Marvell Design IP

String Manager Configuration Guide

Software Development Kit

Revision 1.0

FOR CUSTOMER USE ONLY

Doc. No. NDCN-PA101450
December 13, 2011
CONFIDENTIAL
Document Classification: Restricted Distribution

Document Conventions

	**Note:** Provides related information or information of special importance.
 Caution	**Caution:** Indicates potential damage to hardware or software, or loss of data.
	**Warning:** Indicates a risk of personal injury.

For more information, visit our website at: www.marvell.com

Table of Contents

List of Tables

About This Document

Purpose

This document provides detailed directions on how to define languages, strings, and translations. It includes detailed code examples. Also covered are the topics of internal web pages and various http server concepts that are useful in providing fully localized web pages.

1 How to Define Languages, Strings, and Translations

1.1 How to Define Languages

Each language to be used by the system needs to be defined in the firmware in 2 places:

- Header file enumeration
- C file table

The enumeration of languages defined in oem/<oem>/common/string_mgr/config/string_mgr_config.h must contain a list of every language used in the system. An example of this enumeration is shown below. To define a new language (or to remove a language) simply edit this list and make corresponding edits to the language definition table described below.

```
/**
 * \brief This is a list of the languages supported in this product.
 *        Each language should have a corresponding entry in the table
 *        located in string_mgr_config_languages.c
 */
typedef enum {
    LANG_ENGLISH,
    LANG_SPANISH,
    LANG_FRENCH,
    ...
    LANG_ARABIC,
```

In addition to the enumeration, a language must also contain an entry in the table located in the file oem/<oem>/common/string_mgr/config/string_mgr_config_languages.c. This table should, for every language, define each field of the string_mgr_code_entry_t type. One such way to this is with the pre-processor macro and the example shown here. This C code file may be adjusted to meet the needs of the OEM.

```
#define _ENTRY(lang,id,rtl) {{.lang_code = LANG_ ## lang},\
                             .identifier = #id,\
                             {.right_to_left = rtl}}
static const string_mgr_code_entry_t _languages[] =
{
    _ENTRY(ENGLISH, en, 0),
    _ENTRY(SPANISH, es, 0),
    _ENTRY(FRENCH, fr, 0),
    ...
    _ENTRY(ARABIC, ara, 1),
};
```

```
static const string_mgr_code_tbl_t _language_tbl =
{
  .tbl  = _languages,
  .size = sizeof(_languages)/sizeof(_languages[0]),
};
```

The globally visible symbol, string_mgr_config_language_codes, must be defined as shown in the above example. It is via this symbol that the common code gains access to the table of language definitions.

1.2 How to Define Strings

Every string in the system is symbolically defined as an enumerated value. Each of these values also includes a “one size fits all” (OSFA) translation that is used when a specific language translation is not available.

The enumeration of strings is within the file oem/<oem>/common/string_mgr/config/string_mgr_config.h. There is no order or grouping required within this enumeration but it may help the maintainer to adopt some sort of convention since it is expected that the list of strings may grow to be quite large. An example of a portion of this table is shown below.

```
/**
 * \brief This is a list of the strings supported in this product.
 *        Each string should have a corresponding entry in the table
 *        located in string_mgr_config_strings.c
 */


typedef enum {
/***************************
* Status Strings
***************************/
    STRING_STAT_NONE,
    STRING_STAT_READY,
    STRING_STAT_INIT,
/***************************
* Generic Strings
***************************/
    STRING_GEN_YES,
    STRING_GEN_NO,
    STRING_GEN_OFF,
    STRING_GEN_ON,
    STRING_GEN_AUTO,
    STRING_GEN_AUTOMATIC,
```

```
/***************************
* Control Panel Strings
***************************/
    STRING_CP_MAINMENU,
    STRING_CP_ENGLISH,
    STRING_CP_FRENCH,
    STRING_CP_GERMAN,
    STRING_CP_CTRY_USA,
    STRING_CP_CTRY_FRANCE,
```

