    While (not DonePasting)
        // loop until done with paste operation
    EndWhile
    DialogDismiss (DlgPers; CloseMBtn)
    Return
    /*******

    /*******
    //      ROUTINE NAME: MakePromptAbbr@
    //      INPUT VARIABLES: PromptAbbr
    //      OUTPUT VARIABLES: None
    //      USES: Nothing
    //      DESCRIPTION: Creates the "Template Prompts" abbreviation.
    /*******

    Label (MakePromptAbbr@)
    OnNotFound (SkipDelete@)
    AbbreviationDelete (PromptAbbr; CurrentDoc!)
    Label (SkipDelete@)
    OnNotFound ()
    NumOfPrompts := LBGetCount (UserLink; hwndList)
    ArraySet (1; 0; False) // Replace the <None> item in the array
    If (NumOfPrompts>0)
        AbbrString := LBGetWPText (UserLink; hwndList; 0) + "|" + Array (LBGetItemData
            (UserLink; hwndList; 0); 1)
        For (Count; 1; Count < NumOfPrompts; Count + 1)
            AbbrString := AbbrString + "|" + LBGetWPText (UserLink; hwndList; Count) +
                "|" + Array (LBGetItemData (UserLink; hwndList; Count); 1)
        EndFor
        AbbreviationCreate (PromptAbbr; CurrentDoc!; AbbrString)
    EndIf

```

Return

```
*****
*****
// ROUTINE NAME: BuildBookmarks@
// INPUT VARIABLES: ArrayCount; MarkAbbr
// OUTPUT VARIABLES: BookmarkString
// USES: Nothing
// DESCRIPTION: Builds bookmarks around prompts in document.
*****
Label (BuildBookmarks@)
If (?DocBlank)
    Go (Stop@)
EndIf
// Delete the bookmarks from the document so that renumbering works correctly
GetData (NumBookmarks; Bookmark!; Count!; CurrentDoc!)
For (MarkNumber; 1; MarkNumber <= NumBookmarks; MarkNumber + 1)
    BookmarkDelete (MarkNumber)
EndFor
// Create bookmarks for prompts
MarkNumber := 0
BookmarkString := ""
For (Count; 1; Count <= ArrayCount; Count + 1)
    PosDocVeryTop ()
    While (True)
        OnNotFound (NoMorePersonal@)
        SearchString (StrTransform ("<" + Array (Count; 1) + ">"; "-"; NtoC (0fb05x)))
        MatchSelection ()
        SearchNext (Extended!)
        MarkNumber := MarkNumber + 1
        BookmarkCreate (MarkNumber)
        BookmarkString := BookmarkString + Array (Count; 1) + "|"
    Next
    Label (NoMorePersonal@)
    OnNotFound ()
    While (?Substructure)
        SubDoc := ?CurrentSubDoc
        SubstructureExit ()
        If ((SubDoc = 10) Or (SubDoc = 11))
            BoxEnd (Save!)
        EndIf
    EndWhile
    Break
EndWhile
EndFor
If (BookmarkString = "")
```

```

BookmarkString := ""
Else
BookmarkString := BookmarkString + "|"
EndIf
PromptFound := False
For (Count; 1; Count <= LBGetCount (UserLink; hwndList); Count + 1)
PosDocVeryTop ()
While (True)
OnNotFound (NoMorePrompts@)
SearchString (StrTransform ("[" + LBGetWPTText (UserLink; hwndList; Count-1)
+ "]" ; "-" ; NToC (0fb05x)))
MatchSelection ()
SearchNext (Extended!)
PromptFound := True
MarkNumber := MarkNumber + 1
BookmarkCreate (MarkNumber)
BookmarkString := BookmarkString + Count + "|"
Next
Label (NoMorePrompts@)
OnNotFound ()
While (?Substructure)
SubDoc := ?CurrentSubDoc
SubstructureExit ()
If ((SubDoc = 10) Or (SubDoc = 11))
BoxEnd (Save!)
EndIf
EndWhile
Break
EndWhile
EndFor
If (PromptFound)
BookmarkString := SubStr (BookmarkString; 1; StrLen (BookmarkString)-1)
EndIf
Return
//*****

