Localizing Applications for the Handspring™ Visor

Application Note AN-15 Revision 1.00 Document No. 80-0127-00

Table of Contents

1. Introduction

To support a standardized method for localizing Palm® OS applications, Palm OS 3.5 introduced the overlay manager. The overlay manager allows developers to localize their applications by dividing localized resources into separate resource databases. When the application runs, the overlay manager searches for and opens the appropriate localized resource database to match the country and language IDs of the device's locale.

Refer to www.palmos.com/dev for more information on localizing. Below is an excerpt from Palm's knowledge base for localization. (http://oasis.palm.com/dev/kb/manuals/1747.cfm)

> **Using Overlays to Localize Resources**
>
> *Palm OS version 3.5 adds support for localizing resource databases through **overlays**. Localization overlays provide a method for localizing a software module without requiring a recompile or modification of the software. Each overlay database is a separate resource database that provides an appropriately localized set of resources for a single software module (the PRC file, or **base database**) and a single target **locale** (language and country).*

For developers requiring Palm OS 3.5 and above, Handspring recommends that you follow the Palm OS 3.5 overlay manager provision.

For developers requiring Palm OS 3.1 support and above, Handspring has provided a custom overlay utility to imitate the overlay manager (in Palm OS 3.5).

The Handspring overlay utility implementation is compatible with Palm OS 3.5 (e.g., it will execute properly in Palm OS 3.5) but is not identical to overlay manager. The concept used in Handspring's overlay mechanism is basically the same as in Palm's. This means that the major effort involved in dividing and localizing the resources can be used in either Handspring or Palm's OS 3.5 method. The main differences between Handspring's overlay

utility and Palm OS 3.5's overlay manager are how the localized resource databases are named (with different creator ID and type) and how the localized resource databases are searched for and loaded.

Specific differences between the two overlay methods:

- Handspring's method requires the application to explicitly load the overlays (by calling library functions) whenever they are needed. This means that the application must be modified to load overlays at the right time (usually this is just when the application is run or for any sublaunches that display UI).
- The Handspring method does not allow for localization of an application's Launcher name. This stems from the fact that the application is in charge of loading its own overlays. The application is never called when the launcher is looking for its name, so there is no way for it to provide a localized name.

Note that applications should only implement one method. If applications implement both methods on the same device, the methods will conflict.

2. Handspring's Overlay Utility

The Handspring overlay utility is implemented as a set of two APIs that will search and open localized resource databases specifically named with the Handspring overlay naming convention. These APIs are not present in the Handspring extensions, but instead are provided as a library that is statically linked into your application.

`DmOpenRef  HsUtilOverlayInitialize (CharPtr baseStrP)`

will search and open the application's localized resource database overlay. The localized resource database overlay must be built with the Handspring naming convention.

The convention uses the prefix of the application's program name concatenated with the suffix of `Rsc_`*`CountryCode`*`_`*`LanguageCode`*, where *`CountryCode`* and *`LanguageCode`* are the numeric ID of the current device. For example, a sample application named `SampleLocalization` will have a localized resource database overlay named `SampleLocalizationRsc_7_1`,where `7` is the device's country code for France and `1` is the device's language code for French. The country code and language code defines are present in Palm's header files.

Thus when the main target application calls `HsUtilOverlayInitialize("SampleLocalization")`, `HsUtilOverlayInitialize("SampleLocalization")` will search for and open the overlay named `SampleLocalization_7_1` when on a French localized device. `HsUtilOverlayInitialize` will search for and open the overlay named `SampleLocalization_8_2` when on a German-localized device.

`void HsUtilOverlayCleanup (DmOpenRef openRefP)`

will close the localized resource database overlay that was opened by `HsUtilOverlayInitialize()`.

Once the appropriate localized resource database has been opened, the application will retrieve the correct localized resources.

Note: it is always possible that no valid overlay will be found. This can happen if more localized devices are released. We recommend that you include a default language directly in the base PRC. That way, if no overlay is found your application will still run.

3. Building the Localized Resource Database Overlay

Because the Handspring overlay utility relies on a custom Handspring overlay naming convention, the overlays must be built to this convention.

The convention uses the prefix of the application's program name concatenated with the suffix of `Rsc_CountryCode_LanguageCode` where *`CountryCode`* and *`LanguageCode`* are the numeric country/language IDs of the current device's locale.

Below is a more detailed example of the naming relationship between the application and its overlays.

