Using Different Character Sets

<tip c n>**Note:** We recommend using **UTF-8** charset if that is at all possible for the language you intend to use on your site. This will ensure your site has as few charset related problems as possible going forward.</tip>

To use and have **non-english** character sets (Charset) within our software may need a few different changes, each listed on this page. Our auction and classified software comes to you using **UTF-8** character encoding by default[1)]. This character encoding is fine for the vast majority of sites, even the majority of non-English language sites. Using UTF-8 is even more important if your site uses multiple languages, since UTF-8 can be used across almost all the different languages.

<tip c n>**Note:** The search feature in general is not case-sensitive within the software but that does not apply to special and/or accented characters such as ç and Ç. This is a limitation of the system at the database level because of the way the accented characters are represented in the database. They cannot be searched in a case-insensitive manner.</tip>

Charset in config.php file

In **config.php**, look for a line similar to the one below[2)]:

```
define ('CHARSET_CLEAN','ISO-8859-1');
```

If you do not see that line, then you may need to add it. This setting controls **filtering** of user-input from PHP into the auction and classified software for use. It is also used for what is known as AJAX, when content on the page is updated without re-loading the entire page, the updated content will use the above setting for the charset. To change the setting, you would change **ISO-8859-1** to whatever charset you want to use, but it **must be a compatible charset** listed on this page.

For most sites and languages, the setting above is the only one of the charset settings that needs to be changed in the **config.php** file. If the charset you need to use is **not** one of the compatible charsets, you may be one of the few that need to make changes to other charset related settings in the config.php file. In the **config.php** file itself, it gives a thorough explanation of the different charset settings used for filtering user input, and how each setting is used, and what it should be set to. For your reference, below is the entire section pertaining to charsets, as it appears in the default **config.php** file distributed with the latest version, refer to it for an explanation if you do need to use a charset that is not in the compatible charsets list:

config.php charset section

```
/*
  ----CHARSET Settings----

 The settings below are used for various operations that are charset
sensitive,
 for instance cleaning "user input".  The settings with # in front will
need
```

```
 to be un-commented (remove the #) to use.

 For "input cleaning", and anywhere else the PHP function
htmlspecialchars()
 would normally be used, there is a 3 step process (below) to ensure
that the
 data is not corrupted due to differences in charsets.  Note that step
 1 and 3 are skipped if the appropriate settings are not specified
(Most sites
 will only need to set CHARSET_CLEAN, step 2):

 1. (Optional step, only run if CHARSET_FROM is set): The input's
charset is
    converted from the CHARSET_FROM setting to the CHARSET_CLEAN
setting.  It
    is converted either using mb_convert_string() or iconv(), according
    to CLEAN_METHOD setting.

    See http://www.php.net/mb_convert_encoding for more information on
setting
    CLEAN_METHOD to mb_convert_encoding.  CHARSET_FROM is used as the
3rd var passed
    to that function.  If CLEAN_METHOD is not set, and the function
exists,
    mb_convert_encoding is the default method used to convert the
charset.

    See http://www.php.net/iconv for more information on setting
CLEAN_METHOD
    to iconv.  CHARSET_FROM is used as the 1st var passed to that
function.

    This step, and optionally step 3, are necessary in order to be able
to
    clean any charset that is not compatible with the function
    htmlspecialchars() (see step 2)

 2. (Always run): The input is "cleaned" using the PHP function
htmlspecialchars()
    This step will use the CHARSET_CLEAN setting for the charset, that
charset must
    be compatible with htmlspecialchars().

    This step is always run for security reasons, to prevent a certain
type of
    hacking called "Cross Site Scripting" or XSS attack.  If the
charset is not
    specified, or is not a compatible charset, the default of
ISO-8859-1 is used.
```

```
    See http://www.php.net/htmlspecialchars for a list of compatible
charsets you can
    use.

 3. (Optional step, only run if CHARSET_TO is set):  The cleaned
input's charset
    is converted from the CHARSET_CLEAN setting to the CHARSET_TO
setting. It
    is converted either using mb_convert_string() or iconv(), according
    to CLEAN_METHOD setting.

    See http://www.php.net/mb_convert_encoding for more information on
setting
    CLEAN_METHOD to mb_convert_encoding.  CHARSET_TO is used as the 2nd
var passed
    to that function, at this step.  If CLEAN_METHOD is not set, and
the function exists,
    mb_convert_encoding is the default method used to convert the
charset.

    See http://www.php.net/iconv for more information on setting
CLEAN_METHOD
    to iconv.  CHARSET_TO is used as the 2nd var passed to that
function during
    this step.
*/

define('CHARSET_CLEAN', 'UTF-8');          //Required, see notes above
(step 2)

#define('CHARSET_FROM', 'UTF-8');                //optional, un-
comment and modify 'UTF-8' as needed
                                          //to use.  See notes above
(step 1)

#define('CHARSET_TO','UTF-8');                   //optional, un-
comment and modify 'UTF-8' as needed
                                          //to use.  See notes above
(step 3)

#define('CLEAN_METHOD', 'mb_convert_string'); //optional, un-comment to
use mb_convert_string()
                                          //in steps 1 and 3 above,
or un-comment and change
                                             //the
'mb_convert_string' to 'iconv' to use iconv()
                                             //instead.  Valid
settings are 'mb_convert_string'
                                             //and 'iconv'.  See
notes above (steps 1 and 3)
```

