Kaizen: Thoughts on UI testing

Leveraging Resource UI verification

Jo Gel Santiago
GS LCM

Rev#: 1.0

Table of Contents

Introduction

GS test team is focused on verifying the components UI of the localized product components in their corresponding international platform. UI related testing primarily takes most of the testing coverage for the whole project duration. Considering the number of language platforms the product has to be tested on, testing cost increase incrementally with number of supported localized components in the product suite. To save cost while at the same time widening the range of testing coverage, software testing automation is becoming to be a necessity. Automation tools like Monkey UI testing have been developed and put in place; and found to be useful in finding UI related bugs like truncations, and especially errors which almost are not recognizable to human testers like hidden controls. Other tester utilities like ResourceValidator and GUIzard also proved to be helpful in verifying low-level UI verification on behalf of the tester during testcase execution.

Most of the mentioned tools however, require a working runtime environment i.e. they need to be used on the system where the product is actually installed and working properly. This dependency sometimes becomes a bottleneck or could possibly block the test execution when the product under test could not be installed like in cases of an installer issues, or in cases when some primary functionalities are still left to be fixed. It would be very desirable to be able to run UI related test *statically* without having the need for the running the actual testcase scenarios to display and check on the UI object.

G11N & Resource Files

The process of globalizing a product involves the task of localizing the culture and locale specific UI elements according to the target language platform. There are many ways and formats in externalizing the application's UI elements; Most Windows-based application makes use of Windows resource file formats - .rc, .mc, .resx file formats, properties file in Java-based implementation and several other file formats like XML or plain text files. Windows-based Citrix product components like XenApp and Streaming client use compiled satellite resource UI binaries. The server product DVD installer (autorun.exe) as well utilizes the same framework.

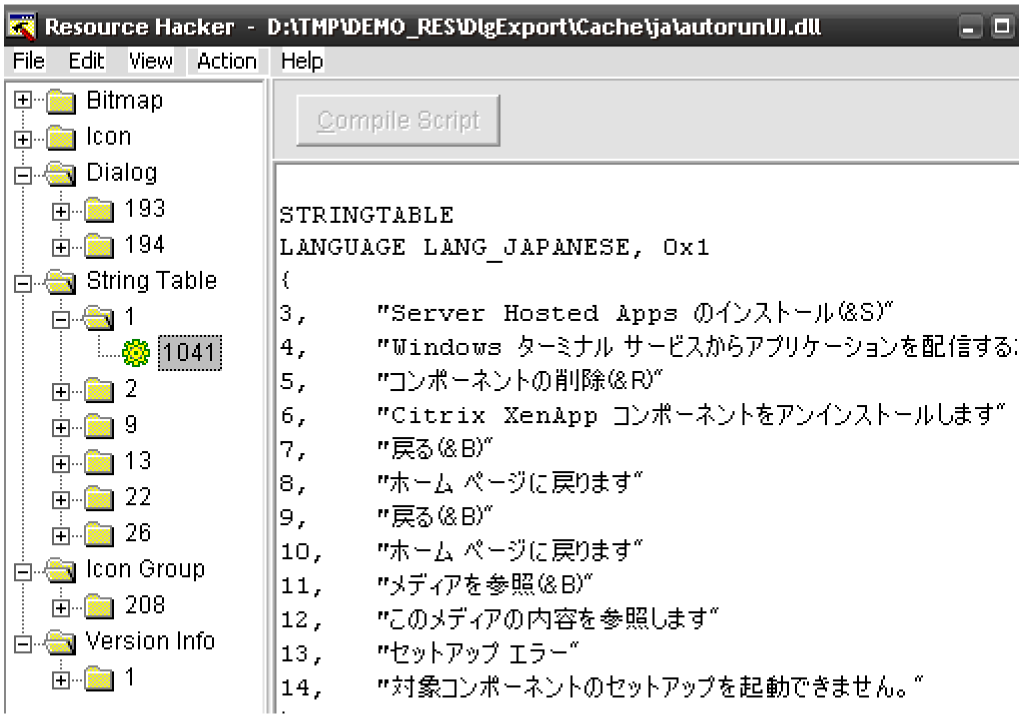
Commonly, text-type data subject to localization are stored into their corresponding language resource files. In some cases, dialogs and menus with localized strings are also stored in the compiled resource files. Using the available APIs, it is possible to view and verify the stored UI elements on module level without having to install the whole product suite.

Windows Resource Files

Localized UI elements including images, icons, strings, menus, dialog boxes and so on are placed into Windows resource file. Windows define the following resource element types for Win32 PE files as shown in Table 1. The resource types of our interest will be *Dialogs*, *Menus*, *String*, *MessageTable*, *Accelerators* and *Version* as these are the common objects which differs from each language platform depending on the granularity of localizing each objects. Figure 1 shows an example of a Win32 based resource UI module (autorunUI.dll) which contains several localized UI entries.

Table 1. Individual Resource Files Defined by Windows

Resource Type	***Element***	***File Format***	***Comment/ Description***
RT_CURSOR	Cursor	.CUR	#include in .RC file
RT_BITMAP	Bitmap or toolbar	.BMP	#include in .RC file
RT_ICON	Icon	.ICO	#include in .RC file
RT_MENU	Menu or pop up menu	.RC	#include in .RC file
RT_DIALOG	Dialog	.DLG or .RC	#include .DLG file in .RC file
RT_STRING	String	.RC	
RT_FONTDIR	Font	.FNT	
RT_FONT	Font	.FNT	
RT_ACCELERATORS	Accelerator	.RC	
RT_RCDATA	User-defined resource	.RC	Can use for constants or application specific structures
RT_MESSAGETABLE	Messages	.MC	#include compiled message table in .RC file
RT_GROUP_CURSOR	Cursor	N/A	Generated internally by resource compiler to provide Windows with information about cursor's resolution and type
RT_GROUP_ICON	Icon	N/A	Generated internally by resource compiler to provide Windows with information about icon's resolution and type
RT_VERSION	Version information	.RC	
RT_DLGINCLUDE	Header file that contains menu and dialog box #define statements	.RC	Used by resource editing tools; Visual C++ uses its own mechanism tools;
RT_PLUGPLAY	Plug and play resource		
RT_VXD			
RT_ANICURSOR			
RT_ANIICON			
RT_HTML			
RT_MANIFEST			



```
STRINGTABLE
LANGUAGE LANG_JAPANESE, 0x1
{
3,      "Server Hosted Apps のインストール(&S)"
4,      "Windows ターミナル サービスからアプリケーションを配信する
5,      "コンポーネントの削除(&R)"
6,      "Citrix XenApp コンポーネントをアンインストールします"
7,      "戻る(&B)"
8,      "ホーム ページに戻ります"
9,      "戻る(&B)"
10,     "ホーム ページに戻ります"
11,     "メディアを参照(&B)"
12,     "このメディアの内容を参照します"
13,     "セットアップ エラー"
14,     "対象コンポーネントのセットアップを起動できません。"
```

Figure 1. Sample view of a Win32 resource UI using Resource Hacker

On the other hand, .Net resource files differ from the old Win32 format. It provides storage for almost all types of data from strings to multimedia streams; however support for UI templates like Dialog and Menu templates was removed. Supported types for reading the resource elements are shown in Table 2. A sample view of a raw resx file in Visual Studio's Resource Editor is shown in Figure 2.

Table 2. .Net resource element types

• ResourceTypeCode.Null	• ResourceTypeCode.Int64
• ResourceTypeCode.String	• ResourceTypeCode.UInt64
• ResourceTypeCode.Boolean	• ResourceTypeCode.Single
• ResourceTypeCode.Char	• ResourceTypeCode.Double
• ResourceTypeCode.Byte	• ResourceTypeCode.Decimal
• ResourceTypeCode.SByte	• ResourceTypeCode.DateTime
• ResourceTypeCode.Int16	• ResourceTypeCode.TimeSpan
• ResourceTypeCode.UInt16	• ResourceTypeCode.ByteArray
• ResourceTypeCode.Int32	• ResourceTypeCode.Stream
• ResourceTypeCode.UInt32	

ResourceTypeCode.String is our primary concern for UI text verification of .Net-based assemblies. Regardless, which component or class the string entry is included, the embedded resource name could be used in determining whether it is a control label in a WinForm or is part of a global message string. (Related topics are discussed on the later sections).

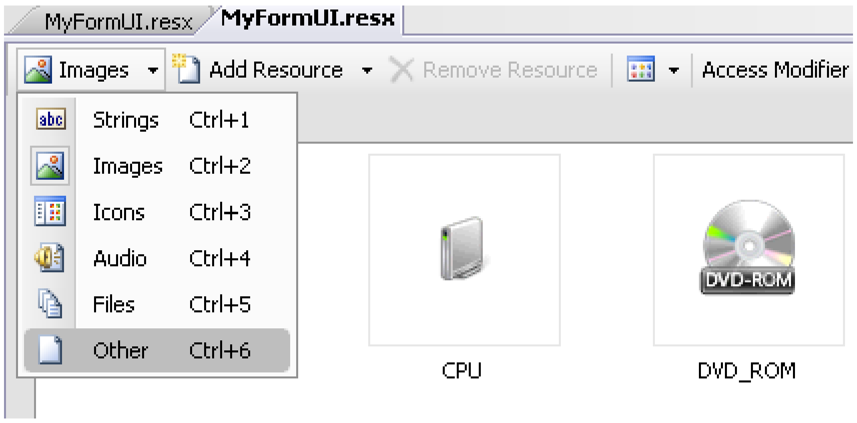


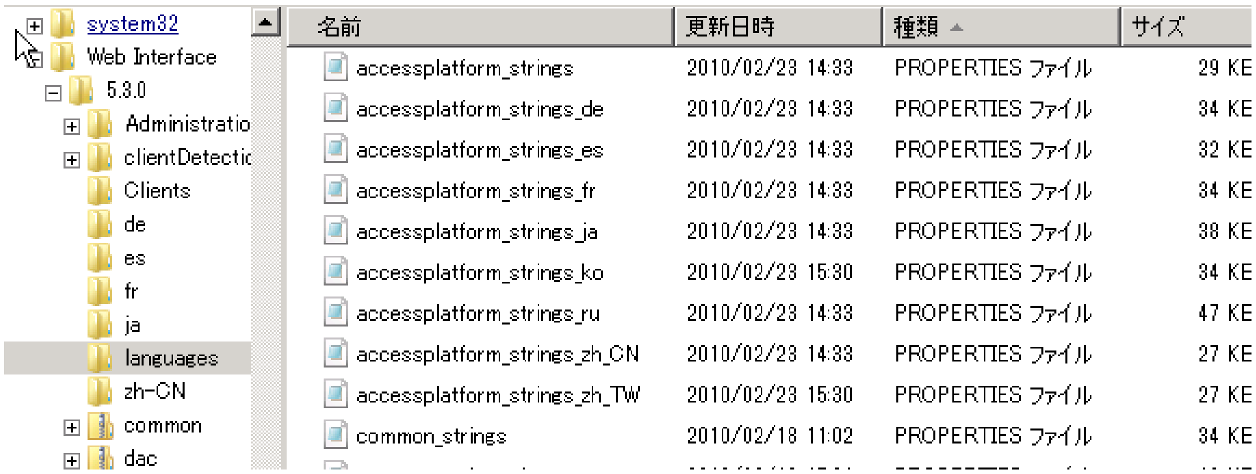
Figure 2. Sample view of a resx file using Visual Studio

Other Resource File formats

Java Properties file

Java properties file is a plain text file used to store configurable parameters in a Java application. The stored data comes into key-value pair strings. These files are also being used for storing localized strings which are called property resource bundles. Java has APIs for reading and writing resource bundles, and does have a support for the language fallback mechanism.

Property resource bundle is being used in some Citrix components especially the Web Interface's Java implementation for Tomcat based servers. There are several properties files included in the Web archive of PNA and XenApp which holds the localized web UI texts and string messages. The Citrix Management Console in Ohio is also another component implemented in Java. It has several resource bundle files included especially on the plug-in functionalities like Resource Manager.



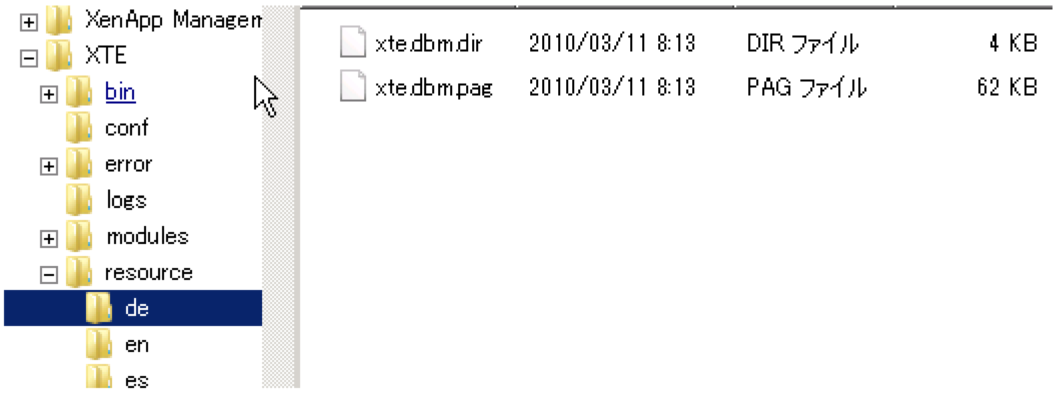
名前	更新日時	種類	サイズ
accessplatform_strings	2010/02/23 14:33	PROPERTIES ファイル	29 KE
accessplatform_strings_de	2010/02/23 14:33	PROPERTIES ファイル	34 KE
accessplatform_strings_es	2010/02/23 14:33	PROPERTIES ファイル	32 KE
accessplatform_strings_fr	2010/02/23 14:33	PROPERTIES ファイル	34 KE
accessplatform_strings_ja	2010/02/23 14:33	PROPERTIES ファイル	38 KE
accessplatform_strings_ko	2010/02/23 15:30	PROPERTIES ファイル	34 KE
accessplatform_strings_ru	2010/02/23 14:33	PROPERTIES ファイル	47 KE
accessplatform_strings_zh_CN	2010/02/23 14:33	PROPERTIES ファイル	27 KE
accessplatform_strings_zh_TW	2010/02/23 15:30	PROPERTIES ファイル	27 KE
common_strings	2010/02/18 11:02	PROPERTIES ファイル	34 KE

Figure 3. Web Interface properties file bundles

DBM hash file

Citrix XTE services make use of DBM (DataBase Manager) file format to store short language-dependent strings in key-value pairs. DBM file actually comes in 2 file-set implementations – the .dir and .pag files. The first file holds the table of contents and the latter holds the data. DBM implements hashing techniques for faster data retrieval using the key. Some implementations limit each key and value pair size to 1000bytes.

