

Design of I18nService module

author: Jan Trejbal, Sun Microsystems, Inc.
2009/01/15 Initial version

Table of Contents

1	Introduction of I18nService module.....	2
2	Relation to other modules.....	2
3	Requirements.....	2
4	Implementation.....	3
5	Public Interfaces.....	3
5.1	Summary.....	3
5.2	Type definitions.....	4
5.3	Functions.....	5
5.4	Error handling.....	6
5.5	Deleted interface functions.....	7
5.6	Static data structures (private).....	7
6	Command-line interface (CLI).....	8

1 Introduction of I18nService module

Purpose of I18nService (I18n = Internationalization) module is to provide data about system languages and locales, and some other I18n-related services.

Including information about packaging of languages and locales, to be able to install and remove them.

2 Relation to other modules

I18nService module will be consumed by the installer application via existing orchestrator wrappers. The module can also be used directly by other modules or even other applications running before or after system installation.

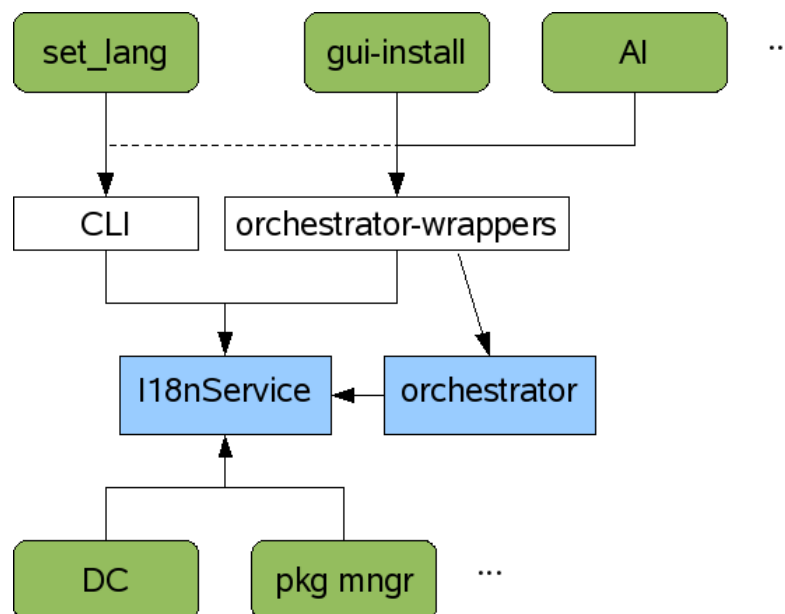
Command-line interface is another way how I18n services will be exposed. CLI will be used by other than C consumers (e.g. set_lang script) as a full-featured API wrapper.

CLI could also be used by QE team as a test driver.

Examples of consumers of the module follows:

- LiveCD start-up language selection (set_lang script)
- gui-install - language, confirm, and install screens
- AI, e.g. when validating install profile which includes languages list
- DC, needs list of pkgs to support a particular language/locale in the image
- package manager or other OpenSolaris languages install mngmt GUI/UI

Drawing shows where I18nService module fits in:



3 Requirements

I18nService module provides following functionality:

- 1) List languages supported by the installer application, including information about:
 - language code, text description, and translated description
- 2) List languages and locales available on the system, including information about:
 - language to locales relation
 - language and locale code, text description, and translated description
 - locale territory/country, code-set, and variant
 - default locale per language
- 3) List languages and locales available at network repository, including the same information as in the item 2) above.
- 4) Get a list of packages which must be installed/removed on the system, in order to support given language(s) or locale(s).
- 5) Set language or locale.
- 6) Get and Set system default locale.
- 7) Recognize and differ Unicode (UTF-8) and C/Posix locales.

4 Implementation

I18nService module will be implemented as a stand-alone shared C library.

Most of code will be taken from orchestrator locale.c component and will be reused. The locale.c interface and code will be deleted and completely moved to I18nService library. Unused interface functions and some private locale.c code will be deleted. Also, more comments will be put to the code to make it better readable.

Additionally, some new functions will be introduced - requirements #3 and #4. Since no install management service for language/locale bits is currently available, the new functions will be implemented using simple static table (languages, locales, package names).

Ultimately, requirements #3 and #4 will be implemented using a new, IPS-based, languages and locales install management layer. That's another, independent project. Currently, possible solution using the IPS facets, variants, and filters is being discussed at pkg-discuss community.

