MUI work for WLPG

Santosh Chapaneri

1) Modify the Base String and ResourceString classes to load the string using Resource Manager's loading mechanism. Provide a fall-back mechanism to load from the old resource location if loading via RM fails (this is a temporary solution and needs to be removed when MUI work is completed).

2) Replace all Win32 resource loading API calls with the RM equivalent (eg. LoadImage -> RMLoadImage)

3) Modify ATL Dialog Template ATL::CDialogImpl to use the RM equivalent calls, i.e. RMDialogBoxParam and RMCreateDialogParam instead of DialogBoxParam and CreateDialogParam. But this work involves rewriting most of the CDialogImpl and CDialogImplBase code, and replacing the usage of ATL class with the new class (which may be tricky for some internal ATL usages). To avoid this, we will just set the resource instance for the appropriate resource set and leave the dialog code as it is. This can be done as: _AtlBaseModule.SetResourceInstance(::RMFindModule(Resource Set, RMALL));

4) The Base LoadString calls will need to know the resource set to be used before it can call RM load function. Instead of passing the required resource set to LoadString, we will declare an extern variable at the module level (e.g, where the DllMain is) that defines which resource set is currently in use. This will avoid modifying a lot of client code.

5) At the module level, call ::RMUpdateResourceSet(Resource Set, Exe/DLL base name, RMCore | RMFMUI); Initialize SmartDuiProcessInit and SmartDuiThreadInit as required.

It was assumed that we will need only one resource set throughout the app to do this. However, RM API does not append the resource set for different Dlls/EXEs. Due to the resource set being overwritten, we now need to have different resource sets, one per each DLL/EXE. Also, it was found that we cannot have two resource sets per DLL/EXE. This will require us to move some of our resources and change their IDs.

6) Modify all the sources.inc and precompiled header files which depend on BaseString. Typically, this will be #include <rmmanagement.h> and definition of resource set in the header file; and including the UXPLATFORM_REF_PATH and UXCore.lib in the sources file.

7) Convert all applicable .rc files to .rcxml format. There are 79 resource files that have to be converted. The list is available at \\dmx\public\sachapan\MUI\RC2RCXML.xlsx

ABrodie has written a tool for doing this conversion: personalmedia\tools\rc2rcxml.cmd. However, there are certain cases where some of the resources in the .rc file do not get converted to .rcxml. We need to manually verify all such cases. Out of the 79 resource files, there are 9 Version.rc files which will be removed since the corresponding rcxml file will have to include the versioning information. From the remaining, about 36 resource files will need manual verification.

Due to the need of one resource set per DLL/EXE as mentioned in 5 above, we will have to merge SharedPhotoLibraryResources with PhotoLibraryDuiResources, so that we can have only one

::RMUpdateResourceSet() call in PhotoLibraryMain. Because of this, we need to change some of the resource ID's which will impact Localization and Test Automation.

8) Setup and Build changes. Have a common rcxml folder in personalmedia having resmakefile.inc and resources.inc which call RCXML.exe. For each RC to RCXML conversion, the sources.inc should have BUILD_PASS0_PRODUCES which will produce the required rcxml resource file and the resource header file. The other sources.inc files (such as in Binaries\ or PhotosUnitTest\) should have BUILD_PASS0_CONSUMES in order to use the generated rcxml resources. The existing resource.h file in each folder needs to be purged or modified to remove the IDs used in RCXML file. For example, in AlbumDownloadWorker.cpp, instead of #include "resource.h", we will now have #include "AlbumDownloadWizardResources.h" which is generated by RCXML.exe. All the resources that cannot go into the RCXML file (such as DaVinci graph specification, etc.) needs to be moved into main.rc. The sources.inc files will specify the RC_FILENAME as this main.rc. Set MUI=1 in the sources file to enable MUI.

MUI conversion for AlbumDownloadWizard: \\dmx\public\sachapan\MUI\albumdownload.cmd -w

We have completed the tasks till step 6, and will be resuming the MUI work from step 7 and 8 onwards. Steps 7 and 8 should happen at the same time while converting each folder in personalmedia\.