Currently there are several dbm successors like Ndbm, Sdbm, Gdbm, Jdbm and so on. DBM file format is commonly used in Perl-based applications which enable the program to store data for a quick lookup. Perl includes several DBM related functions for reading and manipulating of the data in the table.



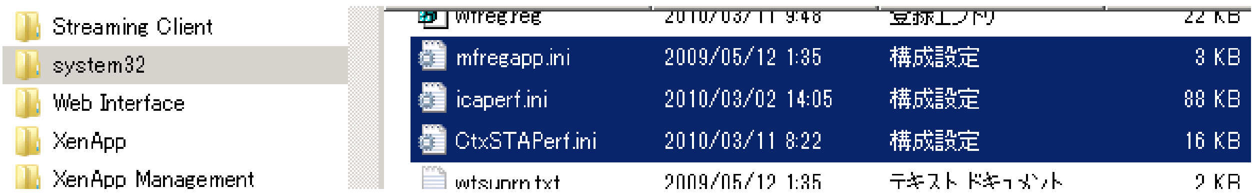
xte.dbm.dir	2010/03/11 8:13	DIR ファイル	4 KB
xte.dbm.pag	2010/03/11 8:13	PAG ファイル	62 KB

Figure 4. XTE service component's dbm resource file

INI file

INI file format is a standard format for configuration files. It is generally a plain text file with a basic structure. The data are stored in a key-value pair ('=' as the delimiter). Entries may be grouped in arbitrary sections.

Some performance counter strings in XenApp server modules utilities INI file structure format. Files like *icaperf.ini*, *ctxSTAperf.ini*, *mflicperf.ini*, *ctxcpuperf.ini* and *mfxpperf.ini* are some examples of the said files.



wfreg.reg	2010/03/11 9:48	登録エントリ	22 KB
mfregapp.ini	2009/05/12 1:35	構成設定	3 KB
icaperf.ini	2010/03/02 14:05	構成設定	88 KB
CtxSTAPerf.ini	2010/03/11 8:22	構成設定	16 KB
wtsuprn.txt	2009/05/12 1:35	テキスト ドキュメント	2 KB

Figure 5. INI resource files in XenApp system32 directory

XML file

Some Citrix products like Citrix Licensing component use XML-based resource files to store localized strings used in its Web UI, generally in its id-value form. The structure or schema may vary on each and every product components.

The XenApp Server Role Manager in Parra also uses XML formatted files with *.cxmi* extension name to store language dependent strings. Entries are treated as a *property* data with defined to a unique *id* key.

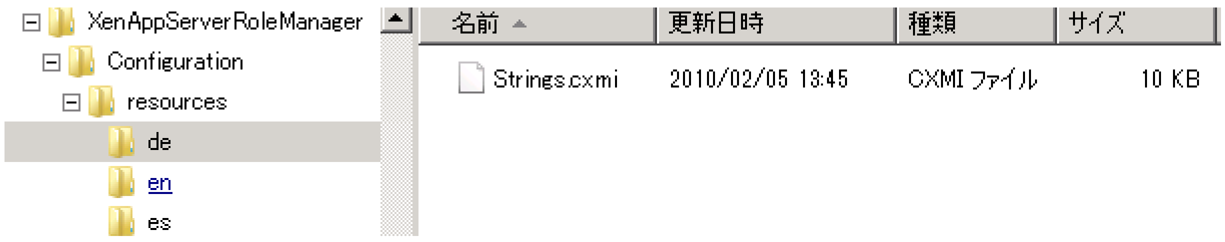


Figure 6. CMXI resource file in XenApp Server Role Manager

Citrix installer (autorun.exe) also utilizes XML files for storing the controls' property value and localized texts used on every wizard dialogs. Again, every product components make use of their own schema to define the contents structure of the file. There are also some components which make use of Microsoft *.resx* file format, which is actually an XML file structure used to store resources for .Net base assemblies. Shockwave Flash also uses an XML format as its localized string container.

ADM file

.adm file format is basically a plain text file with structured contents used for storing administrative policy template. It contains the conditions and registry settings for policy configurations and has a section where strings resources are being stored in *key = value* pair format. ICA client and HdxFlash components utilize adm files and are localized into other tier-1 language platforms.

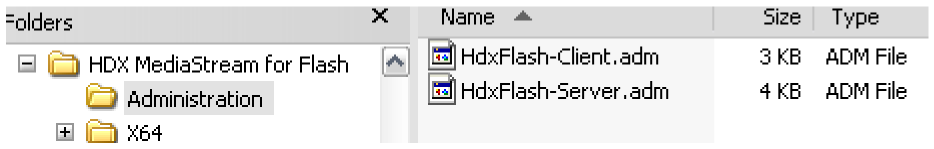


Figure 7. adm files in HDX Flash component

Text file

.Net supports a plain text file with key-value pair entry in *key = value* format as a resource file, and has APIs available to access and manipulate the data. This format however supports only textual string data (not any binary entities).

Text file format also serves as the format for localized long text data like license and/or EULA files where generally, the contents are treated as one entity.

MSI file and other formats

There are several other resource container formats like database file e.g. the msi, msp & mst file formats, *swf* for Adobe Flex, *.mo* for Python, and so on.

Resource UI Verification

Resource UI verification is a static testing methodology for checking UI related issues on the application's binaries especially resource UI dlls without having to install them. This process includes loading the binary module and checking on its resources (embedded or external) using public programming APIs. Basing on the fact that most of the localized components utilize resource container in externalizing the localized UI objects (message strings, dialog templates, menus and so on), verification on resource binary file level will help find issues at the earlier stage. Verification may be done against a single file and can be extended to UI objects' comparison of different binary versions or build, as well as comparison with the binary built on other language platform.

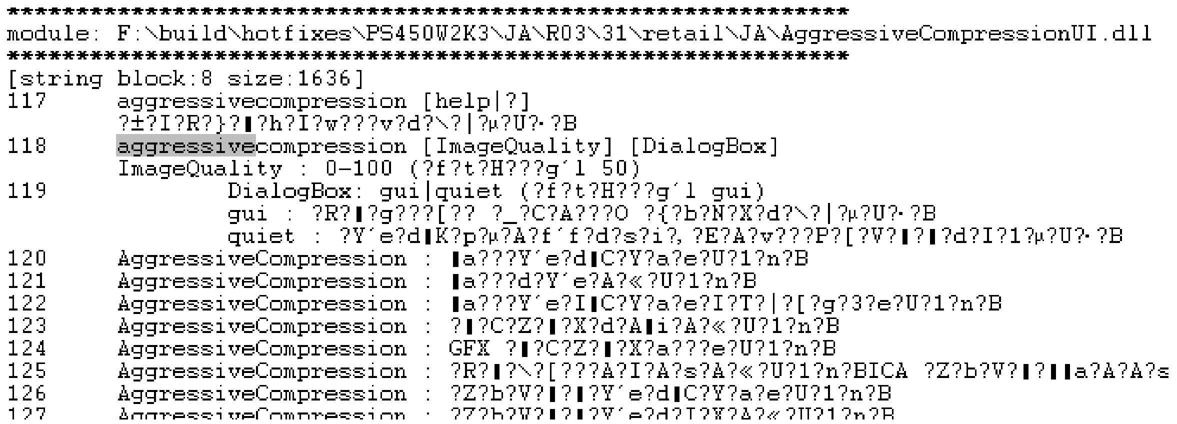
Single file verification

Single UI binary file verification is a process of loading a UI binary (or binary with UI resources) file and checking on the resource UI entries for the following related issues:

Garbled string issues

Garbled strings generally are caused by file encoding issue. When an original module (basically the English module) is copied for translation to another language platform, it some cases, the file's code-page should be set accordingly to the code-page which supports the target language. If file's code page is not properly chosen and set, there is a great possibility for a garbled string issue to occur.

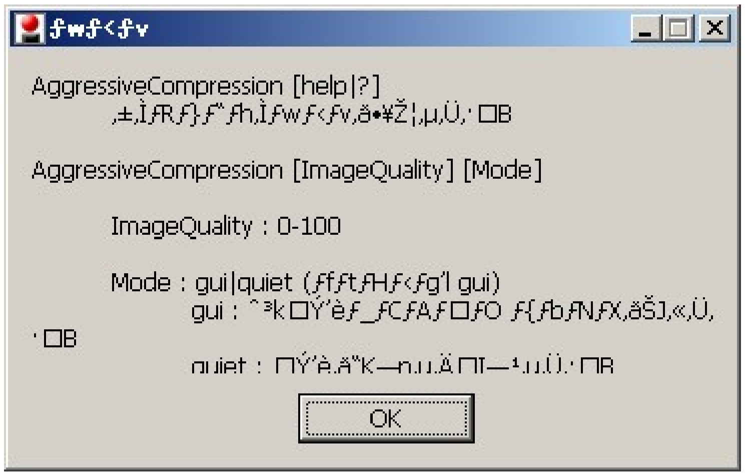
Figure 8 shows the dumped strings of the Japanese resource module aggressivecompressionUI.dll where the Japanese characters are not displayed properly1. It obviously implies that there was something wrong with the resource module which will help prompt the tester to cover the areas in UI testing. Figure 9 shows the actual UI dialog at runtime where the strings are also garbled for the same reason.



```
*************************************************************
module: F:\build\hotfixes\PS450W2K3\JA\R03\31\retail\JA\AggressiveCompressionUI.dll
*************************************************************
[string block:8 size:1636]
117     aggressivecompression [help|?]
        ?±?I?R?}?▮?h?I?w???v?d?\?|?µ?U?·?B
118     aggressivecompression [ImageQuality] [DialogBox]
        ImageQuality : 0-100 (?f?t?H???g´l 50)
119             DialogBox: gui|quiet (?f?t?H???g´l gui)
                gui : ?R?▮?g???[?? ?_?C?A???O ?{?b?N?X?d?\?|?µ?U?·?B
                quiet : ?Y´e?d▮K?p?µ?A?f´f?d?s?i?, ?E?A?v???P?[?V?▮?▮?d?I?1?µ?U?·?B
120     AggressiveCompression : ▮a???Y´e?d▮C?Y?a?e?U?1?n?B
121     AggressiveCompression : ▮a???d?Y´e?A?«?U?1?n?B
122     AggressiveCompression : ▮a???Y´e?I▮C?Y?a?e?I?T?|?[?g?3?e?U?1?n?B
123     AggressiveCompression : ?▮?C?Z?▮?X?d?A▮i?A?«?U?1?n?B
124     AggressiveCompression : GFX ?▮?C?Z?▮?X?a???e?U?1?n?B
125     AggressiveCompression : ?R?▮?\?[???A?I?A?s?A?«?U?1?n?BICA ?Z?b?V?▮?▮▮a?A?A?s
126     AggressiveCompression : ?Z?b?V?▮?▮?Y´e?d▮C?Y?a?e?U?1?n?B
127     AggressiveCompression : ?Z?b?V?▮?▮?Y´e?d?I?X?A?«?U?1?n?B
```

Figure 8. Garbled Japanese characters in the token dump of aggressivecompressionUI.dll

[1] Reported and described in CPR 193489



```
AggressiveCompression [help|?]
        ‚±‚ÌƒRƒ}ƒ“ƒh‚Ìƒwƒ‹ƒv‚ð•\Ž¦‚µ‚Ü‚·□B

AggressiveCompression [ImageQuality] [Mode]

        ImageQuality : 0-100

        Mode : gui|quiet (ƒfƒtƒHƒ‹ƒg'l gui)
                gui : ˆ³k□Ý'èƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX‚ðŠJ‚«‚Ü‚
·□B
                quiet : □Ý'è‚ð"K—p‚µ‚Ä□I—¹‚µ‚Ü‚·□B
```

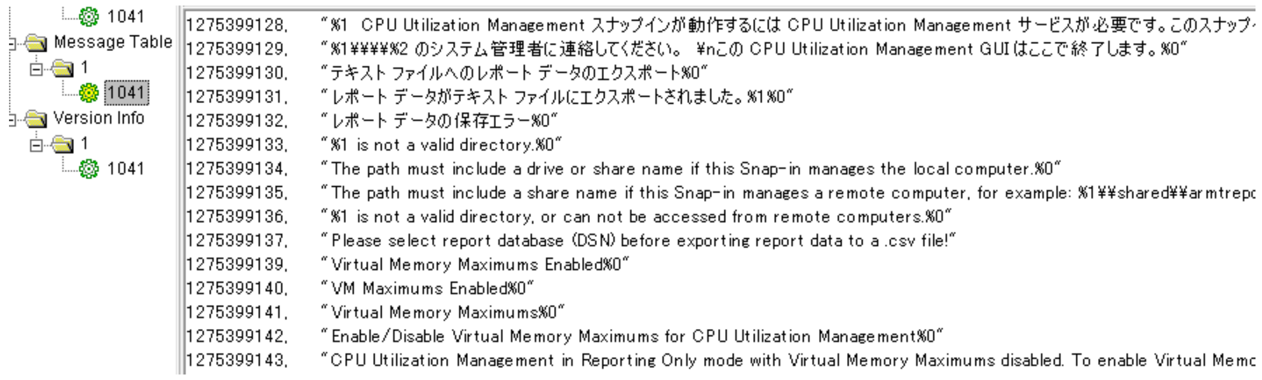
Figure 9. The actual UI dialog at runtime

Incomplete string issues

This is the case where the developer will sometimes add a resource entry to reserve a key but not the value. The value is to empty or marked with tags like *<TBD>*, *<ToBeTranslated>*, *<RequiresApproval>* and so on. The values might be the intended ones but it could be worth asking the linguist if they are really the correct ones.

Un-localized string issues

This is a case where newly added entries in the English platform are added or integrated as they are into the localized files. For newly localized products, the English resource file is copied and the initial values in English serve as the placeholder as well as the guide for the translating linguist. In some cases, there are some entries left un-translated or maybe new entries added later into the English resource file got integrated into the module and were missed to be translated. Figure 10 shows an example of some message strings which were left un-localized in Japanese resource module of ctxcpumsg.dll in Ohio server.