This example will select the name `SampleLocalization` as the project name.

All the localized overlays built with Palm-RC must use this name `SampleLocalization` as the prefix concatenated with `Rsc_`*`CountryCode`*`_`*`LanguageCode`* where *`CountryCode`* and *`LanguageCode`* are the numeric ID of desired localization. Below shows some different overlays build commands.

```
Palm-RC                                                    \
      -rcp FrenchSubDirectory\SampleLocalization.rcp       \
      -o SampleLocalizationFrenchVersionOverlay.prc        \
      -name SampleLocalizationRsc_7_1                      \
      -cr LCLZ    -type HsOl                               \

Palm-RC                                                    \
      -rcp germanSubDirectory\SampleLocalization.rcp       \
      -o SampleLocalizationGermanVersionOverlay.prc        \
      -name SampleLocalizationRsc_8_2                      \
      -cr LCLZ    -type HsOl                               \
```

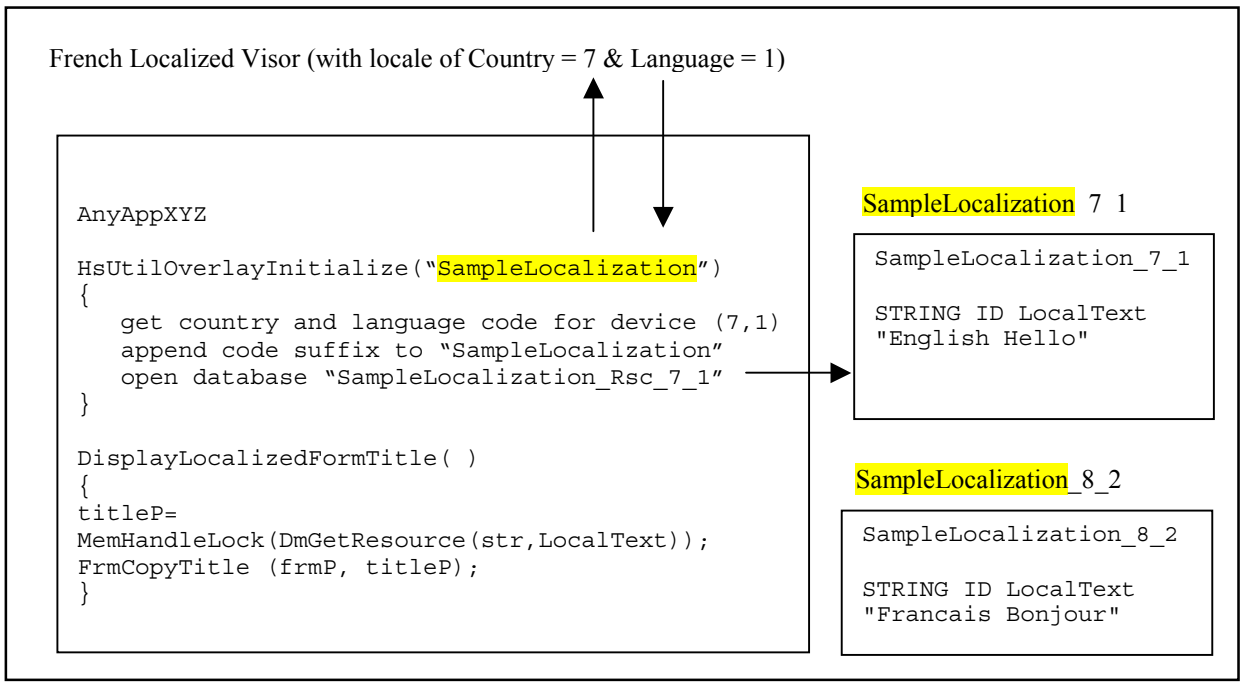
The main target application can optionally choose a different name such as `AnyNameXYZ` when built with Palm-RC, though it is recommended that you keep the two names the same.

```
Palm-RC \
      -rcp USdefault\ SampleLocalization.rcp               \
      -gccApp "AnyNameABC.1.sym"                           \
      -o AnyNameApp                                        \
      -name "AnyNameXYZ"                                   \
      -cr LCLZ    -type Appl                               \
```

The table below shows a listing of the different files and applications and their names.