//*****
// ROUTINE NAME: MakeBookmarkAbbr@
// INPUT VARIABLES: BookmarkString; MarkAbbr
// OUTPUT VARIABLES: None
// USES: Nothing
// DESCRIPTION: Creates the "Bookmarks" abbreviation.
//*****

Label (MakeBookmarkAbbr@)
OnNotFound (DoNotDelete@)
AbbreviationDelete (MarkAbbr; CurrentDoc!)

```

```

Label (DoNotDelete@)
OnNotFound ()
If(BookmarkString<>"" )
    AbbreviationCreate (MarkAbbr; CurrentDoc!; BookmarkString)
EndIf
Discard (BookmarkString)
Return
//*****

//*****
//    ROUTINE NAME: DestroyDialogs@
//    INPUT VARIABLES: None
//    OUTPUT VARIABLES: None
//    USES: Nothing
//    DESCRIPTION: Destroys all macro dialogs.
//*****

Label (DestroyDialogs@)
DialogDestroy (DlgMain)
DialogDestroy (DlgEdit)
DialogDestroy (DlgPers)
Return
//*****

//*****
//    ROUTINE NAME: ErrorChecks@
//    INPUT VARIABLES: NewPrompt; MaxChars
//    OUTPUT VARIABLES: None
//    USES:
//    DESCRIPTION: Performs various error checks on new prompts.
//*****

Label (ErrorChecks@)
ErrorExists := False
// Blank Entry
If (NewPrompt = "")
    MessageBox (; "No Prompt Specified"; "You must enter text before pressing OK.";
    IconExclamation!)
    ErrorExists := True
    Return
EndIf
// Blank Link
If (NewLink = "")
    MessageBox (; "No Link Specified"; "You must choose a link to an address book field
before pressing OK."; IconExclamation!)
    ErrorExists := True
    Return
EndIf

```

```

// Invalid Character(s)
DelPos := StrPos (NewPrompt; "|")
If (DelPos <>0)
    MessageBox (; "Illegal Character"; "You cannot include the pipe character (vertical bar)
    in the prompt."; IconExclamation!)
    ErrorExists := True
    Return
EndIf
// Too Many Characters
PromptLen := StrLen (NewPrompt)
If (PromptLen>MaxChars)
    MessageBox (; "Too Many Characters"; "Your prompt cannot be longer than ^0
    characters."; IconExclamation! | HasParameters!; MaxChars)
    ErrorExists := True
    Return
EndIf
// Duplicate Prompts
For (Count; 0; Count <= LBGetCount (UserLink; hwndList)-1; Count + 1)
    If (ToUpper (NewPrompt) = ToUpper (LBGetWPTText (UserLink; hwndList; Count)))
        MessageBox (; "Duplicate Entry"; "This prompt already exists. Please choose
        another name."; IconExclamation!)
        ErrorExists := True
        Return
    EndIf
EndFor
Return
//*****

//*****
//    PROCEDURE NAME: UpdateDoc
//    INPUT VARIABLES: OldText; NewText
//    OUTPUT VARIABLES: None
//    USES: Nothing
//    DESCRIPTION: Updates the prompt fields in the document.
//*****
Procedure UpdateDoc (OldText; NewText)
TempMark := "_prompt.wcm"           // Sets name of temporary bookmark
Display (Off!)
BookmarkCreate (TempMark)
PosDocVeryTop ()
OnNotFound (RepReturn@)
SearchString (StrTransform ("[" + OldText + "]; "-"; NToC (0fb05x)))
If (NewText <> "")
    ReplaceString (StrTransform ("[" + NewText + "]; "-"; NToC (0fb05x)))
Else
    ReplaceString ("")

```