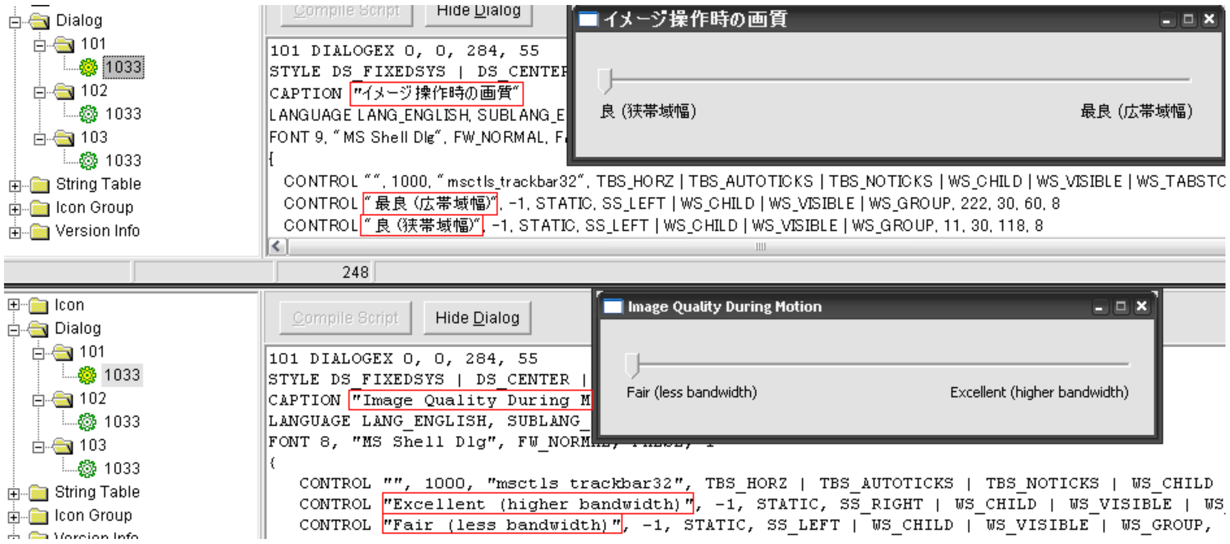
```
1275399128,     "%1  CPU Utilization Management スナップインが動作するには CPU Utilization Management サービスが必要です。このスナップ
1275399129,     "%1¥¥¥¥%2 のシステム管理者に連絡してください。 ¥nこの CPU Utilization Management GUI はここで終了します。%0"
1275399130,     "テキスト ファイルへのレポート データのエクスポート%0"
1275399131,     "レポート データがテキスト ファイルにエクスポートされました。%1%0"
1275399132,     "レポート データの保存エラー%0"
1275399133,     "%1 is not a valid directory.%0"
1275399134,     "The path must include a drive or share name if this Snap-in manages the local computer.%0"
1275399135,     "The path must include a share name if this Snap-in manages a remote computer, for example: %1¥¥shared¥¥armtrepo
1275399136,     "%1 is not a valid directory, or can not be accessed from remote computers.%0"
1275399137,     "Please select report database (DSN) before exporting report data to a .csv file!"
1275399139,     "Virtual Memory Maximums Enabled%0"
1275399140,     "VM Maximums Enabled%0"
1275399141,     "Virtual Memory Maximums%0"
1275399142,     "Enable/Disable Virtual Memory Maximums for CPU Utilization Management%0"
1275399143,     "CPU Utilization Management in Reporting Only mode with Virtual Memory Maximums disabled. To enable Virtual Memo
```

Figure 10. Un-localized messages in ctxcpumsg.dll

Hard-coded strings

In some UI resource binaries, Dialog and Menu templates are being used. Control labels and text properties are sometimes not externalized to String resource container. Rather, these text properties are placed on the localized Dialog or Menu template entries. They are localized but are hard-coded in the template level.

Figure 11 shows an example of dialog template resource used in aggressivecompressionUI.dll in Ohio product. The UI is localized in template level, but the control labels were hardcoded in the template.



```
101 DIALOGEX 0, 0, 284, 55
STYLE DS_FIXEDSYS | DS_CENTER
CAPTION "イメージ操作時の画質"
LANGUAGE LANG_ENGLISH, SUBLANG_E
FONT 9, "MS Shell Dlg", FW_NORMAL, F
{
  CONTROL "", 1000, "msctls_trackbar32", TBS_HORZ | TBS_AUTOTICKS | TBS_NOTICKS | WS_CHILD | WS_VISIBLE | WS_TABSTC
  CONTROL "最良 (広帯域幅)", -1, STATIC, SS_LEFT | WS_CHILD | WS_VISIBLE | WS_GROUP, 222, 30, 60, 8
  CONTROL "良 (狭帯域幅)", -1, STATIC, SS_LEFT | WS_CHILD | WS_VISIBLE | WS_GROUP, 11, 30, 118, 8
```

```
101 DIALOGEX 0, 0, 284, 55
STYLE DS_FIXEDSYS | DS_CENTER |
CAPTION "Image Quality During M
LANGUAGE LANG_ENGLISH, SUBLANG_
FONT 8, "MS Shell Dlg", FW_NORM
{
   CONTROL "", 1000, "msctls trackbar32", TBS_HORZ | TBS_AUTOTICKS | TBS_NOTICKS | WS_CHILD
   CONTROL "Excellent (higher bandwidth)", -1, STATIC, SS_RIGHT | WS_CHILD | WS_VISIBLE | WS
   CONTROL "Fair (less bandwidth)", -1, STATIC, SS_LEFT | WS_CHILD | WS_VISIBLE | WS_GROUP,
```

Figure 11. Dialog template with hard-coded control labels

Duplicate entries

There are some cases when newly added resource entry is assigned with an existing key; thus resulting to a duplicate-key issue. The value returned for querying the said key during runtime will not be guaranteed to be the correct data. In some cases, existing key-value pair also gets duplicated due to developer miss or due to code integration error. Figure 12 is an actual example of the issue[2] found during Ohio HRP02 testing where a new IMA message was incorrectly added into the module being assigned to an existing resource id.

-**2147217506** 免責注意! このツールにより、すべての公開アプリケーションのアプリケーション名が変更されます。Program Neighborhood のカスタム コネクションや静的な ICA ファイルを使用している場合は、手作業で新しいアプリケーション名を設定する必要があります。

続行しますか [y/n] ?

-**2147217506** サーバー %1 が見つかりません。データ ストアが無効です。dscheck /full servers コマンドを使用して修正してください。

Figure 12. Duplicate resource key issue in imamsgsui.dll (Ohio HRP02 Build 05)

[2] Reported in CPR 178238 under Ohio HRP02

Possible Implications:

Having duplicate keys may not affect the application's functionality but may cause a runtime error/exception in some codes like in resource managers using hashed collections where unique keys are required. In .Net, API *Dictionary.Add* will throw an *ArgumentException* if the added entry key is not unique. In other case when only one value of the two or many duplicated entries is loaded, the value returned by the resource manager might not be the intended value. Consider the following example where the application was suppose to display the second message instead of the first one. Since the first entry found for the resource key 00001 is returned to the app, the un-intended message string gets displayed.

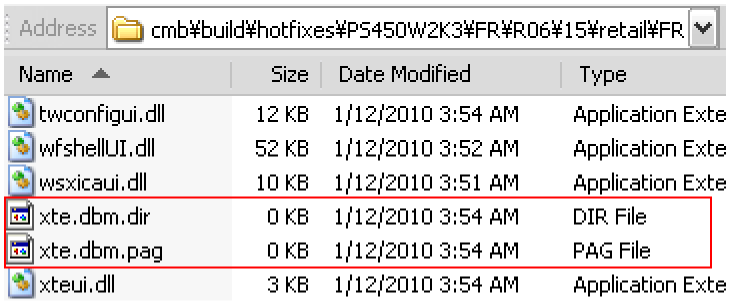
00001	Missiles launched!
00001	Missiles were turned off.

Build related issues

In some rare cases, the build script may have a wrong path to the source file which results to an empty file or a corrupted one after the build process. Some Citrix components like AIE, Streaming Profiler and Streaming client do have shared binaries. Pulling in those shared binaries from each components code branch needs careful attention considering the version and fix level of each code. In Ohio HRP builds, there are several cases where an un-intended version of the modules (like rademsgui.dll) with newly added strings and/or reverted entries are being added into the build[3]. The build script was pulling it the some binaries from the TC branch which are still yet to be worked on and in some cases, the string resources are all or partly still in English. Figure 13 shows an example of the strings which got reverted to English texts on the later build of Ohio HRP05 RU. Figure 14 shows an example of an empty dbm resource file which occurred on build 15 of the French Ohio HRP06[4].

906	Certificate Trust List (CTL) is not installed on the local machine.
1020	Unable to save the profile. The target folder could not be copied.
1110	Unable to create a target because the target already exists.
1153	Unable to save the target. The creation of a short file name index failed.
2004	Required routing information was not provided.

Figure 13. Strings of Russian RadeMsgUI.dll in HRP05 got reverted in later build


Address cmb¥build¥hotfixes¥PS450W2K3¥FR¥R06¥15¥retail¥FR

Name	Size	Date Modified	Type
twconfigui.dll	12 KB	1/12/2010 3:54 AM	Application Exte
wfshellUI.dll	52 KB	1/12/2010 3:52 AM	Application Exte
wsxicaui.dll	10 KB	1/12/2010 3:51 AM	Application Exte
xte.dbm.dir	0 KB	1/12/2010 3:54 AM	DIR File
xte.dbm.pag	0 KB	1/12/2010 3:54 AM	PAG File
xteui.dll	3 KB	1/12/2010 3:54 AM	Application Exte

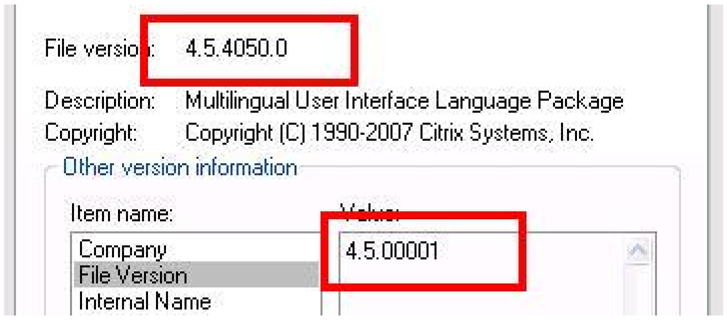
Figure 14. Empty resource file issue on Ohio HRP06 FR build 15

[3] CPRs 225525, 210659 and 212792 are related to rademsgui.dll resource strings reversion.
[4] Described in CPR 227715

Version Inconsistencies

Generally, all binary modules should have their version info but there are some Citrix binaries which do not have[5] any version info. Version info block has the fixed version info and the translated version info blocks. There are many cases where the version number specified in the fixed version info block is inconsistent with the value specified in the translated block. Figure 15 shows an example of file version number inconsistency in the Fixed version info block and the value in the Translated Block.

There are unlimited implications when the values differ on each block depending on how the values are being accessed and used. In some tools such as installer builders, the version info is critical when creating the transform files[6]. Generally, most tools use the value in the fixed block however; some tools especially those implemented in .Net might use the value in the translated block. .Net's API *FileVersionInfo.FileVersion* returns the version info from the translated block and not from the fixed block.



File version: 4.5.4050.0

Description: Multilingual User Interface Language Package

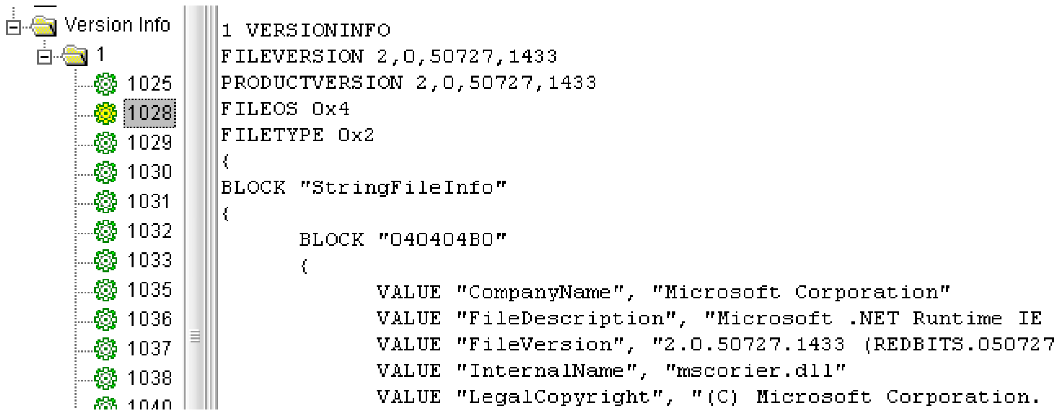
Copyright: Copyright (C) 1990-2007 Citrix Systems, Inc.

Other version information

Item name:	Value:
Company	4.5.00001
File Version	
Internal Name	

Figure 15. Inconsistency of file version between the *fixed* and *translated* block

Another rare case is when the version info is localized into several languages especially for the case of an MUI binary. One good example is the MUI binary mscorier.dll of MS Internet Explorer which has its version info localized to 24 languages (Figure 15). For this case, file attributes like file and product version number should be carefully checked for their consistency across the 24 localized languages.