<tip c w>**Warning:** In the **advanced database settings** section in the config.php file, there is another charset setting that will look similar to the line below:

```
#$force_db_connection_charset = 'charset_name';
```

Most sites **should not use or change this setting**, it is used in a work-around for a very rare server configuration issue. If you use this setting when not necessary, or use it incorrectly, it can **corrupt the database data** when the data is being inserted into the database by the software. Do not change this setting unless you are absolutely certain that it needs to be changed, or if **Geo support has instructed you to change it**. If in doubt, do not touch the setting.</tip>

Charset in Admin Panel

You will need to change the character encoding set within the admin tool to set the character encoding used in the admin tool here:

Site Setup > General Settings > Character Encoding

If you do not have this setting we recommend updating to the latest version which will have it. Changing the setting here will allow you to insert characters directly into the browser when making changes in your admin panel, and those characters displayed properly without translation. This will affect all text controlled by the system and administered through the admin tool like template editing, language text, etc.

Charset in Template(s)

Within the templates used for your site, you may need to change the charset in the actual HTML code for your templates. Depending on your design, this may be a change in a single file, or it may be a change in multiple template files. If you are using design based off of the default templates for 5.0, the place will be **head.tpl** in the **main_page** templates within your template set. When editing the templates, make sure you are using the **<..> Source Code Editor** tab so that you can see the HTML code, since the alterations needed are to the actual HTML code and are not viewable or changeable using the WYSIWYG editor.

The main change will be in the "XML Tag" in your template(s), it will be located at the top of any templates that display the HTML head section. It will look similar to the below:

```
<?xml version="1.0" encoding="utf-8"?>
```

You would change **utf-8** to the charset needed for your language, if not using UTF-8. The majority of web browsers understand and interpret the charset specified in the XML declaration listed above to determine the Charset of the page, however older versions of Internet Explorer do not recognize the charset specified in the XML tag properly. To address this, you must also have the charset specified in HTML META tag in your templates, which does work for older versions of Internet Explorer. The best

place for this tag, if it is not already found in your template(s), is directly after the **<head>** in your template(s). The HTML META tag will look similar to below:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

You would change **utf-8** to the charset needed for your language, if not using UTF-8.

1)

This includes the charset used in the admin panel, as well as the charset used in the default templates. This does **not** include the charset used in the **config.php** for "filtering inputs" and AJAX contents, see Charset in config.php file section below for more information on that.

2)

Note that the "default" charset used for the config.php setting is ISO-8859-1, which is different than what is used for the default templates and in the admin panel by default. This was done for better backward compatibility, however we highly recommend changing it to UTF-8 if that is what your site uses in the templates and other locations. In future versions, starting with 5.1.0 when it is released, the default charset for this setting will be UTF-8.