Source filename on PC	PRC name on PC	PRC "database" name on Visor
`FrenchSubDirectory\ SampleLocalization.rcp`	`SampleLocalizationFrenchVersionOverlay.prc`	`SampleLocalizationRsc_7_1`
`GermanSubDirectory\ SampleLocalization.rcp`	`SampleLocalizationGermanVersionOverlay.prc`	`SampleLocalizationRsc_8_2`
`AnyNameABC`	`AnyNameApp.prc`	`AnyNameXYZ`

To summarize the Handspring overlay utility and naming convention, only the prefix name the localized overlays' database names must match the parameter name of `HsUtilOverlayInitialize()`. The below diagram illustrates the necessary relationship.



Note that when building the ROM image, each localized resource database overlay should be included in the ROM build using Palm-MakeROM –romDB.

```
Palm-MakeROM
       -romDB SampleLocalizationFrenchVersionOverlay.prc
       -romDB SampleLocalizationGermanVersionOverlay.prc
```

4. Handspring's Language Spoofer Utility

Handspring's implementation of the overlay utility contains a useful feature to aid in testing of localized applications. The `HsUtilOverlayInitialize()` API implementation first checks for Handspring's "feature" country code and language code overrides of the ROM's default country code and language code. If the Handspring's feature override for the country code or language code does not exist, the ROM's default country code and language code will be retrieved. This override provision allows for test applications such as Handspring's Language Spoofer to dynamically alter the feature setting for the Visor's country code and language code. Thus the test application allows for testing of all the different localizations (except Kanji base languages) on the Visor by changing its country and language code features.

The Language Spoofer application and source code is available on the Handspring website.