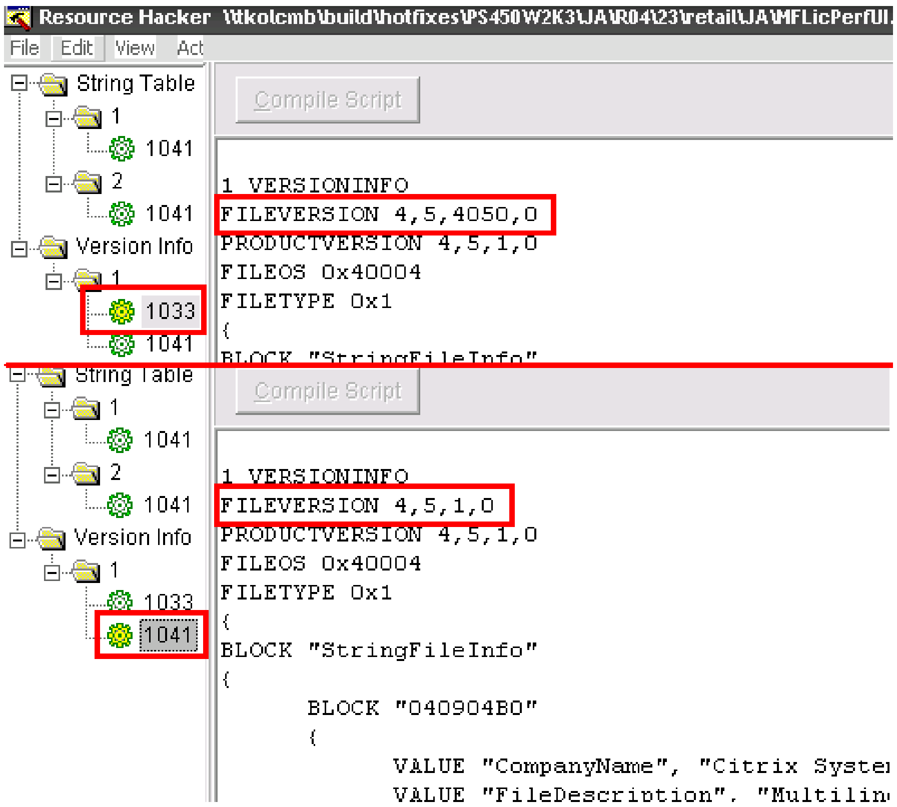


```
1 VERSIONINFO
FILEVERSION 2,0,50727,1433
PRODUCTVERSION 2,0,50727,1433
FILEOS 0x4
FILETYPE 0x2
{
BLOCK "StringFileInfo"
{
        BLOCK "040404B0"
        {
                VALUE "CompanyName", "Microsoft Corporation"
                VALUE "FileDescription", "Microsoft .NET Runtime IE
                VALUE "FileVersion", "2.0.50727.1433 (REDBITS.050727
                VALUE "InternalName", "mscorier.dll"
                VALUE "LegalCopyright", "(C) Microsoft Corporation.
```

Figure 16. IE mscorier.dll version info localized in 24 languages

[5] Currently, there's no guideline with regards to binary versioning.
[6] CPR 230096 describes related issue

In Citrix products, multi-language versioning is not put into practice. Some MUI resource modules have their strings and message table resources localized to the tier-1 languages but not the Version Info. In a very rare case however, build issue sometimes cause an insertion of a localized entry for its version info, resulting to having an inconsistent file versioning. Figure 17 shows an invalid insertion of another entry into the version info resource which caused an inconsistency in the file version attribute between the language platforms[7]. File version number if 4.5.4050 in EN while 4.5.1 in JA.



```
1 VERSIONINFO
FILEVERSION 4,5,4050,0
PRODUCTVERSION 4,5,1,0
FILEOS 0x40004
FILETYPE 0x1
{
BLOCK "StringFileInfo"
```

```
1 VERSIONINFO
FILEVERSION 4,5,1,0
PRODUCTVERSION 4,5,1,0
FILEOS 0x40004
FILETYPE 0x1
{
BLOCK "StringFileInfo"
{
        BLOCK "040904B0"
        {
                VALUE "CompanyName", "Citrix Syste
                VALUE "FileDescription", "Multilin
```

Figure 17. Inconsistent file versioning in MFLicPerfUI.dll

[7] Reported and described in CPR 230096

Cross-Release Comparison

Cross-release or cross-build comparison involves comparing the multiple versions of the same UI binary. It could be a comparison of the RTM version against the new version of the file included in the build of the new product upgrade; or could also be between the files in different builds as shown in Figure 18.

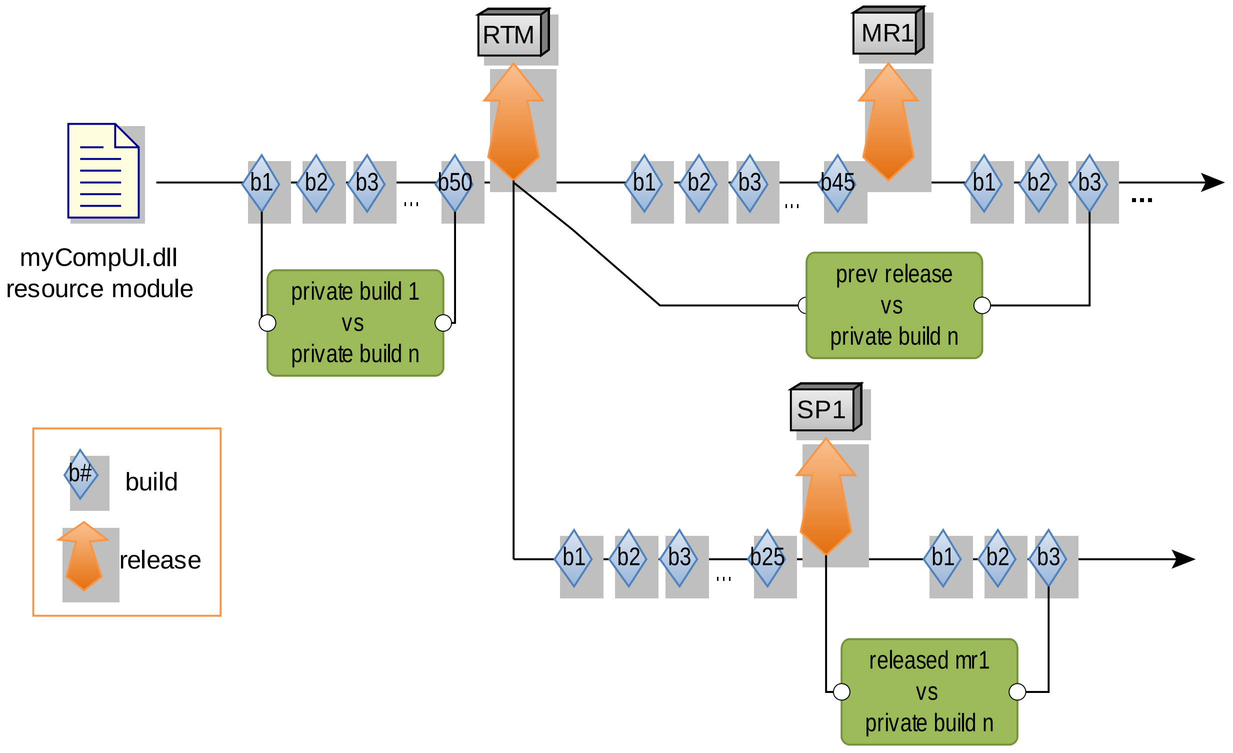
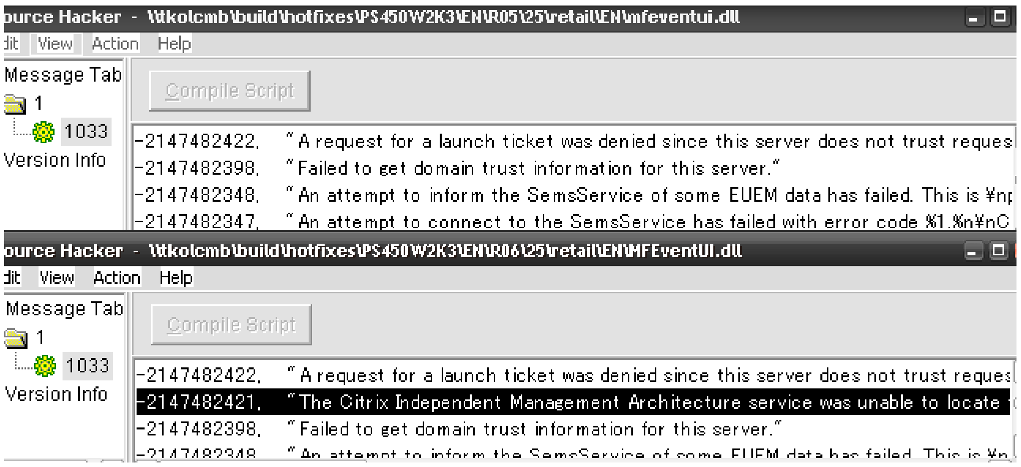

RTM
MR1
SP1
myCompUI.dll resource module
b1
b2
b3
...
b50
b45
b25
private build 1 vs private build n
prev release vs private build n
released mr1 vs private build n
b#
build
release

Figure 18. Cross-Release resource file comparison

By comparing and differentiating the between two versions of the same UI binaries, regressions and other unintended changes to the UI entries can be easily detected. The following items and possible issues can be verified in cross-release file comparison.

Newly added entries

When UI entries in the compared file version are not found in the base or reference file, these entries can be considered as *newly added entries* into the modified file. New UI objects like event messages or UI labels are sometimes included with the new functionalities on service packages or feature releases. In some cases, customer fixes may also require new UI entries depending on the issue being resolved. Figure 19 shows a newly added event message string into the mfeventui.dll module which was part of the fix to the customer issue reported on CPR209308. The corresponding strings in localized modules can also be verified in the same manner.


ource Hacker - \\tkolcmb\build\hotfixes\PS450W2K3\EN\R05\25\retail\EN\mfeventui.dll
dit View Action Help
Message Tab
1
1033
Version Info
Compile Script
ource Hacker - \\tkolcmb\build\hotfixes\PS450W2K3\EN\R06\25\retail\EN\MFEventUI.dll
dit View Action Help
Message Tab
1
1033
Version Info
Compile Script

```
-2147482422,  "A request for a launch ticket was denied since this server does not trust reques
-2147482398,  "Failed to get domain trust information for this server."
-2147482348,  "An attempt to inform the SemsService of some EUEM data has failed. This is ¥n
-2147482347,  "An attempt to connect to the SemsService has failed with error code %1.%n¥nC
```

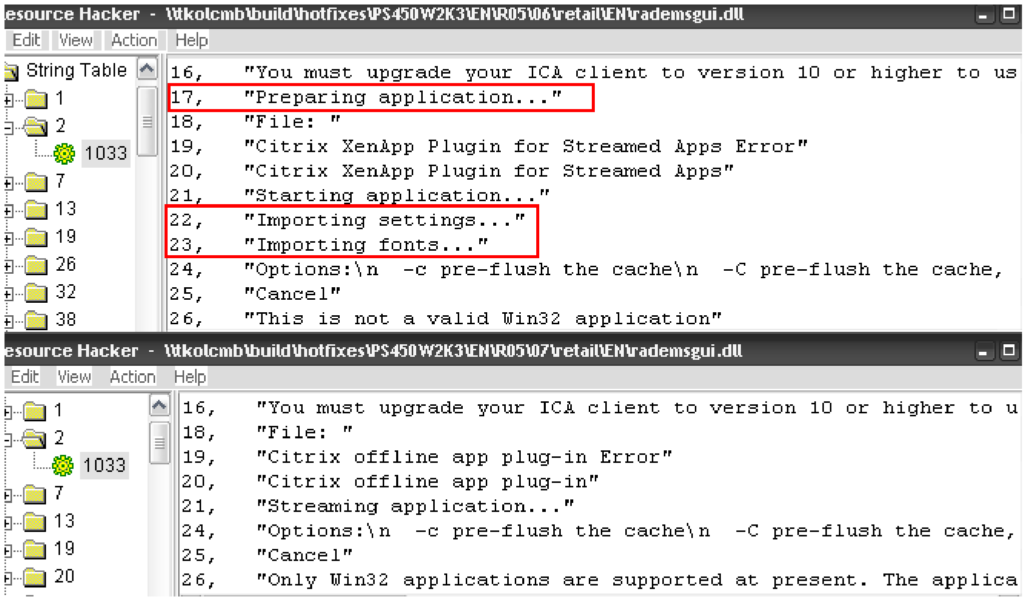
```
-2147482422,  "A request for a launch ticket was denied since this server does not trust reques
-2147482421,  "The Citrix Independent Management Architecture service was unable to locate
-2147482398,  "Failed to get domain trust information for this server."
-2147482348,  "An attempt to inform the SemsService of some EUEM data has failed. This is ¥n
```

Figure 19. Newly added message table string entry in resource UI dll

Deleted or missing entries

Differentiating UI entries between two versions will also give us the missing or removed entries in the newer version. Basically, if the product does support downward compatibility, resource UI entries are not removed from the newer file version. However, in some cases like code refactoring/cleanup, unused entries may have been removed intentionally by the developer. Missing entries could also be caused by code integration or build problem where a revision older than the reference file version was used. Figure 20 shows an example where some entries in the String table of rademsgui.dll were removed on the succeeding build version during Ohio HRP05 testing.

Missing resource entry may cause a functional regression on the affected component. It might trigger a runtime exception or an application crash in worst case scenario depending on the application's code implementation.


esource Hacker - \\tkolcmb\build\hotfixes\PS450W2K3\EN\R05\06\retail\EN\rademsgui.dll
Edit View Action Help
String Table
1
2
1033
7
13
19
26
32
38
esource Hacker - \\tkolcmb\build\hotfixes\PS450W2K3\EN\R05\07\retail\EN\rademsgui.dll
Edit View Action Help
1
2
1033
7
13
19
20

```
16,    "You must upgrade your ICA client to version 10 or higher to us
17,    "Preparing application..."
18,    "File: "
19,    "Citrix XenApp Plugin for Streamed Apps Error"
20,    "Citrix XenApp Plugin for Streamed Apps"
21,    "Starting application..."
22,    "Importing settings..."
23,    "Importing fonts..."
24,    "Options:\n  -c pre-flush the cache\n  -C pre-flush the cache,
25,    "Cancel"
26,    "This is not a valid Win32 application"
```

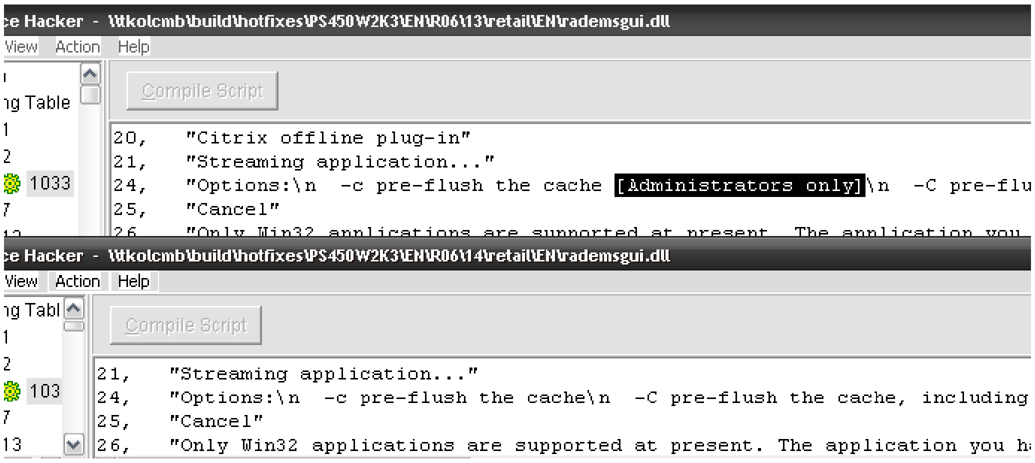
```
16,    "You must upgrade your ICA client to version 10 or higher to u
18,    "File: "
19,    "Citrix offline app plug-in Error"
20,    "Citrix offline app plug-in"
21,    "Streaming application..."
24,    "Options:\n  -c pre-flush the cache\n  -C pre-flush the cache,
25,    "Cancel"
26,    "Only Win32 applications are supported at present. The applica
```

Figure 20. Missing string entries in the later builds of rademsgui.dll in Ohio HRP05

Reverted or modified entries

Changes or any reversion to existing UI entries can also be observed by comparing two versions of the UI resource files. Modifications may be intentional like when the original text context was incorrect or additional descriptions were added. But in most cases, integration and build problems are the main cause of the UI entries' reversion.