When the new IPS-based layer will be ready, implementation of the I18nService module will most likely need to be updated. Moreover, Python wrappers of the C module may be introduced, or furthermore module may be rewritten in Python.

5 Public Interfaces

This section defines C interfaces exposed by I18nService module. Basic implementation information is also included (new/reused/updated code).

5.1 Summary

Name	Short description
locale_info_t	Locale information structure
lang_info_t	Language information structure
ii_init_i18nsvc()	Initialize I18n service
ii_get_install_langs_info()	List langs supported by installer application
ii_get_langs_and_locs_info()	List langs and locales available on the system
ii_get_repo_langs_and_locs_info()	List langs and locales available at net repository
ii_get_lang_pkgs()	List packages related to given lang(s)
ii_get_locale_pkgs()	List packages related to given locale(s)
ii_set_lang()	Set language or locale
ii_set_system_default_locale()	Set the system default locale
ii_is_cposix_locale()	Classify Unicode (UTF-8) locales
ii_is_unicode_locale()	Classify C/Posix locale
ii_free_lang_info()	Free lang_info_t structures list
ii_free_locale_info()	Free locale_info_t structures list
ii_get_error()	Get error code of the last function call

5.2 Type definitions

- **locale_info_t**
 - Locale information structure.
 - Reuses code from orchestrator locale.c type "locale_info_t", and amends additional fields.

```
typedef struct locale_info {
    char    *locale_name;    /* locale code, e.g. "en_CA"    */
    char    *locale_desc;   /* locale name, e.g. "French (Canada)" */
    char    *locale_country; /* territory/country code, e.g. "CA" */
    char    *locale_codeset; /* code-set, e.g. "UTF-8", "ISO8859-1" */
    char    *locale_variant; /* locale variant, e.g. "@euro" */
    boolean_t def_locale;   /* is this default locale for the lang */
    struct  locale_info next;
} locale_info_t;
```

Following `locale_info_t` fields are new to the original orchestrator `locale.c` code:

```
char    *locale_country;
char    *locale_codeset;
char    *locale_variant;
```

- **lang_info_t**
 - Language information structure.
 - Reuses code from orchestrator `locale.c` type "lang_info_t".

```
typedef struct lang_info {
    char    *lang;      /* lang code, e.g. "en", "fr" */
    char    *lang_name; /* lang name (translated), e.g. "French" */
    boolean_t def_lang; /* is this the default language */
    int     n_locales; /* number of locales for the language */
    locale_info_t *locale_info; /* pointer to all locales for the lang */
    struct  lang_info *next;
} lang_info_t;
```

5.3 Functions

- **int ii_init_i18nsvc(char *root_path)**
 - Initialize the I18nService module.
 - New code.
- **lang_info_t *om_get_install_langs_info(int *total)**
 - Returns linked list containing information about all languages supported by the installer application. Language names are translated appropriately.
 - Reuses code from orchestrator `locale.c` func. "om_get_install_lang_info()".
- **lang_info_t *ii_get_langs_and_locs_info(int *total)**
 - Returns linked list containing information about all languages, and all locales available on the system.
 - Reuses `locale.c` func. "om_get_lang_info()".
- **lang_info_t *ii_get_repo_langs_and_locs_info(char *repo, int *total)**
 - Returns linked list containing information about all languages and all locales available at given network repository.
 - New code.
- **char **ii_get_lang_pkgs (lang_info_t *langsp, int *total_pkgs)**
char **ii_get_locale_pkgs(locale_info_t *localesp, int *total_pkgs)
 - Returns a list of names of packages which must be installed/removed on the system, in order to support given language(s) or locale(s).