```

EndIf
ReplaceForward (Extended!)
While (?Substructure)
    SubDoc := ?CurrentSubDoc
    SubstructureExit ()
    If ((SubDoc = 10) Or (SubDoc = 11))
        BoxEnd (Save!)
    EndIf
EndWhile
Label (RepReturn@)
OnNotFound ()
BookmarkFind (TempMark)
BookmarkDelete (TempMark)
Display (On!)
EndProc
//*****

//*****
//    FUNCTION NAME: FindFile
//    INPUT: Name of file to search for
//    OUTPUT: Path of file or NULL string
//    USES: Nothing
//    DESCRIPTION: Locates a file in the template directories or the current directory.
//*****

Function FindFile (Name)
NewName := ?PathTemplate + Name
FileExists (FExists; NewName)
If (FExists)
    Return (NewName)
Else
    NewName := ?PathTemplateSupplemental + Name
EndIf
FileExists (FExists; NewName)
If (FExists)
    Return (NewName)
EndIf
FileExists (FExists; Name)
If (FExists)
    Return (Name)
EndIf
Return ("")
EndFunc
//*****

//*****
//    FUNCTION NAME: CheckQuickMark

```

```

//      INPUT: Nothing
//      OUTPUT: True if QuickMark exists, otherwise False
//      USES: Nothing
//      DESCRIPTION: Checks on the existence of a QuickMark.
//*****
Function CheckQuickMark ()
Existance := False
GetData (Num; Bookmark!; Count!; CurrentDoc!)
If (Num = 0)
    Return (Existance)
EndIf
For (Curr; 1; Curr <= Num; Curr + 1)
    GetData (CurrName; Bookmark!; Name!; CurrentDoc!; Curr)
    If (CurrName = "QuickMark")
        Existance := True
        Break
    EndIf
EndFor
Return (Existance)
EndFunc
//*****

//*****
//      PROCEDURE NAME: MoveItemUp
//      INPUT: Link; Hwnd; Index
//      OUTPUT: None
//      USES: LBGetCurSel; LBGetWPText; LBGetItemData; LBDeleteString;
LBInsertWPString; LBSetItemData; LBSetCurSel
//      DESCRIPTION: Moves the highlighted item up one location.
//*****
Procedure MoveItemUp (Link; Hwnd)
Index := LBGetCurSel (Link; Hwnd)
If (Index>0)
    MoveString := LBGetWPText (Link; Hwnd; Index)
    MoveData := LBGetItemData (Link; Hwnd; Index)
    LBDeleteString (Link; Hwnd; Index)
    Index := Index-1
    LBInsertWPString (Link; Hwnd; Index; MoveString)
    LBSetItemData (Link; Hwnd; Index; MoveData)
    LBSetCurSel (Link; Hwnd; Index)
Else
    Beep
EndIf
EndProc
//*****

```

```

*****
//      PROCEDURE NAME: MoveItemDown
//      INPUT: Link; Hwnd
//      OUTPUT: None
//      USES: LBGetCount; LBGetCurSel; LBGetWPText; LBGetItemData; LBDeleteString;
LBInsertWPString; LBSetItemData; LBSetCurSel
//      DESCRIPTION: Moves hilighted item down one location.
*****
Procedure MoveItemDown (Link; Hwnd)
ListCount := LBGetCount (Link; Hwnd)
Index := LBGetCurSel (Link; Hwnd)
If (Index < ListCount-1)
    MoveString := LBGetWPText (Link; Hwnd; Index)
    MoveData := LBGetItemData (Link; Hwnd; Index)
    LBDeleteString (Link; Hwnd; Index)
    Index := Index + 1
    LBInsertWPString (Link; Hwnd; Index; MoveString)
    LBSetItemData (Link; Hwnd; Index; MoveData)
    LBSetCurSel (Link; Hwnd; Index)
Else
    Beep
EndIf
EndProc
*****