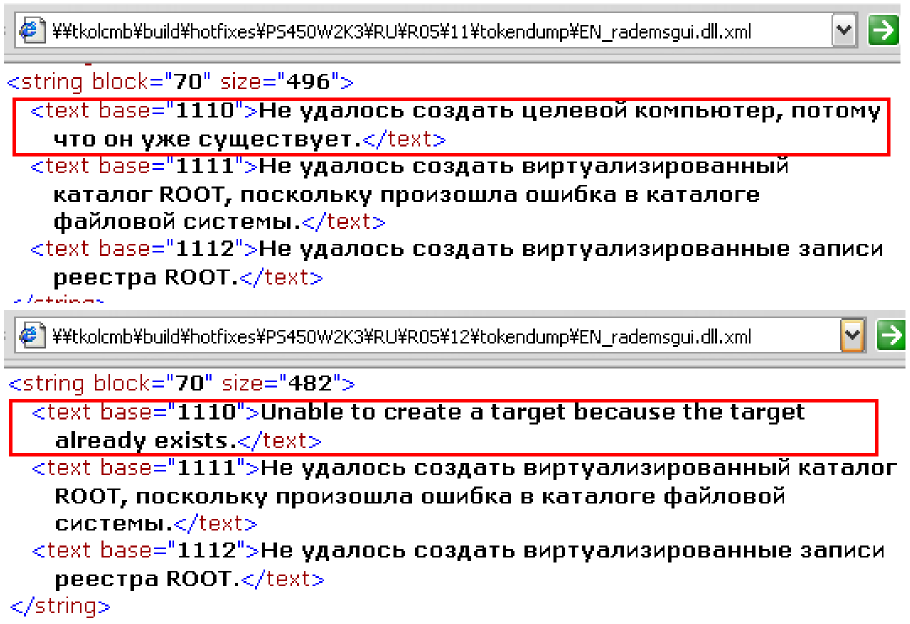
Figure 21 shows an example of a string resource entry which has some words "[Administrators only]" removed on the next build version. Figure 22 on the other hand is total reversion on the value from Russian strings to English strings.


ce Hacker - \\tkolcmb\build\hotfixes\PS450W2K3\EN\R06\13\retail\EN\rademsgui.dll
View Action Help
Compile Script
ng Table
1033
ce Hacker - \\tkolcmb\build\hotfixes\PS450W2K3\EN\R06\14\retail\EN\rademsgui.dll
View Action Help
Compile Script
ng Tabl
103

```
20,    "Citrix offline plug-in"
21,    "Streaming application..."
24,    "Options:\n  -c pre-flush the cache [Administrators only]\n  -C pre-flu
25,    "Cancel"
```

```
21,    "Streaming application..."
24,    "Options:\n  -c pre-flush the cache\n  -C pre-flush the cache, including
25,    "Cancel"
26,    "Only Win32 applications are supported at present. The application you h
```

Figure 21. Some words removed from the string resource entry


¥¥tkolcmb¥build¥hotfixes¥PS450W2K3¥RU¥R05¥11¥tokendump¥EN_rademsgui.dll.xml
¥¥tkolcmb¥build¥hotfixes¥PS450W2K3¥RU¥R05¥12¥tokendump¥EN_rademsgui.dll.xml

```
<string block="70" size="496">
  <text base="1110">Не удалось создать целевой компьютер, потому
    что он уже существует.</text>
  <text base="1111">Не удалось создать виртуализированный
    каталог ROOT, поскольку произошла ошибка в каталоге
    файловой системы.</text>
  <text base="1112">Не удалось создать виртуализированные записи
    реестра ROOT.</text>
```

```
<string block="70" size="482">
  <text base="1110">Unable to create a target because the target
    already exists.</text>
  <text base="1111">Не удалось создать виртуализированный каталог
    ROOT, поскольку произошла ошибка в каталоге файловой
    системы.</text>
  <text base="1112">Не удалось создать виртуализированные записи
    реестра ROOT.</text>
</string>
```

Figure 22. String entry in Russian reverted to English in the succeeding build.

Key alignment issue

Key alignment issue occurs when an added resource entry is inserted in between the existing entries. The succeeding existing entries are pushed down the table getting assigned with different or incremented keys. This is a pretty critical issue considering that the context of the actual resource value might be different to what it used to be. This issue is primarily caused by a miss on the part of the responsible developer; and in some rare case could be caused by build configuration issue.

Versioning issue

If the resource UI binaries being compared contain the versioning info, it is also possible to trace any versioning inconsistencies. Newer version or build are expected to have a higher version number than the original one. Changes to other file version properties like file description and product version can also be traced thru this comparison.

Cross-Language Comparison

Cross-language file comparison involves comparing the localized versions of the same UI binary. It could be a comparison of EN resource binary to other localized language versions like JA. Generally, the cross-language file comparison is conducted on the same version level of the UI resource modules. However, the resource entries comparison may also be done on multiple file of different release or build versions depending on the regression tasks being performed. By differentiating the localized versions with the base language implementation (generally English), un-localized strings as well as other possible issues can be detected.

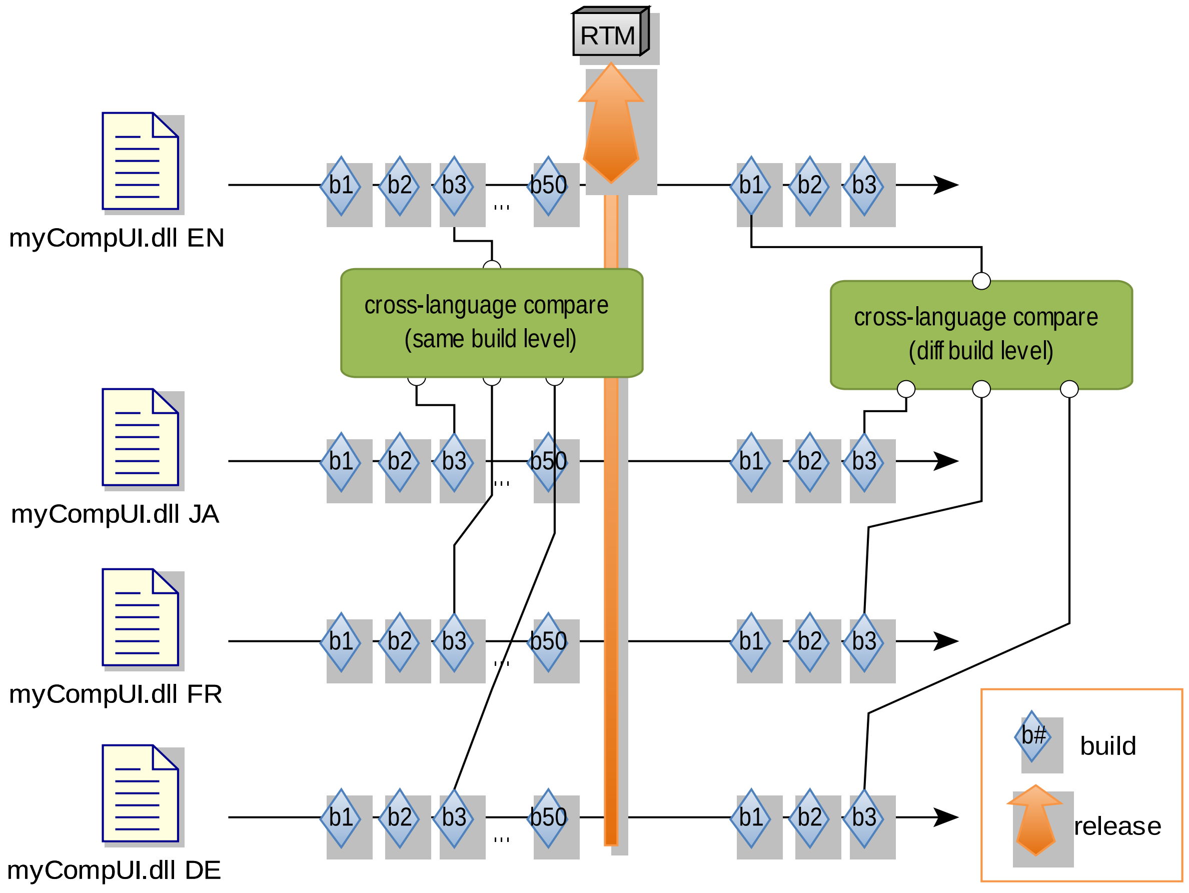
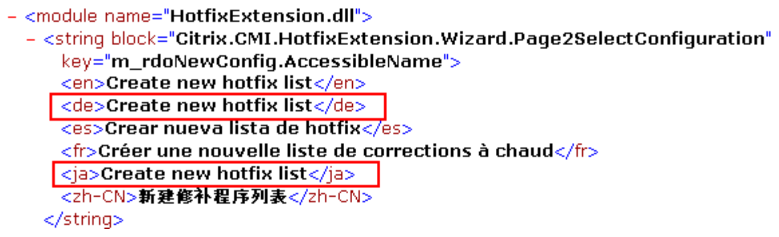


Figure 23. Cross-Language resource file comparison

Un-localized entries

When a UI resource entry is similar with that of the base language value, it is possible that the said entry was un-localized to the target language platform. The granularity of localizing entries depends on the context where the resources are being used. Strings like descriptions and tips are primarily being translated while some string resources like performance counter units and object instance names are not intended to be localized. Which UI objects subject to localization are determined solely the by the localization developers and linguists[8]. Figure 24 shows an example of un-translated strings in German and Japanese resource modules of HotfixExtension.dll in Parra.



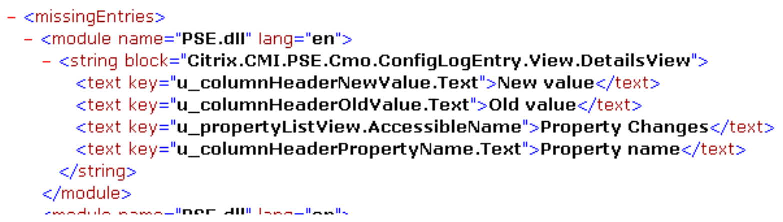
```
- <module name="HotfixExtension.dll">
  - <string block="Citrix.CMI.HotfixExtension.Wizard.Page2SelectConfiguration"
      key="m_rdoNewConfig.AccessibleName">
      <en>Create new hotfix list</en>
      <de>Create new hotfix list</de>
      <es>Crear nueva lista de hotfix</es>
      <fr>Créer une nouvelle liste de corrections à chaud</fr>
      <ja>Create new hotfix list</ja>
      <zh-CN>新建修补程序列表</zh-CN>
    </string>
```

Figure 24. Un-localized strings in Parra's XA Mgmt Console

Missing entries

Entries not found on the localized binaries and vice-versa can also be detected by cross-language comparison. In some cases, not only the text entries are missing rather the whole resource blocks were not localized in the corresponding language platforms. Problem in code integration or referencing incorrect branch during build sometimes causes inconsistencies of entries between each language versions.

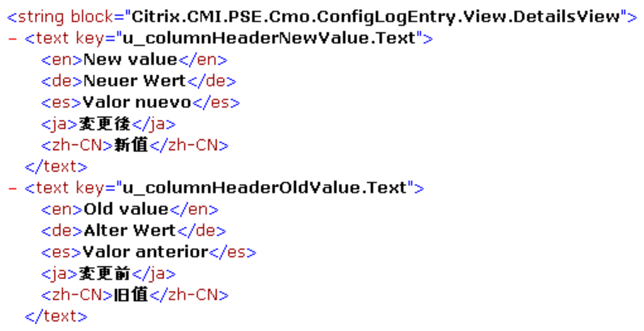
Figure 26 is a part of the output result of rescmp tool indicating that the listed EN entries are missing in one of the compared languages in PSE.dll of Parra XA Mgmt Console.



```
- <missingEntries>
  - <module name="PSE.dll" lang="en">
    - <string block="Citrix.CMI.PSE.Cmo.ConfigLogEntry.View.DetailsView">
        <text key="u_columnHeaderNewValue.Text">New value</text>
        <text key="u_columnHeaderOldValue.Text">Old value</text>
        <text key="u_propertyListView.AccessibleName">Property Changes</text>
        <text key="u_columnHeaderPropertyName.Text">Property name</text>
      </string>
    </module>
```

Figure 25. Missing entries in Parra's

Figure 26 shows the resource values in the merged XML output file where FR strings are missing.

