

NLS and Localization Sample

*This sample is compatible with the Microsoft Game Development Kit (June 2020)*

# Description

This sample demonstrates how to localize strings and assets referenced in the MicrosoftGame.Config, as well as how to properly access localized in-game resources. Some common NLS (National Language Support) APIs are demonstrated as well.

## Main Screen



*In this image the console user’s language setting is es-AR*

|  |  |  |
| --- | --- | --- |
| Action | Gamepad | Keyboard |
| Select the button to run | D-Pad Up/Down | Arrow key/Mouse |
| Press the button | A Button | Enter/Left click |
| Exit | View Button | Esc |

# Building the sample

If using an Xbox One devkit, set the active solution platform to Gaming.Xbox.XboxOne.x64.

If using an Xbox Series X|S devkit, set the active solution platform to Gaming.Xbox.Scarlett.x64.

If using Windows 10, set the active solution platform to Gaming.Desktop.x64.

*For more information, see* Running samples*, in the GDK documentation.*

# Implementation notes

# This sample demonstrates the basics of localizing a title. The resource parser used in the sample is very basic and does not offer any error checking. If a game intends to follow this setup it should not copy this resource parser directly but instead see this as an example of how a more fully-featured resource parser could function in this context. The goal of the sample is to help developers familiarize with the procedure for localizing resources.

**Enumerate current localization settings**

# When the application is launched, it will display the output of some common NLS APIs. You can manually display them again by pressing “Enumerate current localization settings” button.

The GetUserDefaultLocaleName API retrieves the locale that the console is set in and the GetUserDefaultGeoName API retrieves the location that the console is set in. The XPackageGetUserLocale API retrieves the user locale that most closely matches the package locale. The result can be converted into LCID.

**System settings-driven language selection**

The sample will change its in-game image and text based on its application language settings. The sample’s default language is determined by the XPackageGetUserLocale API. The XPackageGetUserLocale API should be a game’s source of truth for locale selection as it will determine the locale that is the best match for the title on the user’s system. The locale is based on all the available data including console and user settings and what is supported by the title. The result will always be a language that is declared in the game config or a user language if no languages are declared. This locale is then used by the sample at runtime to select the appropriate localized image and text to display.

In the sample, 7 locales are defined in the MicrosoftGame.config:

* en-US
* en-GB
* zh-Hans-CN
* zh-Hant
* ja-JP
* es
* fr

These locales are selected to demonstrate common fallback scenarios. A fallback occurs when the console language and location setting does not exactly correspond to languages supported by the game and it needs to select from the available options.

For example, when the console language is “en-CA”, it will fallback to “en-GB”. Another scenario is with “fr”, where in this sample, only “fr” without region is defined. In this case, when the console language is French, it will fallback to “fr” no matter what the console language region is.

Note console settings divide the localization settings between Language, Language region and Location. On Xbox One Manager, Preferred language merges the first two and Geographic region corresponds to Location. For this sample, only the language settings affect the language selection; location/geographic region only affects GetUserDefaultGeoName.

**MicrosoftGame.Config string localization**

The display name and description of the sample are also localized based on the console language. To support localization these fields in the MicrosoftGame.Config set to ms-resource references such as:

OverrideDisplayName="ms-resource:ApplicationDisplayName"

These values are then filled in based on the values present in the Resources.resw files underneath their respective locale. For example, if the console language is set to Japanese (ja) and region to Japan (JP) then the display name will be pulled from the Resources.resw file in the ja-JP folder of the project, which for this sample would be the string “NLS And Localization (ja-JP)”. By default, the Resources.resw file and each language folder are expected to be at the root project folder. If the Resources.resw files are in another location such as in the “String” directory used by this sample (e.g. ProjectFolder\Strings\Resources.resw, ProjectFolder\Strings\ja-JP\Resources.resw, etc. ), please make sure to specify the folder in “Package Localization Dir” property in the project property page.

**MicrosoftGame.Config package image localization**

The package images of the sample can also be localized based on the console language. First, the path for the default images is specified as usual in MicrosoftGame.config. The below snippet indicates where the default images are in the sample, under the “Images” directory:

StoreLogo="Images\StoreLogo.png"

Square480x480Logo="Images\Square480x480Logo.png"

Square150x150Logo="Images\Square150x150Logo.png"

Square44x44Logo="Images\Square44x44Logo.png"

SplashScreenImage="Images\SplashScreen.png"

Then, in the same directory as the default images, subdirectories for each language defined in the MicrosoftGame.config will contain the localized variants (e.g. ProjectFolder\Images\ja-JP\StoreLogo.png, ProjectFolder\Images\ja-JP\Square480x480Logo.png, etc.). For any language, it first checks for the image file in the particular language folder and uses that file, if it is available. Otherwise, if the image is missing from the language folder or the language isn’t supported (that is, if the language folder is missing), it falls back to the default logo. Please make sure that each logo is correctly copied to Loose folder when the sample is built.

Note that if the product corresponding to the MicrosoftGame.config settings is actually published and an account in that sandbox (or retail) is signed in, strings and images can be overridden by equivalent fields configured in Partner Center product pages. There also does not need to be a 1 to 1 correspondence of languages supported by the game and Store listings in Partner Center, but it would be very helpful.

# Update history

April 2020 Initial release

May 2021 Sample is updated to support additional NLS functionalities including UX refresh

July 2021 MicrosoftGame.config is updated

# Privacy statement

When compiling and running a sample, the file name of the sample executable will be sent to Microsoft to help track sample usage. To opt-out of this data collection, you can remove the block of code in Main.cpp labeled “Sample Usage Telemetry”.

For more information about Microsoft’s privacy policies in general, see the [Microsoft Privacy Statement](https://privacy.microsoft.com/en-us/privacystatement/).