5. Sample Code

This section contains the full sample source for a simple Hello World project with English default and French localization.

```
///////////////     SampleLocalization.c file        ///////////////

/*********************************************************************
 *
 *  Project:
 *       Handspring Sample PalmOS Application
 *
 * Copyright info:
 *
 *       This is free software; you can redistribute it and/or modify
 *       it as you like.
 *
 *       This program is distributed in the hope that it will be useful,
 *       but WITHOUT ANY WARRANTY; without even the implied warranty of
 *       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 *
 *  FileName:
 *       SampleLocalization.c
 *
 *  Description:
 *       This is the main source file for the sample application. It is
 *       derived from the "Tex 2 Hex" 1.0 sample source written by
 *       Andrew Howlett (howlett@iosphere.net) provided with his
 *       ASDK tutorial.
 *
 * ToDo:
 *
 * History:
 *       10-Feb-1999  - Created
 ********************************************************************/

#include <PalmOS.h>
#include <CoreCompatibility.h>
#include "SampleLocalization.h"

#include <OverlayUtils.h>



// Allow VC++ to compile...
#if defined(_MSC_VER)
  #define OVERLAY_BASENAME ""
#endif


// ---------------------------------------------------------------
// Prototypes
// ---------------------------------------------------------------
static int          StartApplication (void);
static void         EventLoop (void);
static void         StopApplication(void);
static Boolean      SampleLocalization (EventPtr event);
static void         DisplayBitmap (void);
```

```
// ------------------------------------------------------------
// Globals
// ------------------------------------------------------------
FieldPtr            FieldTextP;


// ============================================================================
// Pilot Main
// ============================================================================
DWord
PilotMain (Word cmd, Ptr cmdPBP, Word launchFlags)
{
  int error;

  if (cmd == sysAppLaunchCmdNormalLaunch)
       {
         DmOpenRef overlayDb = NULL;

         // Open overlay...
         overlayDb = HsUtilOverlayInitialize (OVERLAY_BASENAME);

         error = StartApplication();     // Application start code
         if (error) return error;

         EventLoop();                    // Event loop

         StopApplication ();             // Application stop code

       if (overlayDb != NULL)
         {
           HsUtilOverlayCleanup (overlayDb);
         }

       }

  return 0;
}



// ============================================================================
// Start Application
// ============================================================================
static int
StartApplication(void)
{

  FrmGotoForm (resFormIDSampleLocalization);
  return 0;
}


// ============================================================================
// Event Loop
// ============================================================================
static void
EventLoop(void)
{
  short       err;
  int         formID;
  FormPtr     formP;
  EventType   event;
```

```
  do
       {

         EvtGetEvent (&event, sysTicksPerSecond/2);

         if (SysHandleEvent (&event)) continue;
         if (MenuHandleEvent ((void *)0, &event, &err)) continue;

         if (event.eType == frmLoadEvent)
              {
                formID = event.data.frmLoad.formID;
                formP = FrmInitForm (formID);
                FrmSetActiveForm (formP);

                switch (formID)
                     {
                     case resFormIDSampleLocalization:
                       FrmSetEventHandler (formP, (FormEventHandlerPtr)
SampleLocalization);
                       break;
                     }
              }

         FrmDispatchEvent(&event);

        } while(event.eType != appStopEvent);
}


// ====================================================================
// Stop Application
// ====================================================================
static void
StopApplication (void)
{
  FldSetTextHandle (FieldTextP, NULL);
}



// ====================================================================
// Form Event Handler
// ====================================================================
static Boolean
SampleLocalization (EventPtr event)
{
  FormPtr    formP;
  VoidHand   strH;
  int        handled = 0;

  switch (event->eType)
       {
       case frmOpenEvent:
         formP = FrmGetActiveForm();

         strH = DmGetResource (strRsc, resTextIDCommon);
         FieldTextP = FrmGetObjectPtr (formP, FrmGetObjectIndex (formP,
resFieldIDCommon));
         FldSetTextHandle (FieldTextP, strH );
         DmReleaseResource (strH);

         strH = DmGetResource (strRsc, resTextIDLocalized);
```

```
        FieldTextP = FrmGetObjectPtr (formP, FrmGetObjectIndex (formP,
resFieldIDLocalized));
        FldSetTextHandle (FieldTextP, strH );
        DmReleaseResource (strH);


        DisplayBitmap();

        FrmDrawForm(formP);
        handled = 1;
        break;


      default:
        break;
      }
  return handled;
}




// ================================================================
// ChangeBitmap
// ================================================================
static void
DisplayBitmap(void)
{
  VoidHand     bitmapH;
  BitmapPtr    bitmapP;


  bitmapH = DmGet1Resource('Tbmp', resBitmapIDLocalized);

  if (!bitmapH) return;

  bitmapP = MemHandleLock (bitmapH);
  WinDrawBitmap (bitmapP, 20, 80);
  MemHandleUnlock (bitmapH);
}
```

```
///////////////      English default resource file    /////////////////

#include "SampleLocalization.h"

// Common resource section
VERSION 1 "1.0"
ICON "SampleLocalization.bmp"
STRING ID resTextIDCommon  "Handspring Visor"



// Localized resource section

FORM ID resFormIDSampleLocalization AT ( 0 0 160 160 )
NOFRAME
USABLE
BEGIN
    TITLE "English Localized Application"
    FIELD ID resFieldIDCommon AT (20 30 80 20 ) USABLE LEFTALIGN FONT 0 NONEDITABLE
                          MULTIPLELINES MAXCHARS 90