*****
//      FUNCTION NAME: CheckSelection
//      INPUT: Link; Hwnd
//      OUTPUT: True if an item is selected.
//      USES: LBGetCurSel
//      DESCRIPTION: Determines if an item is currently highlighted in the listbox.
*****
Function CheckSelection (Link; Hwnd)
Highlight := True
If (LBGetCurSel (Link; Hwnd) < 0)
    MessageBox (; "No Prompt Selected"; "You must select an item before using this
option."; IconExclamation!)
    HighLight := False
EndIf
Return (HighLight)
EndFunc
*****

*****
//      FUNCTION NAME: LBGetCurSel
//      INPUT: Link; Hwnd

```

```

//      OUTPUT: Index of highlighted item.
//      USES: Nothing
//      DESCRIPTION: Gets the index of the currently highlighted item or -1 if none is
highlighted.
//*****
Function LBGetCurSel (Link; Hwnd)
DLLCall (Link; "SendMessageA"; nIndex: INTEGER; {Hwnd; 0188x; 0; 0})
Return (nIndex)
EndFunc
//*****

//*****
//      FUNCTION NAME: LBSetCurSel
//      INPUT: Link; Hwnd; Index
//      OUTPUT: None.
//      USES: Nothing
//      DESCRIPTION: Sets the selection in a listbox.
//*****
Procedure LBSetCurSel (Link; Hwnd; Index)
DLLCall (Link; "SendMessageA"; nIndex: INTEGER; {Hwnd; 0186x; Index; 0})
EndProc
//*****

//*****
//      PROCEDURE NAME: LBDeleteString
//      INPUT: Link; Hwnd; Index
//      OUTPUT: Number of items left in list or -1 if error.
//      USES: Nothing
//      DESCRIPTION: Deletes an item from a listbox.
//*****
Procedure LBDeleteString (Link; Hwnd; Index)
DLLCall (Link; "SendMessageA"; nNum: INTEGER; {Hwnd; 0182x; Index; 0})
EndProc
//*****

//*****
//      PROCEDURE NAME: LBInsertWPString
//      INPUT: Link; Hwnd; Index; NewString
//      OUTPUT: Position where string was inserted or -1 if error.
//      USES: Nothing
//      DESCRIPTION: Adds an item to a dialog list at the specified position.
//*****
Procedure LBInsertWPString (Link; Hwnd; Index; NewString)
DllCall (Link; "SendMessageA"; Ret: INTEGER; {Hwnd; 0797x; Index; WPString
(NewString)})
EndProc

```

```

//*****
//*****
//      FUNCTION NAME: LBGetItemData
//      INPUT: Link; Hwnd; Index
//      OUTPUT: Value associated with specified list item.
//      USES: Nothing
//      DESCRIPTION: Returns the value associated with the specified list item.
//*****
Function LBGetItemData (Link; Hwnd; Index)
DllCall (Link; "SendMessageA"; dwData: DWORD; {Hwnd; 0199x; Index; 0})
Return (dwData)
EndFunc
//*****

//*****
//      PROCEDURE NAME: LBSetItemData
//      INPUT: Link; Hwnd; Index; Data
//      OUTPUT: None
//      USES: Nothing
//      DESCRIPTION: Sets the value associated with the specified list item.
//*****
Procedure LBSetItemData (Link; Hwnd; Index; Data)
DllCall (Link; "SendMessageA"; RetVal: INTEGER; {Hwnd; 019Ax; Index; Data})
EndProc
//*****

//*****
//      FUNCTION NAME: LBGetCount
//      INPUT: Link; Hwnd
//      OUTPUT: Number of items in list box.
//      USES: Nothing
//      DESCRIPTION: Gets the number of items in the list box.
//*****
Function LBGetCount (Link; Hwnd)
DllCall (Link; "SendMessageA"; Count: INTEGER; {Hwnd; 018Bx; 0; 0})
Return (Count)
EndFunc
//*****

//*****
//      FUNCTION NAME: LBGetWPText
//      INPUT: Link; Hwnd; Index
//      OUTPUT: Text of specified list box item.
//      USES: Nothing
//      DESCRIPTION: Returns text of specified list item.

```