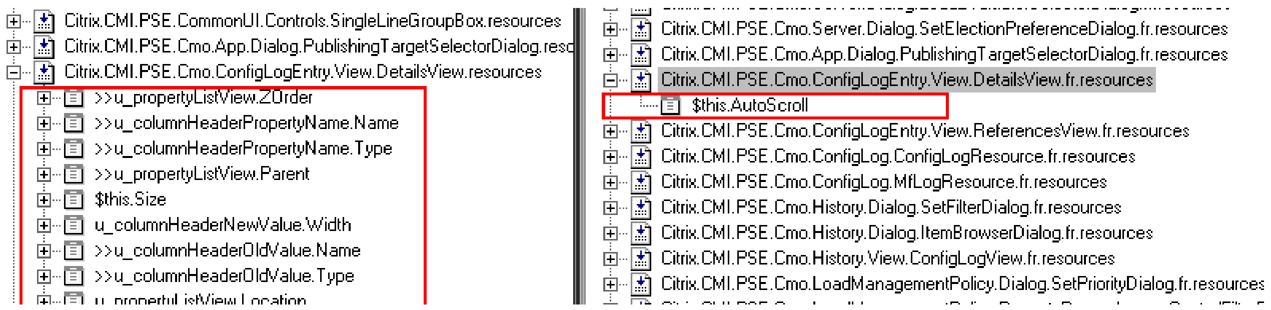


```
<string block="Citrix.CMI.PSE.Cmo.ConfigLogEntry.View.DetailsView">
- <text key="u_columnHeaderNewValue.Text">
    <en>New value</en>
    <de>Neuer Wert</de>
    <es>Valor nuevo</es>
    <ja>変更後</ja>
    <zh-CN>新值</zh-CN>
  </text>
- <text key="u_columnHeaderOldValue.Text">
    <en>Old value</en>
    <de>Alter Wert</de>
    <es>Valor anterior</es>
    <ja>変更前</ja>
    <zh-CN>旧值</zh-CN>
  </text>
```

Figure 26. Un-localized block in Parra's FR resource module

[8] Currently, there's no guideline which define the granularity of UI objects subject to localization.

Using an assembly analyzer tool (Asmex) and comparing on the embedded resources of both the English and French modules will help us confirm the issue as shown in Figure 27. The FR resources were embedded but the entries are missing i.e. they were not translated.



```
Citrix.CMI.PSE.CommonUI.Controls.SingleLineGroupBox.resources
Citrix.CMI.PSE.Cmo.App.Dialog.PublishingTargetSelectorDialog.reso
Citrix.CMI.PSE.Cmo.ConfigLogEntry.View.DetailsView.resources
    >>u_propertyListView.ZOrder
    >>u_columnHeaderPropertyName.Name
    >>u_columnHeaderPropertyName.Type
    >>u_propertyListView.Parent
    $this.Size
    u_columnHeaderNewValue.Width
    >>u_columnHeaderOldValue.Name
    >>u_columnHeaderOldValue.Type
```

```
Citrix.CMI.PSE.Cmo.Server.Dialog.SetElectionPreferenceDialog.fr.resources
Citrix.CMI.PSE.Cmo.App.Dialog.PublishingTargetSelectorDialog.fr.resources
Citrix.CMI.PSE.Cmo.ConfigLogEntry.View.DetailsView.fr.resources
    $this.AutoScroll
Citrix.CMI.PSE.Cmo.ConfigLogEntry.View.ReferencesView.fr.resources
Citrix.CMI.PSE.Cmo.ConfigLog.ConfigLogResource.fr.resources
Citrix.CMI.PSE.Cmo.ConfigLog.MfLogResource.fr.resources
Citrix.CMI.PSE.Cmo.History.Dialog.SetFilterDialog.fr.resources
Citrix.CMI.PSE.Cmo.History.Dialog.ItemBrowserDialog.fr.resources
Citrix.CMI.PSE.Cmo.History.View.ConfigLogView.fr.resources
Citrix.CMI.PSE.Cmo.LoadManagementPolicy.Dialog.SetPriorityDialog.fr.resources
```

Figure 27. Comparison of the FR resource with EN resource (using Asmex)

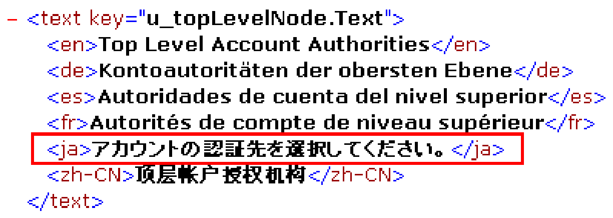
Translation issues

Translation issues are generally context-related issues i.e. the original English strings were translated on a different context scope resulting to a *context-issue*. Likewise, some English strings which need to be consistent across the localized languages somehow get being translated resulting to *over-translation.*

Context-issues

By comparing the localized strings with the base English texts, it is possible to find context related issues. Context issue arises when the meaning or context of the localized strings deviates from the intended semantic. For example, the English word "close" will have different meaning when used in different context. It could be translated to 閉じる (shut) 、近い (near) or 親しい (intimate). To be able to find context-related issues in the translated strings, certain knowledge or understanding of the localized language becomes necessary.

In some rare cases, the original English strings are sometimes found to be contextually or structurally incorrect. Certain knowledge on when and where the strings are used in the application is required. For example, full sentence/s may be suitable for strings used in descriptions, tooltips and messages, but for labels and titles, the translated string may have to follow certain localization rules. Figure 28 shows an example where a presumed label text "Top Level Account Authorities" in English is translated into a complete sentence in Japanese which now means "Please select the account authority.". The nuance differs a bit between the original and translated texts.

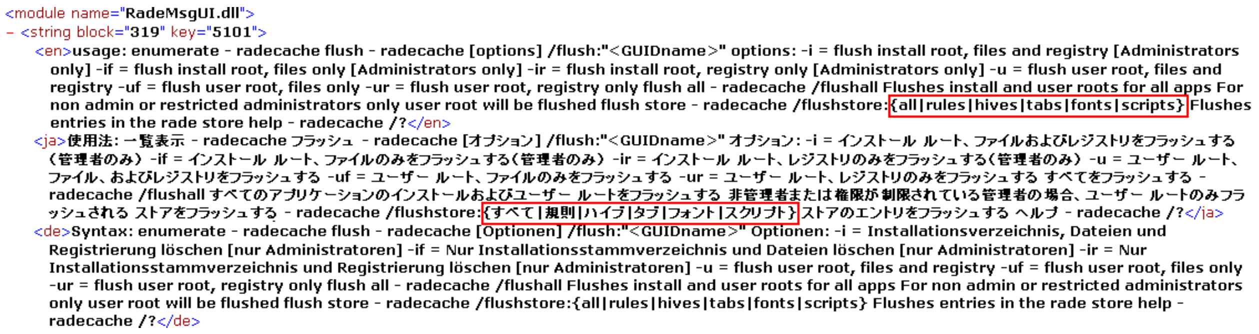


```
- <text key="u_topLevelNode.Text">
    <en>Top Level Account Authorities</en>
    <de>Kontoautoritäten der obersten Ebene</de>
    <es>Autoridades de cuenta del nivel superior</es>
    <fr>Autorités de compte de niveau supérieur</fr>
    <ja>アカウントの認証先を選択してください。</ja>
    <zh-CN>顶层帐户授权机构</zh-CN>
  </text>
```

Figure 28. Example of different nuance in translated text

Over-translation

In some cases, over-translation can also be found through this procedure. Over-translation is the term used to refer to translating texts which aren't deemed necessary or should be consistent all across language platforms. Some strings like performance counters, measurement units, command options and SQL statements may not be subject to translation; and translating those strings sometimes caused functionality regressions in some components which loads them. Date-time and numeric entries may not need to be translated but their format might be modified according to the target locale; but the context where they are being used should still be considered. Figure 29 shows an example of over-translation in RadeMsgUI.dll in Parra where the acceptable parameters to **/flushstore** switch was localized in Japanese language. Surely, the application will not accept those Japanese parameter strings and not work properly.



```
<module name="RadeMsgUI.dll">
- <string block="319" key="5101">
    <en>usage: enumerate - radecache flush - radecache [options] /flush:"<GUIDname>" options: -i = flush install root, files and registry [Administrators
      only] -if = flush install root, files only [Administrators only] -ir = flush install root, registry only [Administrators only] -u = flush user root, files and
      registry -uf = flush user root, files only -ur = flush user root, registry only flush all - radecache /flushall Flushes install and user roots for all apps For
      non admin or restricted administrators only user root will be flushed flush store - radecache /flushstore:{all|rules|hives|tabs|fonts|scripts} Flushes
      entries in the rade store help - radecache /?</en>
    <ja>使用法: 一覧表示 - radecache フラッシュ - radecache [オプション] /flush:"<GUIDname>" オプション: -i = インストール ルート、ファイルおよびレジストリをフラッシュする
      (管理者のみ) -if = インストール ルート、ファイルのみをフラッシュする(管理者のみ) -ir = インストール ルート、レジストリのみをフラッシュする(管理者のみ) -u = ユーザー ルート、
      ファイル、およびレジストリをフラッシュする -uf = ユーザー ルート、ファイルのみをフラッシュする -ur = ユーザー ルート、レジストリのみをフラッシュする すべてをフラッシュする -
      radecache /flushall すべてのアプリケーションのインストールおよびユーザー ルートをフラッシュする 非管理者または権限が制限されている管理者の場合、ユーザー ルートのみフラ
      ッシュされる ストアをフラッシュする - radecache /flushstore:{すべて|規則|ハイブ|タブ|フォント|スクリプト} ストアのエントリをフラッシュする ヘルプ - radecache /?</ja>
    <de>Syntax: enumerate - radecache flush - radecache [Optionen] /flush:"<GUIDname>" Optionen: -i = Installationsverzeichnis, Dateien und
      Registrierung löschen [nur Administratoren] -if = Nur Installationsstammverzeichnis und Dateien löschen [nur Administratoren] -ir = Nur
      Installationsstammverzeichnis und Registrierung löschen [nur Administratoren] -u = flush user root, files and registry -uf = flush user root, files only
      -ur = flush user root, registry only flush all - radecache /flushall Flushes install and user roots for all apps For non admin or restricted administrators
      only user root will be flushed flush store - radecache /flushstore:{all|rules|hives|tabs|fonts|scripts} Flushes entries in the rade store help -
      radecache /?</de>
```

Figure 29. Command-line switch parameters translated to Japanese strings

Format tags issues

Tag order

When a parametized string is localized, the order of the format tags or items should be carefully taken into consideration. Examples of format tag are **%s**, **%5d** in C/C++'s and **{0}**, **{1:E}** in .Net. In other localized language platforms, these format tags may have to be re-ordered according to the language grammar and sentence structure. The following Figure 30 shows an example of formatting tags which were re-ordered in Japanese translated text according to correct grammar use and semantics.

English:	{0} over {1} is the answer.
Japanese:	正解は{1}分の{0}です。

Figure 30. An example of re-ordered formatting tags

Missing placeholders

In addition to parameter order issue, in some cases, the parameter placeholders are found missing on the translated texts. Some placeholders may have been accidentally removed during the translation process or dropped intentionally by the linguists. Figure 31 shows an example where the second tag ({1}) was dropped in the translated texts for the reason that Japanese yen doesn't have the notion of cents.

English:	Total is ${0} and {1} cents.
Japanese:	合計は￥{0}です。

Figure 31. An example of missing or unused formatting tags

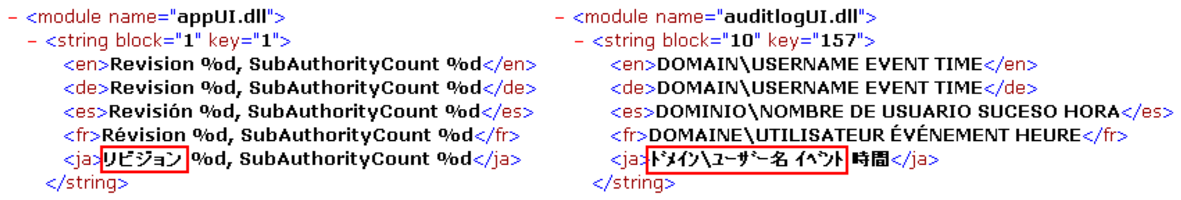
Relative glyph sizes

In some languages like Japanese and Hangul, some characters can be represented in half-width and full-width forms. Full-width (zenkaku 全角) and half-width (hankaku 半角) characters are two different things – where the former is represented in 1-byte[9] and the latter in 2-byte forms. The context being implied by the both characters is similar but they're treated differently in code level. For example, the numeric value of 3000 will be similar to ３０００only that its representation differs a bit in size. Latin characters which are considered half-width have their full-width representation e.g. the word "Test" can also be represented as "Ｔｅｓｔ" in full-width form.

Resource strings with Latin and numeric characters subject to translation in CJK languages, might take full-width forms when localized especially numeric characters and Latin words which aren't suitable to being translated like SQL statements. The following cases are some examples where the use of full-width forms may cause functionality issues.

- Full-width numeric characters in parameter placeholders e.g. "{２}" instead of "{2}".
- Full-width numeric characters in date/time e.g. "２０１０" instead of "2010"
- A column name in SQL statement is in full-width form e.g. "Ｎａｍｅ" instead of "Name".

Moreover, Japanese katakana characters can be expressed in full-width and half-width representations. Which glyph size to take depends solely on the translating linguists or localization developers. Figure 32 shows an example where full-width katakana characters are used in the Japanese translated text "リボジョン" for the English word "Revision" (left). On the hand, half-width katakana characters are used in translating the words "domain", "user" and "event" into Japanese as shown on the same figure (right). Using full-width form will require more spaces the half-width when used as text labels in controls. If their length and size are not properly set, it would cause a string truncation in the control during display.



```
- <module name="appUI.dll">
  - <string block="1" key="1">
      <en>Revision %d, SubAuthorityCount %d</en>
      <de>Revision %d, SubAuthorityCount %d</de>
      <es>Revisión %d, SubAuthorityCount %d</es>
      <fr>Révision %d, SubAuthorityCount %d</fr>
      <ja>リビジョン %d, SubAuthorityCount %d</ja>
    </string>
```

```
- <module name="auditlogUI.dll">
  - <string block="10" key="157">
      <en>DOMAIN\USERNAME EVENT TIME</en>
      <de>DOMAIN\USERNAME EVENT TIME</de>
      <es>DOMINIO\NOMBRE DE USUARIO SUCESO HORA</es>
      <fr>DOMAINE\UTILISATEUR ÉVÉNEMENT HEURE</fr>
      <ja>ﾄﾞﾒｲﾝ\ﾕｰｻﾞｰ名 ｲﾍﾞﾝﾄ 時間</ja>
    </string>
```

Figure 32. Full-width & half-with Japanese Katakana in translated texts

Versioning issues

Comparing the version info of the localized binaries, inconsistencies on the version number and other properties expected to be similar across language platforms can be detected. Generally, localized modules included for the same build release should have consistent fixed file version number and their corresponding version info properties. On the other hand, properties in the Translation block info which are subject to localization (e.g. File Description field) can be compared side-by-side and textually verified.

