NAME

Locale::Codes::API - a description of the callable function in each module

DESCRIPTION

Although there are several modules in the Locale-Codes distribution, all of them (as of version 3.10) use exactly the same API. As a matter of fact, each of the main callable modules are just wrappers around a central module which does all the real work.

ROUTINES

In order to maintain the documentation for the modules consistently, the functions are all documented here, rather than in the documentation for the separate modules.

The name of the function depends on the module. For example, every module contains a function "code2XXX" where XXX refers to the type of data. The values of XXX are described in the following table:

```
XXX MODULE

country Locale::Codes::Country
language Locale::Codes::Language
currency Locale::Codes::Currency
script Locale::Codes::Script
langext Locale::Codes::LangExt
langvar Locale::Codes::LangVar
langfam Locale::Codes::LangFAM
```

So, the Locale::Country module contains the function code2country, the Locale::Language module contains the function code2language, etc.

In all of the functions below, CODE refers to a code for one element in the code set. For example, in the two-letter country codes from ISO 3166-1, the code 'fi' is used to refer to the country Finland. CODE is always case insensitive (though when a code is returned, it will always be in the case as used in the standard), so 'fi', 'FI', and 'Fi' would all be equivalent.

CODESET refers to a constant specified in the documentation for each module to label the various code sets. For example, in the Locale::Language module, CODESET could be LOCALE_CODE_ALPHA_2 or LOCALE_CODE_ALPHA_3 (among others). Most functions have a default one, so they do not need to be specified. So the following calls are valid:

```
code2country("fi");
code2country("fi",LOCALE_CODE_ALPHA_2);
code2country("fin",LOCALE_CODE_ALPHA_3);
```

Since LOCALE_CODE_ALPHA_2 is the default code set, the first two are identical.

code2XXX (CODE [,CODESET] [,'retired'])

These functions take a code and returns a string which contains the name of the element identified. If the code is not a valid code in the CODESET specified then `undef` will be returned.

The name of the element is the name as specified in the standard, and as a result, different variations of an element name may be returned for different values of CODESET.

For example, the **alpha-2** country code set defines the two-letter code "bo" to be "Bolivia, Plurinational State of", whereas the **alpha-3** code set defines the code 'bol' to be the country "Bolivia (Plurinational State of)". So:

```
code2country('bo',LOCALE_CODE_ALPHA_2);
=> 'Bolivia, Plurinational State of'

code2country('bol',LOCALE_CODE_ALPHA_3);
=> 'Bolivia (Plurinational State of)'
```

By default, only active codes will be used, but if the string 'retired' is passed in as an argument, both active and retired codes will be examined.

XXX2code (NAME [,CODESET] [,'retired'])

These functions takes the name of an element (or any of it's aliases) and returns the code that corresponds to it, if it exists. If NAME could not be identified as the name of one of the elements, then `undef` will be returned.

The name is not case sensitive. Also, any known variation of a name may be passed in.

For example, even though the country name returned using LOCALE_CODE_ALPHA_2 and LOCALE_CODE_ALPHA_3 country codes for Bolivia is different, either country name may be passed in since for each code set, in addition to the alias 'Bolivia'. So:

```
country2code('Bolivia, Plurinational State of',
LOCALE_CODE_ALPHA_2);
=> bo

country2code('Bolivia (Plurinational State of)',
LOCALE_CODE_ALPHA_2);
=> bo

country2code('Bolivia',LOCALE_CODE_ALPHA_2);
=> bo
```

By default, only active names will be used, but if the string 'retired' is passed in as an argument, both active and retired names will be examined.

XXX_code2code (CODE ,CODESET ,CODESET2)

These functions takes a code from one code set, and returns the corresponding code from another code set. CODE must exists in the code set specified by CODESET and must have a corresponding code in the code set specified by CODESET2 or `undef` will be returned.

Both CODESETs must be explicitly entered.

```
country_code2code('fin', LOCALE_CODE_ALPHA_3,
LOCALE_CODE_ALPHA_2);
=> 'fi'
```

Note that this function does NOT support retired codes.

all_XXX_codes ([CODESET] [,'retired'])

These returns a list of all code in the code set. The codes will be sorted.

By default, only active codes will be returned, but if the string 'retired' is passed in as an argument, both active and retired codes will be returned.

all_XXX_names ([CODESET] [,'retired'])

These return a list of all elements names for which there is a corresponding code in the specified code set.

The names returned are exactly as they are specified in the standard, and are sorted.

Since not all elements are listed in all code sets, the list of elements may differ depending on the code set specified.