```
*****
Function LBGetWPText (Link; Hwnd; Index)
DllCall (Link; "SendMessageA"; Ret: INTEGER; {Hwnd; 0796x; Index; Address (WpString
(NewString))})
Return (NewString)
EndFunc
*****
```

```
*****
//      FUNCTION NAME: LBGetText
//      INPUT: Link; Hwnd; Index
//      OUTPUT: Text of specified list box item.
//      USES: Nothing
//      DESCRIPTION: Returns text of specified list item.
*****
```

```
Function LBGetText (Link; Hwnd; Index)
DllCall (Link; "SendMessageA"; Ret: INTEGER; {Hwnd; 0189x; Index; Address (AnsiString
(NewString))})
Return (NewString)
EndFunc
*****
```

```
*****
//      PROCEDURE NAME: SetWindowText
//      INPUT: Link; Hwnd; NewString
//      OUTPUT: None
//      USES: Nothing
//      DESCRIPTION: Sets the text of a specified window.
*****
```

```
Procedure SetWindowText (Link; Hwnd; NewString)
DllCall (Link; "SetWindowTextA"; Ret: Bool; {Hwnd; AnsiString (NewString)})
EndProc
*****
```

```
*****
//      PROCEDURE NAME: EnableWindow
//      INPUT: Link; Hwnd; Mode (1 = enable, 0 = disable)
//      OUTPUT: None
//      USES: Nothing
//      DESCRIPTION: Enables or disables the specified window.
*****
```

```
Procedure EnableWindow (Link; Hwnd; Mode)
DllCall (Link; "EnableWindow"; Ret: Bool; {Hwnd; DWord (Mode)})
EndProc
*****
```

```

//*****
//      PROCEDURE NAME: ArraySet
//      INPUT: Index; FieldName; FieldLink
//      OUTPUT: None
//      USES: Nothing
//      DESCRIPTION: Sets an entry in the dynamic array.
//*****
Procedure ArraySet (Index; FieldName; CanDelete)
If (Index > ArrayCount)
    ArrayCount := ArrayCount + 1
    If (Not Exists (Indirect ("DynArray1_" + Index); Global!))
        Global (Indirect ("DynArray1_" + Index))
        Global (Indirect ("DynArray2_" + Index))
    EndIf
EndIf
Assign (Indirect ("DynArray1_" + Index); FieldName)
Assign (Indirect ("DynArray2_" + Index); CanDelete)
EndProc
//*****

```

```

//*****
//      FUNCTION NAME: Array
//      INPUT: Index1; Index2
//      OUTPUT: The specified array element.
//      USES: Nothing
//      DESCRIPTION: Retrives an entry in the dynamic array.
//*****
Function Array (Index1; Index2)
Return (Indirect ("DynArray" + Index2 + "_" + Index1))
EndFunc
//*****

```

```

//*****
//      FUNCTION NAME: LookupLink
//      INPUT: LinkName
//      OUTPUT: The location of this link in the dynamic array.
//      USES: Nothing
//      DESCRIPTION: Look up the location of this link in the dynamic array.
//*****
Function LookupLink (LinkName)
For (x; 1; x <= ArrayCount; x + 1)
    If (ToUpper (Array (x; 1)) = ToUpper (LinkName))
        // Mark this entry as non-deletable
        ArraySet (x; Array (x; 1); False)
        Return (x)
    EndIf
EndIf

```

```

EndFor
// The link wasn't found, so add it to the array
ArraySet (ArrayCount + 1; LinkName; False)
// Now add it to the lists
If(LinkName<>"0") //Prompt is linked to an abbr field
    DialogAddListItem (DlgEdit; AbkLinkMCombo; LinkName)
    DialogAddListItem (DlgPers; PersDataListBx; LinkName)
Else
    Return (1) //Set list item to <none>
EndIf
Return (ArrayCount)
EndFunc
//*****