In addition to the simple enumeration, each string must have an entry in the table located in the file oem/<oem>/common/string_mgr/config/string_mgr_config_strings.c. This table must, for every string, define each field of the string_mgr_code_entry_t type. It is strongly suggested, for clarity, that the identifier value be a C-string representation of the str_code, but this is not required. In the example code that follows this is the case. This C code file may be adjusted to meet the needs of the OEM.

```
#define _ENTRY(code,osfa) {{.str_code = STRING_ ## code},\
                           .identifier = #code,\
                           {.osfa_str = osfa}}
static const string_mgr_code_entry_t _strings[] = {

    /* Status Strings */
    _ENTRY(STAT_NONE,""),
    _ENTRY(STAT_READY,"Ready"),
    _ENTRY(STAT_INIT,"Initializing..."),

    /* Generic Strings */
    _ENTRY(GEN_YES,"Yes"),
    _ENTRY(GEN_NO,"No"),
    ...
};
static const string_mgr_code_tbl_t _strings_tbl =
{
  .tbl  = _strings,
  .size = sizeof(_strings)/sizeof(_strings[0]),
```

The globally visible symbol, string_mgr_config_language_codes, must be defined as shown in the above example. It is via this symbol that the common code will gain access to the table of language definitions.

1.3 How to Define Translations

Translations are provided as text files, one file per language. These files are named strings. <language id> where the <language id> for a particular language is identifier field for that language as defined in

the language table described above. For instance, according to the examples given so far, translations for the English language (LANG_ENGLISH) would be given in a file named strings.en since the identifier for the LANG_ENGLISH language is given as “en” in the language table.

Each translation file contains lines of the form <string id>, <translated string>. Note that <string id> is not the string enumeration value (that is, STRING_GEN_AUTOMATIC) but is, rather, the identifier assigned to the string in the string table (that is, GEN_AUTOMATIC in the examples). This allows the enumeration values to remain somewhat generic even if the OEM wants to employ a different system of string naming for translation purposes.

Any line that does not match this pattern is ignored. In this method, comments or unused translations may easily be present in the translation files as necessary. It is also important to note that not all strings must be translated by all languages. In many cases, such as for technical terms, the OSFA translation (see above) may be acceptable and, therefore, a language specific translation may be skipped. The OSFA system also allows partially complete language translations to be used.

```
Here is an example translation file, strings.es. Only a few translations
are show.


GEN_YES,Si
GEN_NO,No
# The above may have been skipped since the OSFA translation of GEN_NO is
also “No”


# Note that translations can appear in any order (also note that I do not
speak Spanish)
STAT_NONE,Estatus es Nada
```

Finally each string file should be available, on the device, in the device directory /data/strings/string.<language id>. The best way to do this is to define a ROMFILES file in the configuration makefile as shown by the examples below.

```
ROMFILES += data/strings.en:strings/strings.en
ROMFILES += data/strings.es:strings/strings.es
...
```

1.4 The Use of Translated Strings with the Embedded Web Server

The OEM provided HTML files should use SSI tags to include translated strings. An example of this method is shown below. Note that the identifier of the string is STRING_<string id>. The necessary STRING_ prefix is side-effect of historic nature. It is important to understand that <string id> is the identifier value from the strings table and not the enumeration value. In the examples they are the same but they need not be and some OEMs may find it useful to use unmatched string identifiers for translation purposes.

```
<body>The translated value of GEN_YES is <i><!--#ssi IDs STRING_GEN_YES --
></i> </body>
```

The string translation language for the HTTP served HTML is chosen by first checking for the presence of a web browser supplied cookie named EWSlanguage. If such a cookie is provided and is set to a

valid <language id> then the HTML is translated into that language. If this cookie is not present or is not set to a recognized language then the HTML is translated into the system's default language.

A Revision History

Table 1: Revision History

Document Number	Revision	Description	Date
NDCN-PA101450	1.0	Initial release	12/12/2011

Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA

Tel: 1.408.222.2500
Fax: 1.408.752.9028

www.marvell.com

Marvell. Moving Forward Faster