By default, only active names will be returned, but if the string 'retired' is passed in as an argument, both active and retired names will be returned.

SEMI-PRIVATE ROUTINES

Additional semi-private routines which may be used to modify the internal data are also available. Given their status, they aren't exported, and so need to be called by prefixing the function name with the package name.

These routines do not currently work with retired codes.

MODULE::rename_XXX (CODE ,NEW_NAME [,CODESET])

These routines are used to change the official name of an element. At that point, the name returned by the code2XXX routine would be NEW_NAME instead of the name specified in the standard.

The original name will remain as an alias.

For example, the official country name for code 'gb' is 'United Kingdom'. If you want to change that, you might call:

```
Locale::Codes::Country::rename_country('gb', 'Great Britain');
```

This means that calling code2country('gb') will now return 'Great Britain' instead of 'United Kingdom'.

If any error occurs, a warning is issued and 0 is returned. An error occurs if CODE doesn't exist in the specified code set, or if NEW_NAME is already in use but for a different element.

If the routine succeeds, 1 is returned.

MODULE::add_XXX (CODE ,NAME [,CODESET])

These routines are used to add a new code and name to the data.

Both CODE and NAME must be unused in the data set or an error occurs (though NAME may be used in a different data set).

For example, to create the fictitious country named ''Duchy of Grand Fenwick'' with codes ''gf'' and ''fen'', use the following:

```
Locale::Codes::Country::add_country("fe","Duchy of Grand Fenwick",
LOCALE_CODE_ALPHA_2);

Locale::Codes::Country::add_country("fen","Duchy of Grand Fenwick",
LOCALE_CODE_ALPHA_3);
```

The return value is 1 on success, 0 on an error.

MODULE::delete_XXX (CODE [,CODESET])

These routines are used to delete a code from the data.

CODE must refer to an existing code in the code set.

The return value is 1 on success, 0 on an error.

MODULE::add_XXX_alias (NAME ,NEW_NAME)

These routines are used to add a new alias to the data. They do not alter the return value of the code2XXX function.

NAME must be an existing element name, and NEW_NAME must be unused or an error occurs.

The return value is 1 on success, 0 on an error.

MODULE::delete_XXX_alias (NAME)

These routines are used to delete an alias from the data. Once removed, the element may not be referred to by NAME.

NAME must be one of a list of at least two names that may be used to specify an element. If the element may only be referred to by a single name, you'll need to use the add_XXX_alias function to add a new alias first, or the remove_XXX function to remove the element entirely.

If the alias is used as the name in any code set, one of the other names will be used instead. Predicting exactly which one will be used requires you to know the order in which the

standards were read, which is not reliable, so you may want to use the rename_XXX function to force one of the alternate names to be used.

The return value is 1 on success, 0 on an error.

MODULE::rename_XXX_code (CODE ,NEW_CODE [,CODESET])
These routines are used to change the official code for an element. At that point, the code returned by the XXX2code routine would be NEW_CODE instead of the code specified in the standard.

NEW_CODE may either be a code that is not in use, or it may be an alias for CODE (in which case, CODE becomes and alias and NEW_CODE becomes the ''real'' code).

The original code is kept as an alias, so that the code2XXX routines will work with either the code from the standard or the new code.

However, the all_XXX_codes routine will only return the codes which are considered ''real'' (which means that the list of codes will now contain NEW_CODE, but will not contain CODE).

MODULE::add_XXX_code_alias (CODE ,NEW_CODE [,CODESET])
These routines add an alias for the code. At that point, NEW_CODE and CODE will both work in the code2XXX routines. However, the XXX2code routines will still return the original code.

MODULE::delete_XXX_code_alias (CODE [,CODESET])
These routines delete an alias for the code.

These will only work if CODE is actually an alias. If it is the ''real'' code, it will not be deleted. You will need to use the rename_XXX_code function to switch the real code with one of the aliases, and then delete the alias.

KNOWN BUGS AND LIMITATIONS

Relationship between code sets

Because each code set uses a slightly different list of elements, and they are not necessarily one-to-one, there may be some confusion about the relationship between codes from different code sets.

For example, ISO 3166 assigns one code to the country ''United States Minor Outlying Islands'', but the IANA codes give different codes to different islands (Baker Island, Howland Island, etc.).

This may cause some confusion... I've done the best that I could do to minimize it.

Non-ASCII characters not supported

Currently all names must be all ASCII. I plan on relaxing that limitation in the future.

SEE ALSO

Locale::Codes

AUTHOR

See Locale::Codes for full author history.

Currently maintained by Sullivan Beck (sbeck@cpan.org).

COPYRIGHT