//*****
// PROCEDURE NAME: AddAllFields
// INPUT: None
// OUTPUT: None
// USES: ArraySet; Array
// DESCRIPTION: Add all possible fields into the dynamic array.
//*****

Procedure AddAllFields (Region)
ShowAllFields := 1
CurrentSelection := RegionGetSelectedText (Region)
WaitMsgDisplay ("Retrieving fields...")
// For each available Address Book
For (x; AddressBookGetCount (); x >= 1; x - 1)
    ABName := AddressBookGetName (x)
    // Get each field from the Address Book
    For (y; AddressBookGetFieldCount (ABName); y >= 1; y - 1)
        FieldName := AddressBookGetFieldName (ABName; y)
        //Exclusions 990730NP
        if (FieldName <> "Hidden" and FieldName <> "Read Only" and FieldName <>
            "Owner" and FieldName <> "Resource Type" and FieldName <> "E-mail
            Type")
            // Check if this field is already in the dynamic array
            Found := False
            For (z; 1; z <= ArrayCount; z + 1)
                If (ToUpper (Array (z; 1)) = ToUpper (FieldName))
                    // It was found, so go on
                    Found := True
                    Break
                EndIf
            EndFor
        EndFor
        If (Not Found)
            // The link wasn't found, so add it to the array

```



```

        ArraySet (ArrayCount + 1; FieldName; True)
        // Now add it to the lists
        DialogAddListItem (DlgEdit; AbkLinkMCombo; FieldName)
        DialogAddListItem (DlgPers; PersDataListBx; FieldName)
    EndIf
EndIf
EndFor
EndFor
WaitMsgHide ()
RegionSelectListItem (Region; CurrentSelection)
EndProc
/*****
/*****
//    PROCEDURE NAME: RemoveExtraFields
//    INPUT: Region
//    OUTPUT: None
//    USES: Array
//    DESCRIPTION: Remove extra fields from dynamic array.
/*****
Procedure RemoveExtraFields (Region)
ShowAllFields := 0
EndField := 1
CurrentSelection := RegionGetSelectedText (Region)
WaitMsgDisplay ("Removing extra fields...")
For (x; 1; x <= ArrayCount; x + 1)
    If ((Array (x; 2) = False) Or (ToUpper (CurrentSelection) = ToUpper (Array (x; 1))))
        // This field cannot be removed
        If (EndField <> x)
            ArraySet (EndField; Array (x; 1); Array (x; 2))
            EndField := EndField + 1
        Else
            EndField := x + 1
        EndIf
    EndIf
EndFor
ArrayCount := EndField - 1
// Refill the lists
RegionResetList (DlgEdit + " ." + ABkLinkMCombo)
RegionResetList (DlgPers + " ." + PersDataListBx)
DialogAddListItem (DlgEdit; AbkLinkMCombo; Array (1; 1))
For (Count; 2; Count <= ArrayCount; Count + 1) // skip the "<none>" entry
    DialogAddListItem (DlgPers; PersDataListBx; Array (Count; 1))
    DialogAddListItem (DlgEdit; AbkLinkMCombo; Array (Count; 1))
EndFor
RegionSelectListItem (Region; CurrentSelection)
WaitMsgHide ()

```

EndProc

```
/**
*****

```

```
/**
*****

```

```
// ROUTINE NAME: InitVariables@
```

```
// INPUT VARIABLES: None
```

```
// OUTPUT VARIABLES: Many
```

```
// USES:
```

```
// DESCRIPTION: Initializes the variables we will use
```

```
/**
*****

```

Label (InitVariables@)

```
If (Exists (PromptsMacroRunning) <>3) // has this persist variable been defined
```

```
    PERSIST PromptsMacroRunning // indicate PROMPTS.WCM running
```

```
    PromptsMacroRunning := True
```

```
EndIf
```

```
// initialize variables
```

```
PromptAbbr := "Template Prompt Info" // Sets abbreviation name for prompts
```

```
MarkAbbr := "Template Bookmark Info" // Sets abbreviation name for bookmarks
```