[9] Except in EUC-JP and ISO 10646 where half-width katakana characters are expressed in 2 bytes.

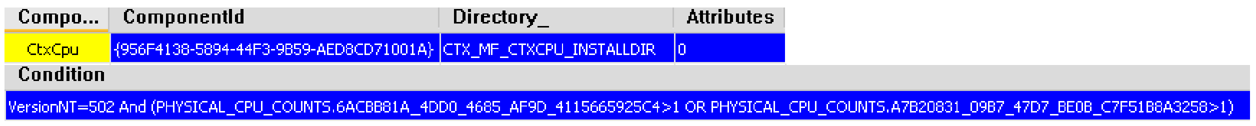
Leveraging resource UI verification

As shown in the previous sections, supported UI elements vary on each type of resource file formats and by verifying and comparing those resource elements, several issues may be found in the earlier stage of development. This section describes where static resource UI verification on file unit level serves useful and lists some possible cases where this method suits most.

Hardware Dependencies

Depending on the products being tested, there are some cases when a specific binary/ies will only be installed into the system which satisfies the hardware requirements. This hardware dependency could be CPU specific – CPU type, core count, supported feature like VT, or could a dependency to the underlying OS platforms – OS feature installed, patch or specific object version level. This kind of dependencies should be known at hand prior testing as it would be impossible to verify the UI elements of the related binaries at runtime if they won't be available in the target system.

Citrix's XenApp includes several CPU optimization binaries (generally prefixed with *ctxcpu** in their filenames) which *only get* installed if the target host has more than two CPU or cores. Figure 33 shows the condition set for installing CtxCpu entry in Components table of mps.msi. When the condition is satisfied, the component binaries gets installed into the system including localized message resource modules.



Compo...	ComponentId	Directory_	Attributes
CtxCpu	{956F4138-5894-44F3-9B59-AED8CD71001A}	CTX_MF_CTXCPU_INSTALLDIR	0

Condition

```
VersionNT=502 And (PHYSICAL_CPU_COUNTS.6ACBB81A_4DD0_4685_AF9D_4115665925C4>1 OR PHYSICAL_CPU_COUNTS.A7B20831_09B7_47D7_BE0B_C7F51B8A3258>1)
```

Figure 33. CtxCpu components' install conditions (mps.msi)

Software Design & Configuration

Every product has a unique design in terms of usability and interactivity. Access to the offered functionalities or services is provided with the help of event-triggering objects like buttons and keyboard actions. In window-based applications, functionalities are being categorized and generally come with a user interface where the user can place desired inputs or see the output of the tasks. Depending on software design architecture, some user actions may not be made available if certain conditions are not met or are only made available for a very specific context.

Context menus

In some cases, only applicable menus are shown and other menus not applicable to the current system configuration or selected object are made invisible (i.e. visible=false) or intentionally not included (i.e. menuObject=null) in the UI. If menu items are disabled instead of hiding or making them unavailable, testers could still be able to find and satisfy the necessary condition/s to enable them; thus enabling the tester or the automation tools to continue testing on that branch. Figure 34 shows an example of a "New Zone create" context menu in XenApp Management Console which is only made available in the popup menu if there are more than 1 servers added in the farm.

For the case of hidden or unavailable menus, certain knowledge has to be communicated from the solution architect or developer to the application tester to be able to find ways to satisfy the required conditions to entering another test path or conditional branch. In a pure block box testing, hidden menus are prone to be often mis-looked and left untested.

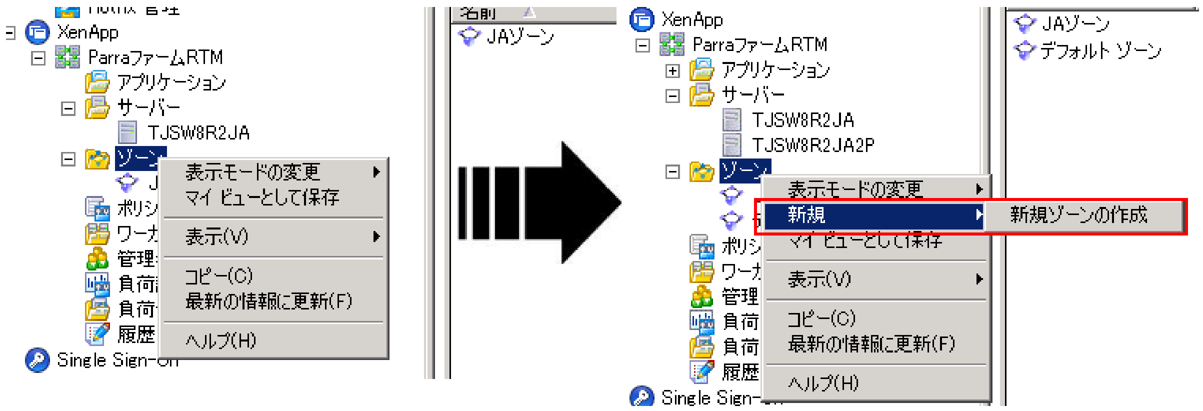


Figure 34. Zone create context menu shown only on specific condition

Complex UI structure

Full test path testing cannot be achieved thru a black box testing. Gray or white-box testing methodologies *may* enable testers or automation tools to cover a full test path but purely depends on the size and scale of the product-under-test. User interfaces are logically structured in a tree-like manner as shown in the example with the initial application window instance being the root UI element providing access to its peer and child windows thru events (e.g. menu, keyboard, timer events). Events to show up another UI object or window state is thrown when a condition or a specific set of conditions are satisfied. A complete map of all UI windows and their relations has to be known prior to product testing. Corresponding conditions on every branch (or edge) need to be specified in the map to be able to test the specific application state.

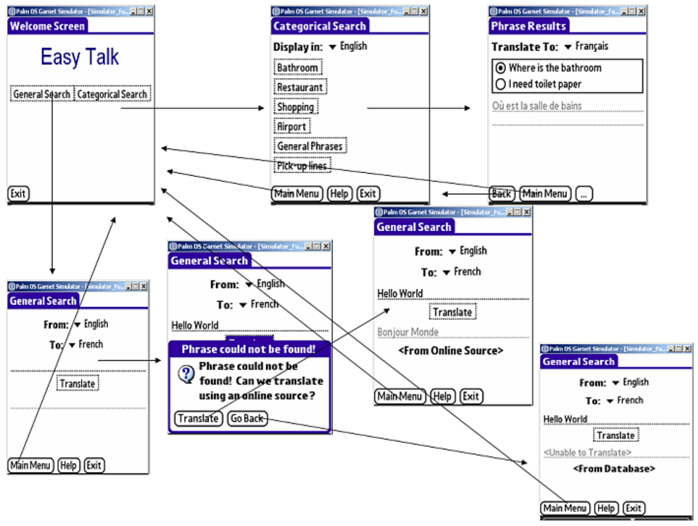
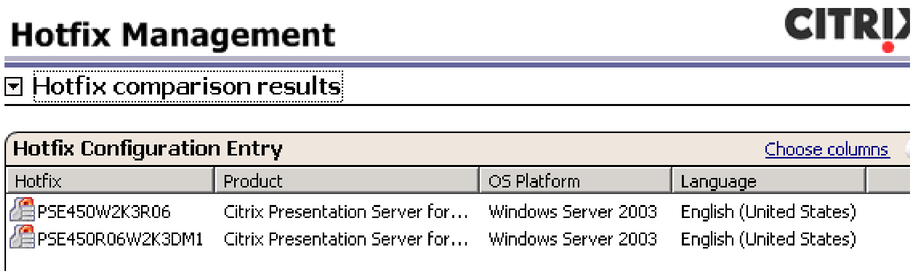


Figure 35. UI map example

Object/State Dependency

In some cases, certain functionalities or features of a product may not be available or accessible without the presence of an external object. These objects or state could be something that will be available later during the development phase or later after a certain usage condition is satisfied. For example, a log viewer will need a log file to be tested of its functionalities. This object i.e. the log file may not be available upon the very initial state of the system and its creation might have a dependency to system's configuration and usage. Another example is a database compact purge feature of a management console which might only be enabled or available when the database record size grows to certain size threshold e.g. 1GB.

With XenApp server, patches or hotfixes can only be created only after the product is released. Dummy patches are created in the latter phase of the development cycle and seems to be used primarily for product patch-ability testing. XenApp management console has a set of UI windows for administering installed hotfixes in the server. Some UI elements in these hotfix administration windows are only accessible with the presence of an installed hotfix objects. With dummy hotfixes being created in the later phase of the project (i.e. new RC timeframe), it would be too late to make the necessary UI changes into the application if there'll be UI issues found during dummy testing. Thus, software objects required for specific UI testing should be known and prepared beforehand. In cases when the required objects cannot be made available, static resource UI verification may serve the least alternative for those UI objects which cannot be verified on the actual application runtime.


Hotfix Management

Hotfix comparison results

Hotfix Configuration Entry

Hotfix	Product	OS Platform	Language
PSE450W2K3R06	Citrix Presentation Server for...	Windows Server 2003	English (United States)
PSE450R06W2K3DM1	Citrix Presentation Server for...	Windows Server 2003	English (United States)

Figure 36. Hotfix management dialogs in XA

UI testing coverage assessment

In UI testing, it is ideal to have the list of all the UI elements in the product. Basing on that list, it will be possible to know the coverage of the current testing scope. If testcases were created basing on a pure black box interaction with the product-under-test, other areas not accessible due to some cases like of those described in the previous section, will not be part of that coverage. Risk for UI related issue increases if not covered in the testcases.

Using static resource UI verification, there are some cases where this methodology will help in finding UI elements not covered in testcases. Resource UI binaries found to have some un-localized strings may not have been part or covered into the testcases. Basing on the un-localized resource key or embedded resource/class name, we can deduct where the resource is used by the context of its name or the context described in the value itself. The module's filename and even the resource type also serve as a good hint to where the resources are intended to be used. With the help of deduction, the features or scenarios which are related to these UI objects can be determined and may be used in doing certain level of test case coverage assessment.

Deducting from filename & resource type

Filenames like *HDXFlashEventMessages.dll* would imply that the resource UI module contains event messages and which specific feature/s it is related (which for this case is related to HDX Media Flash). Looking into the contained resource types inside the module will give more info e.g. RT_MESSAGETABLE resources are basically used in event logs and possibly are not used within another UI object like dialogs or windows.

Deducting from embedded resource name

.Net's embedded resource names also serve as a hint in determining the scenarios where the resources are used. The example shown in Figure 37 is a part of PSE.dll module's resource dump. The embedded resource name "Citrix.CMI.PSE.Cmo.Zone.View.NewZoneStartPage" is quite descriptive enough to denote that the resources contained are being used or related to creating new zone scenario in XA management console.

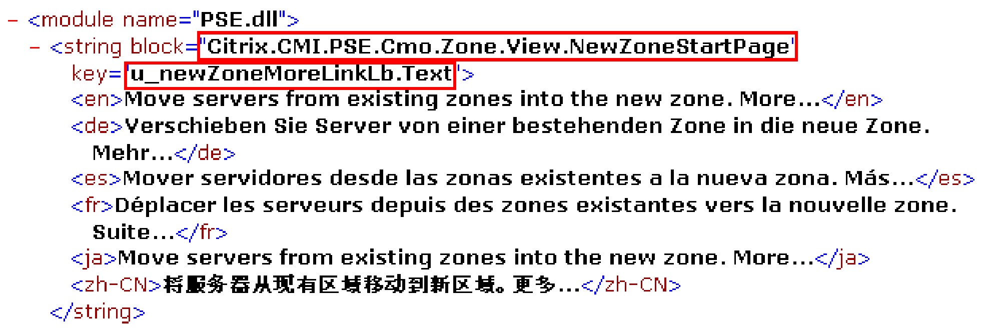

```
- <module name="PSE.dll">
  - <string block="Citrix.CMI.PSE.Cmo.Zone.View.NewZoneStartPage"
      key="u_newZoneMoreLinkLb.Text">
      <en>Move servers from existing zones into the new zone. More...</en>
      <de>Verschieben Sie Server von einer bestehenden Zone in die neue Zone.
        Mehr...</de>
      <es>Mover servidores desde las zonas existentes a la nueva zona. Más...</es>
      <fr>Déplacer les serveurs depuis des zones existantes vers la nouvelle zone.
        Suite...</fr>
      <ja>Move servers from existing zones into the new zone. More...</ja>
      <zh-CN>将服务器从现有区域移动到新区域。更多...</zh-CN>
    </string>
```

Figure 37. Description embedded resource name in .Net's module

Deducting from resource key

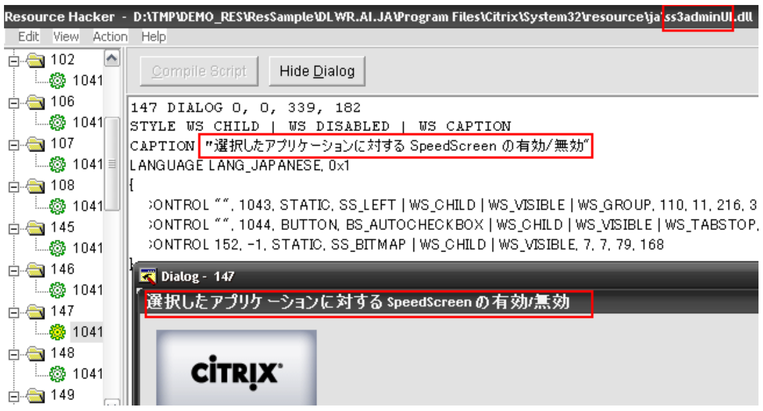
Moreover, the resource key names used in .Net's resource modules are also descriptive enough in knowing which specific control they are used or related with. For example, on the same Figure 37, it is pretty obvious that the resource string is used in the Text property of u_newZoneMoreLinkLb control. The resource key name does help in locating where the un-localized text is used in specific forms, pages or dialogs.