- New code.
- **int ii_set_lang(lang_info_t *localep)**
 - Sets language or locale using given info.
 - Reuses locale.c func. “om_set_install_lang_by_value()”.
- **int ii_set_system_default_locale(locale_info_t *localep)**
 - Sets the system default locale (in etc/default/init) using given locale.
 - Partly reuses code from libict ict.c func. “ict_set_lang_locale()”.
- **boolean_t ii_is_cposix_locale(locale_info_t *localep)**
 - Determine if given locale is the C/Posix locale.
 - Reuses and moves code from gui-install orchestrator-wrappers.c func. “orchestrator_om_locale_is_cposix()”.
- **boolean_t ii_is_unicode_locale(locale_info_t *localep)**
 - Determine if given locale is a Unicode (UTF-8) locale.
 - Reuses and moves code from gui-install orchestrator-wrappers.c func. “orchestrator_om_locale_is_utf8()”.
- **void ii_free_lang_info (lang_info_t *langp)**
void ii_free_locale_info(locale_info_t *localep)
 - Frees memory associated with given language/locale linked list.
 - Reuses locale.c func. “om_free_lang_info()” and “om_free_locale_info()”.

5.4 Error handling

Basic status of I18nService library calls can be determined using function return values. In case of error every function returns:

- either II_FAILURE (or II_SUCCESS if the call is successful),
- or NULL (e.g. when no languages/locales/etc. were found).

Following interface function provides more details on the encountered error:

- **int ii_get_error()**
 - Returns error code of the last library function call. A list of possible error codes follows.

Error code	Description
II_SUCCESS	Function call finished successfully
II_FAILURE	Error encountered, no further details available
II_NO_LOCALE_DIR	No locale dirs were found
II_PERMS	Permission error (when reading locale dirs/files)
II_TOO_MANY_FD	Too many open files
II_NOT_LANG	Given language does not exist (ISO 639)

II_INVALID_LANG_LIST	Given language list is empty or not valid
II_INVALID_LOCALE	Given locale is not valid
II_NO_SPACE	Memory allocation error
II_INVALID_LOCALES_ROOT	Invalid locales root path specified

5.5 Deleted interface functions

Following functions will be deleted from the original orchestrator locale.c interface. Those functions are obsolete and are not called from any of Slim source code.

```

char      **om_get_install_lang_names  (int *total)
char      **om_get_lang_names         (int *total)
locale_info_t  *om_get_locale_info    (char *lang, int *total)
char      **om_get_locale_names       (char *lang, int *total)
void       om_save_locale              (char *locale, boolean_t install_only)
int        om_set_install_lang_by_name (char *lang)
int        om_set_default_locale_by_name (char *localep)
void       om_free_lang_names          (char **listp)

```

5.6 Static data structures (private)

Following static lists will be part of the I18nService module. Those lists will be taken from orchestrator_lang_codes.h header file, which will be deleted. Note, the lists are private data and are not exposed as part of public interface.

- **ii_languages_list[]**
 - Static array of all language names and codes (ISO 639).
 - Reuses orchestrator_lang_codes.h var. “orchestrator_lang_list[]”.

```

struct iso639_1_languages {
    char *lang_code;
    char *lang_name;
} ii_languages_list[] = {
    ...
    {"en", "English"},
    ...
}

```
- **ii_countries_list[]**
 - Static array of all country/territory names and codes (ISO 3166).
 - Reuses orchestrator_lang_codes.h var. “orchestrator_country_list[]”.

```

struct iso3166_countries {

```

```

        char *country_code;
        char *country_name;
    } ii_countries_list[] = {
        ...
        {"CA", "Canada"},
        ...
    }

```

6 Command-line interface (CLI)

All I18nService functionality will be available also via `i18nsvc_cli` command. That command will accept following options:

- `i18nsvc_cli -h`
Print usage information.
- `i18nsvc_cli -l [-R root_path] [-i]`
Print out languages available on the system.
 - i print languages supported by the installer application
 - R root_path set alternative root path
- `i18nsvc_cli -L [-R root_path] [-r repo]`
Print out languages and locales available on the system.
 - r repo print langs and locales available at given network repository
 - R root_path set alternative root path
- `i18nsvc_cli -p -g lang | -c locale`
Print out names of packages which must be installed/removed on the system, in order to support given language(s) or locale(s).
 - g lang languages list, use comma as languages delimiter
 - c locale locales list, use comma as locales delimiter
- `i18nsvc_cli -s lang|locale`
Set language or locale.
 - s lang|locale language or locale to be set
- `i18nsvc_cli -S locale`
Set the system default locale.
 - S locale locale to be set as the system default
- `i18nsvc_cli -c locale`
Determine if given locale is the C/Posix locale.
 - c locale locale to be examined
- `i18nsvc_cli -u locale`
Determine if given locale is a Unicode (UTF-8) locale.
 - u locale locale to be examined