```
ObjectFilename := "_autotmp.wpx" // Name of sample automated template
```

```
ObjectName := "<dofiller>" // Name of automation macro
```

```
MaxChars := 60 // Maximum number of prompt characters
```

```
MaxPrompts := 64 // Maximum number of prompts
```

```
Global ArrayCount := 0 // Number of entries in the Dynamic array
```

```
Global ShowAllFields := 0 // Indicates if we are showing all possible  
fields
```

```
BookmarkString:=""
```

```
// Define address book links
```

```
ArraySet (1; "<None>"; False)
```

```
ArraySet (2; "Anniversary"; False)
```

```
ArraySet (3; "Assistant's Name"; False )
```

```
ArraySet (4; "Assistant's Phone"; False )
```

```
ArraySet (5; "Birthday"; False)
```

```
ArraySet (6; "Business Comments"; False)
```

```
ArraySet (7; "Business Fax"; False)
```

```
ArraySet (8; "Business Phone"; False)
```

```
ArraySet (9; "Business Phone 2"; False)
```

```
ArraySet (10; "Cellular Phone"; False)
```

```
ArraySet (11; "City"; False )
```

```
ArraySet (12; "Country"; False )
```

```
ArraySet (13; "Created By"; False )
```

```
ArraySet (14; "Created On"; False )
```

```
ArraySet (15; "Department"; False)
```

```
ArraySet (16; "Display Name"; False)
```



```

Global MoveUpMBtn := "MoveUpMBtn" // Region name for the Move Up
button
Global MoveDnMBtn := "MoveDnMBtn" // Region name for the Move Down
button
Global OKMBtn := "OKMBtn" // Region name for the OK button
Global PasteMBtn := "PasteMBtn" // Region name for the Paste button
Global PersDataListBx := "PersonalDataMListBx" // Region name for the Personal data
list
Global PersonalMBtn := "PersonalMBtn" // Region name for the Personal
button
Global PromptTxtMEditBx := "PromptTextMEditBx" // Region name for the Prompt edit
bx
Global PromptlistMListBx := "PromptlistMListBx" // Region name for the Prompt list
Return
//*****

//*****
// PROCEDURE: WaitMsgInit ()
// PURPOSE: Initializes macro wait messages.
//*****
PROCEDURE WaitMsgInit ()
DialogDefine ("WaitMsg"; 50; 50; 150; 36; 16; "Please Wait")
DialogAddText ("WaitMsg"; "WaitText"; 8; 14; 144; 16; 1; "")
DialogLoad ("WaitMsg")
Return
ENDPROC

//*****
// PROCEDURE: WaitMsgDisplay (Text)
// PURPOSE: Displays a previously defined wait message dialog.
//*****
PROCEDURE WaitMsgDisplay (Text)
RegionSetWindowText ("WaitMsg" + ".WaitText"; Text)
DialogShow ("WaitMsg"; WaitDlgCallBack)
ENDPROC

PROCEDURE WaitDlgCallBack ()
If (WaitDlgCallBack[5] = 274 And WaitDlgCallBack[6] = 61536)
Assert (CancelCondition!)
EndIf
ENDPROC

//*****
// PROCEDURE: WaitMsgHide
// PURPOSE: Hides a previously defined wait message dialog.
//*****

```

```
PROCEDURE WaitMsgHide ()
DialogDismiss ("WaitMsg"; "WaitText")
ENDPROC
```

```
*****
//      PROCEDURE: WaitMsgDestroy
//      PURPOSE: Destroys a previously defined wait message dialog.
*****
PROCEDURE WaitMsgDestroy ()
DialogDestroy ("WaitMsg")
ENDPROC
*****
```

```
Label (End@)
```

```
If( SmartQuotesWereActivated )
    // Turn Single Smart Quotes back on if WE turned them off for this script
    SingleSmartQuote (State: On!)
Endif
```

```
Discard PromptsMacroRunning          // indicate PROMPTS.WCM not running
Quit
```