Deducting from resource string value

The functionality or feature to which a specific resource string value is used may be deduced basing on the context of the string value itself in addition to the useful hints described previously. Win32 resource modules generally won't have a descriptive block or key names on its resource entries. The resource filename and string value (or set of values) may help in the deduction process.

Deducting from UI resource element

In other cases where the resource types other than text are being used, it may be possible to determine the features or functionalities to which they could be related with by using the available template resources. Template resources may have labeled controls and captions available; and are useful in deducting the scenarios or functionalities they are used. Figure 38 shows an example of a dialog template included into Delaware's ss3adminUI.dll. By looking into the dialog's caption strings, it suggests its connection to the SpeedScreen configuration functionality.


Resource Hacker - D:\TMP\DEMO_RES\ResSample\DLWR.AI.JA\Program Files\Citrix\System32\resource\ja\ss3adminUI.dll
Edit
View
Action
Help
102
1041
106
1041
107
1041
108
1041
145
1041
146
1041
147
1041
148
1041
149
Compile Script
Hide Dialog
Dialog - 147
選択したアプリケーションに対する SpeedScreen の有効/無効

```
147 DIALOG 0, 0, 339, 182
STYLE WS CHILD | WS DISABLED | WS CAPTION
CAPTION "選択したアプリケーションに対する SpeedScreen の有効/無効"
LANGUAGE LANG_JAPANESE, 0x1
{
  ;ONTROL "", 1043, STATIC, SS_LEFT | WS_CHILD | WS_VISIBLE | WS_GROUP, 110, 11, 216, 3
  ;ONTROL "", 1044, BUTTON, BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
  ;ONTROL 152, -1, STATIC, SS_BITMAP | WS_CHILD | WS_VISIBLE, 7, 7, 79, 168
}
```

Figure 38. Captions and labels in dialog templates

Verification of "untestable" UI

As mentioned in the previous sections, not all UI features may be covered or passed thru by running the regression testcases alone. Branching conditions and other system state dependencies should be fulfilled for some of the UI objects to be enabled or shown to the user. Dependency to unavailable objects during the time of testing will make some of the UI objects unreachable and consequently left untested.

Static resource UI verification may not be able to generate the actual UI representation, but could help in locating un-localized UI elements used in certain UI window forms. Applications using menu and dialog templates can be statically verified using available Windows API or tools capable of viewing them like Resource Hacker. It is recommended to run at least a static resource UI check on UI objects which seems to be "un-testable" during the test execution.

Catching build or code-integration issues

Running UI resource verification on every build will also help in finding build and/or code-level integration regression issues. In cases of automatic or mass code integrations, there is a great risk for code reversion which causes not only UI but functionality regression issues as well. Build related issues may also occur due to developer's error like referencing incorrect file paths and also in other build system configuration changes e.g. file codepage. An example of system change was the switch from the legacy CIBA system to HFDB build system which caused several issues on the output builds.

There are also some cases when a need to track back and find the build to which the regression or a change in the UI was introduced especially when an issue is found on the later builds.

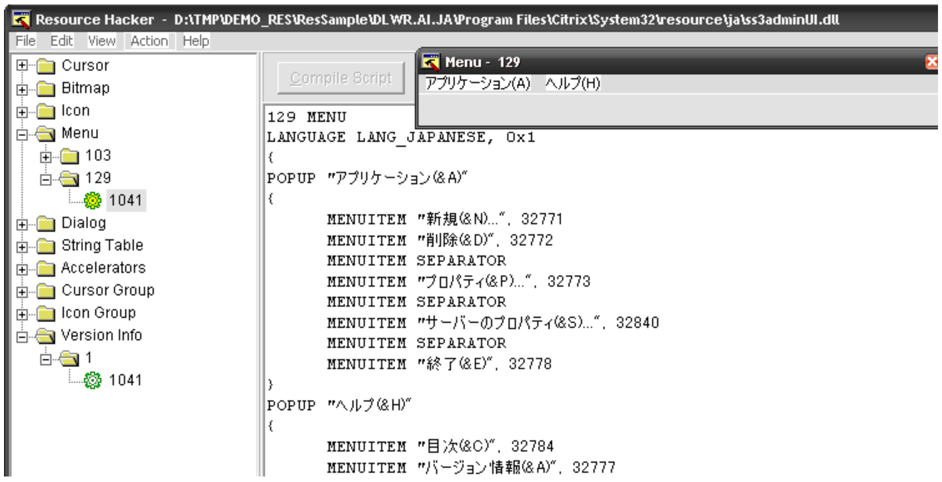
Tools & Utilities

There are several 3rd party and in-house tools which may help in doing the described static resource UI verification. Currently, there's no tool available which could do all of the verifications described in the previous sections. This section lists some of tools used within the GS product and LCM team.

3rd-Party Tools

Resource Hacker

Resource Hacker is a freeware GUI-based resource editor for Win32 modules. It is GUI-based application which has some functionality for viewing and editing resource entries. This tool is recommended especially for visual verification of UI objects like dialog/menu templates as well as icon and bitmap resources. This tool however has some difficulty in reading Russian text resources and in some cases cannot read some dll modules like *HDXFlashEventMessages.dll* of XenApp.



```
129 MENU
LANGUAGE LANG_JAPANESE, 0x1
{
POPUP "アプリケーション(&A)"
{
        MENUITEM "新規(&N)...", 32771
        MENUITEM "削除(&D)", 32772
        MENUITEM SEPARATOR
        MENUITEM "プロパティ(&P)...", 32773
        MENUITEM SEPARATOR
        MENUITEM "サーバーのプロパティ(&S)...", 32840
        MENUITEM SEPARATOR
        MENUITEM "終了(&E)", 32778
}
POPUP "ヘルプ(&H)"
{
        MENUITEM "目次(&C)", 32784
        MENUITEM "バージョン情報(&A)", 32777
```

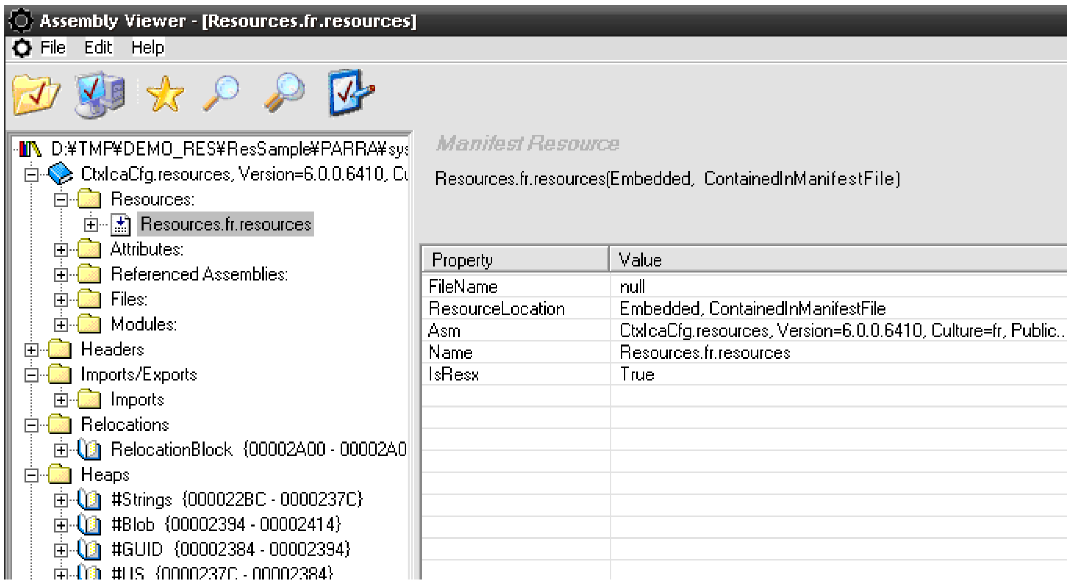
Figure 39. Screenshot of Resource Hacker

.Net Reflector

The .NET Reflector is a commercial utility for exploring and analyzing .Net assemblies, and boasts it decompiling features. It has a free version with limited functionalities.

Asmex

Asmex is an open-source .Net assembly explorer (or browser) which is used primarily for examining classes and module analysis. It has a good GUI for viewing the internal structure and information of an assembly including the embedded resources.



```
D:¥TMP¥DEMO_RES¥ResSample¥PARRA¥sys
  CtxIcaCfg.resources, Version=6.0.0.6410, Cu
    Resources:
      Resources.fr.resources
    Attributes:
    Referenced Assemblies:
    Files:
    Modules:
  Headers
  Imports/Exports
    Imports
  Relocations
    RelocationBlock {00002A00 - 00002A0
  Heaps
    #Strings {000022BC - 0000237C}
    #Blob {00002394 - 00002414}
    #GUID {00002384 - 00002394}
    #US {0000237C - 00002384}
```

Manifest Resource

Resources.fr.resources(Embedded, ContainedInManifestFile)

Property	Value
FileName	null
ResourceLocation	Embedded, ContainedInManifestFile
Asm	CtxIcaCfg.resources, Version=6.0.0.6410, Culture=fr, Public..
Name	Resources.fr.resources
IsResx	True

Figure 40. Screenshot of the Asmex tool

In-house Tools

ResourceValidator

ResourceValidator is a command-line application for checking resource entries consistency between the base language (EN) and the localized satellite resource files. It can export resource strings from both Win32 and .Net resource assemblies. It can detect inconsistent resource ID and un-localized entries (i.e. resource strings are still in English texts). This tool may serve useful in some areas of UI testing like cross-language comparison, but fall short in some areas which deemed necessary.

Limited resource type support

It supports only RT_MESSAGETABLE resource in Win32 modules and doesn't include RT_STRING resources into it output. Other possible resource types which need to be considered for string resource check are RT_VERSION, RT_DIALOG, RT_ACCELERATORS, (RT_FONT) and RT_MENU.

Limited features

It only supports cross-language comparison and two types of issue detections. More support for cross-language issue detection and cross-release comparison is desirable.

Portability and configurability

It is implemented in .Net and C++ platforms. It uses separate external dlls for its Win32 and .Net resource enumeration. It also requires an application config.xml where the parameters should be set. It would be hard to update parameters during runtime if this tool is used within a script.

smrdump/resdump (GS LCM)

smrdump is a console app which dumps Win32 RT_MESSAGETABLE and RT_STRING resources into the system console. resdump is also a console app which dumps .Net strings (ResourceTypeCode.String) into the system console as well. Console outputs may be saved by redirecting the system output stream.

enumres (GS LCM)

New command-line based tool for exporting strings resources in a Win32 module as well as .Net resource assembly into an xml file. Currently, it supports dumping resource types Win32 RT_MESSAGETABLE and RT_STRING, as well as .Net .Net strings (ResourceTypeCode.String). Support for additional Win32 resource types (version, dialog, menu and accelerators) is planned in the future release.

rescmp (GS LCM)

rescmp is console app for comparing and verifying possible issues in the resource modules using the exported XML files by enumres utility. It produces a merged xml files for side-by-side comparison of the resources entries, and an output xml file for possible issues found in the subjects. It supports both cross-release and cross-language file comparison and have functionalities for detecting possible issues on both comparison types.

Detection features

- Unlocalized modules – modules which have resources but do not have localized satellite files
- Unlocalized resource entries – entries whose value is similar to base (EN) language
- Missing resource entries – entries not found in the localized satellite resources
- New resource entries – entries found in base (EN) modules but not found in any localized files
- Duplicate resource key – entries whose key (or id) are the same
- Duplicate resource value – entries of different keys whose values are similar
- Empty resource value – entries having an empty value
- Grouped Items – group items according to the specified key prefix e.g. “items”

resport (GS LCM)

This is a command line prototype for exporting dialog template to bitmap image. It simply enumerates the RT_DIALOG resources, call Windows API to create a window instance of the template and get the screenshot of that window. (still need a lot of work).

Future Plans

There are still several things which need to be added into the new toolset (enumres, rescmp and resport). The following are the list of features which are being planned:

- **enumres**
    - additional Win32 resource type support (exporting the text parts)
        - RT_VERSION: all file version properties
        - RT_DIALOG: dialog captions and dialog items' text label
        - RT_MENU: text labels of each menu and menuitems
        - RT_ACCELERATOR: accelerator key characters
- **rescmp**
    - additional pattern detection & comparison
        - formatting tag order check and compare
        - command-line switch pattern
- **resport**
    - port functionality to enumres

Summary

As explained in the previous sections, there is a great advantage and benefit in running Resource UI verification within the G11N testing. As seen in the many examples, it does help in finding possible issues at the earlier stage of development. In a different perspective, it helps a lot in assessment the uncovered test scenarios and also help in tracing UI related regression issues. There could still be more benefits other than those mentioned in this document but those reasons are quite enough to accept that the result it brings will surely help in the efforts to improve the quality of the localized products.

References

1. “Windows Resource Files”, http://msdn.microsoft.com/en-us/library/cc194804%28v=MSDN.10%29.aspx
2. “Understanding MUI”, http://msdn.microsoft.com/en-us/goglobal/dd218459.aspx
3. “Globalization Step-by-step”, http://msdn.microsoft.com/en-us/goglobal/bb688110.aspx
4. “Microsoft .Net I18N”, http://msdn.microsoft.com/en-us/goglobal/bb688096.aspx
5. CJKV Information Processing, Ken Lunde, O’Reilly
6. “ResourceValidator” in CitrixWiki, http://citrixwiki/ResourceValidator
7. Human computer interaction: Issues and Challenges, Qiyang Chen, http://books.google.co.jp/books?id=nS5Lp7BgR80C&lpg=PA25&dq=UI%20testing&hl=en&pg=PA20#v=onepage&q=UI%20testing&f=false
8. GS Docs on UI Methodologies, http://sharepoint.citrite.net/sites/Engineering/groups/g11n/public/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2fsites%2fEngineering%2fgroups%2fg11n%2fpublic%2fShared%20Documents%2fDepartments%2fTest%2fInterface%20Model%2fDocuments%2fUI_Methodologies

For Your Reading

1. .NET Internationalization: The Developer's Guide to Building Global Windows and Web Applications, Guy Smith-Ferrier, Addison-Wesley Professional
2. Usability and Internationalization of Information Technology, Nuray Aykin, CRC Press, USA
3. Usability and Internationalization - HCI and Culture, Nuray Aykin, Second International Conference on Usability and Internationalization, Springer

Useful Links

1. Citrix Usability Team page: http://ftlengreddot01/UICS/4.htm