NAME

Encode::Alias - alias definitions to encodings

SYNOPSIS

```
use Encode;
use Encode::Alias;
define_alias( "newName" => ENCODING);
define_alias( qr/.../ => ENCODING);
define_alias( sub { return ENCODING if ...; } );
```

DESCRIPTION

Allows newName to be used as an alias for ENCODING. ENCODING may be either the name of an encoding or an encoding object (as described in *Encode*).

Currently the first argument to define_alias() can be specified in the following ways:

As a simple string.

As a qr// compiled regular expression, e.g.:

```
define_alias( qr/^iso8859-(\d+)$/i => '"iso-8859-$1"' );
```

In this case, if *ENCODING* is not a reference, it is `eval`-ed in order to allow `$1` etc. to be substituted. The example is one way to alias names as used in X11 fonts to the MIME names for the iso-8859-* family. Note the double quotes inside the single quotes.

(or, you don't have to do this yourself because this example is predefined)

If you are using a regex here, you have to use the quotes as shown or it won't work. Also note that regex handling is tricky even for the experienced. Use this feature with caution.

As a code reference, e.g.:

```
  define_alias( sub {shift =~ /^iso8859-(\d+)$/i ? "iso-8859-$1" :
undef } );
```

The same effect as the example above in a different way. The coderef takes the alias name as an argument and returns a canonical name on success or undef if not. Note the second argument is ignored if provided. Use this with even more caution than the regex version.

Changes in code reference aliasing

As of Encode 1.87, the older form

```
  define_alias( sub { return  /^iso8859-(\d+)$/i ? "iso-8859-$1" : undef }
);
```

no longer works.

Encode up to 1.86 internally used "local $_" to implement this older form. But consider the code below;

```
  use Encode;
  $_ = "eeeee" ;
  while (/(e)/g) {
    my $utf = decode('aliased-encoding-name', $1);
    print "position:",pos,"\n";
  }
```

Prior to Encode 1.86 this fails because of "local $_".

Alias overloading

You can override predefined aliases by simply applying define_alias(). The new alias is always evaluated first, and when necessary, define_alias() flushes the internal cache to make the new definition available.

```
# redirect SHIFT_JIS to MS/IBM Code Page 932, which is a
# superset of SHIFT_JIS

define_alias( qr/shift.*jis$/i  => '"cp932"' );
define_alias( qr/sjis$/i        => '"cp932"' );
```

If you want to zap all predefined aliases, you can use

```
Encode::Alias->undef_aliases;
```

to do so. And

```
Encode::Alias->init_aliases;
```

gets the factory settings back.

Note that define_alias() will not be able to override the canonical name of encodings. Encodings are first looked up by canonical name before potential aliases are tried.

SEE ALSO

Encode, *Encode::Supported*