    FIELD ID resFieldIDLocalized AT (40 60 60 20 ) USABLE LEFTALIGN FONT 0 NONEDITABLE
                          MULTIPLELINES MAXCHARS 90
END


BITMAP ID resBitmapIDLocalized "localizedbitmap.bmp"
STRING ID resTextIDLocalized "English Text"
```

```
///////////////      English default resource file    /////////////////

#include "SampleLocalization.h"

// Common resource section
VERSION 1 "1.0"
ICON "SampleLocalization.bmp"
STRING ID resTextIDCommon  "Handspring Visor"



// Localized resource section

FORM ID resFormIDSampleLocalization AT ( 0 0 160 160 )
NOFRAME
USABLE
BEGIN
    TITLE "Application en Francais"
    FIELD ID resFieldIDCommon AT (20 30 80 20 ) USABLE LEFTALIGN FONT 0 NONEDITABLE
                          MULTIPLELINES MAXCHARS 90
    FIELD ID resFieldIDLocalized AT (40 60 80 20 ) USABLE LEFTALIGN FONT 0 NONEDITABLE
                          MULTIPLELINES MAXCHARS 90
END


BITMAP ID resBitmapIDLocalized "localizedbitmap.bmp"
STRING ID resTextIDLocalized "Texte Francais"
```

```
///////////////           Makefile                  /////////////////

######################################################################
#
# Makefile for SampleLocalization
#
# Targets:
#            all                 -      build and install
#            clean         -      clean everything
#            check         -      run self-tests
#
# This makefile uses the following environment variables:
#
#   PALMTOOLSU - path to directory where palm tools should go.
#       ex: "/usr/local/PalmTools"
#
#
# Conventions:
#       "ALLUPPER"  case variable names are generally exported or provided by
#                                  the environment
#       "initLower" case variables are local to the makefile
#
#
# Commonly used  Automatic Variables:
#        $< represents the first dependency
#        $^ represents all dependencies
#        $? represents all dependencies that are newer than the target
#        $@ represents the target
#
#
######################################################################


# ====================================================
# Include the common Make variables for PalmOS executables
# ====================================================
# This one is constant
include $(PALMTOOLSU)/bin/Palm-DefaultVars.make

# This one can be modified for different build options
#  and is optional
-include $(PALMTOOLSU)/bin/Palm-CustomVars.make


# The shell to use
SHELL          = /bin/sh


# Compiler Flags
CFLAGS              = $(PalmCFlags) -I$(rscDir) -DOVERLAY_BASENAME=\"$(progName)\"


# Directory paths
objDir         = ../Obj
srcDir         = ../Src
utilsDir       = ../Src/Utils
rscDir         = ../Rsc
testsDir       = ../Tests

# Target info
progName       = SampleLocalization
progDBName     = "Localized App"
progCreator    = lclz
```

```
progPrcFile   = $(progName)App.prc
progTestFile  = $(progName)Test.prc


# These are fake targets used to perform certain actions. Use the .PHONY
#  command to make sure they don't get confused with actual filenames
.PHONY : all install clean help

# Set the default install directory if not passed down to us
#  from the parent makefile
ifndef InstallDir
  InstallDir = $(PALMTOOLSU)/bin/Device
endif


###############################################################
# Master Builds
###############################################################
all:   $(objDir)/$(progPrcFile)
       $(MAKE) install

install:
       cp $(objDir)/$(progPrcFile) $(InstallDir)

clean:
       rm -f $(objDir)/*

check:
       cmp $(objDir)/$(progTestFile) $(testsDir)/$(progPrcFile)



###############################################################
# Implicit build rules
###############################################################
$(objDir)/%.o :  $(srcDir)/%.c
       $(CC) $(CFLAGS) -c $< -o $@




###############################################################
# Program Build
###############################################################

# -------------------------------------------------------------
# Compile objects
# -------------------------------------------------------------
hdrList = $(srcDir)/SampleLocalization.h       \
               $(utilsDir)/OverlayUtils.h      \
               $(utilsDir)/DmUtils.h



objList = $(objDir)/SampleLocalization.o       \
               $(utilsDir)/OverlayUtils.o      \
               $(utilsDir)/DmUtils.o

$(objDir)/SampleLocalization.o:                \
               $(srcDir)/SampleLocalization.c  \
               $(utilsDir)/OverlayUtils.c      \
               $(utilsDir)/DmUtils.c           \
               $(hdrList)
```

```
# ------------------------------------------------------------
# Link and combine with resources.
# ------------------------------------------------------------
$(objDir)/$(progPrcFile): $(objList) $(rscDir)/English/$(progName).rcp
	$(CC) $(CFLAGS) $(objList) -o $(objDir)/$(progDBName).code.1.sym
	$(PALMRC) -rcp $(rscDir)/English/$(progName).rcp     \
		-gccApp $(objDir)/$(progDBName).code.1.sym     \
		-I $(srcDir) -I $(rscDir) -I $(rscDir)/English \
		-o $(objDir)/$(progPrcFile)                    \
		-name $(progDBName) -cr $(progCreator)

	# The $(progTestFile) PRC has zero creation and mod dates for comparison
	# purposes
	$(PALMRC) -rcp $(rscDir)/English/$(progName).rcp     \
		-gccApp $(objDir)/$(progDBName).code.1.sym     \
		-I $(srcDir) -I $(rscDir) -I $(rscDir)/English \
		-o $(objDir)/$(progTestFile)                   \
		-name $(progDBName) -cr $(progCreator)         \
		-zCrDate -zModDate


# ------------------------------------------------------------
# Overlays
# ------------------------------------------------------------

	Palm-RC                                                        \
		-rcp $(rscDir)/French/SampleLocalization.rcp               \
		-I $(srcDir) -I $(rscDir) -I $(rscDir)/French              \
		-o $(objDir)/SampleLocalizationFrenchVersionOverlay.prc    \
		-name SampleLocalizationRsc_7_1                            \
		-cr LCLZ     -type HsOl                                    \
		-version 1 -copyPrevention -hidden

	ls -l $(objDir)/*.prc
```

6. History

Date	Revision #	Description of changes
15 Jan 01	1.00	Initial release.