7 Internationalization and Localization

Overview

OpenClinica supports internationalization and localization. This allows users and developers to adapt the software for a specific region or language by adding locale-specific components and translating text. A significant number of changes related to internationalization and localization have been implemented in OpenClinica 3.1.3. This chapter applies to OpenClinica 3.1.3 and later versions.

By default, OpenClinica is configured with English language support files in CATALINA_HOME/webapps/OpenClinica/WEB-INF/classes/org/akaza/openclinica/i18n. Other languages/locales may be added as described in this guide.

This page is not approved for publication.

7.1 Configure OpenClinica for Internationalization

Configure the Apache Tomcat Connector

Please change the default connector configuration in tomcat/conf/server.xml to ensure proper processing of UTF-8 characters.

The default setting is as follows:

```
<Connector connectionTimeout="20000" port="8080" protocol="HTTP/1.1"
redirectPort="8443"/>
```

Change this to the following:

```
<Connector connectionTimeout="20000" port="8080" protocol="HTTP/1.1"
redirectPort="8443" URIEncoding="UTF-8"/>
```

Without this modification, UTF-8 characters will not be processed correctly by tables filters and the Calendar Widget.

Note that, if you are using SSL, you must apply this modification to the line that starts with:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"...
```

Ensure Operating System Requirements are Met

OpenClinica has been tested on operating systems configured for the English language only. Please install OpenClinica on a supported OS that is configured for the English language. This will help ensure proper behavior of date formats and locale-specific text, as described in the guide below.

This page is not approved for publication.

7.2 Install Desired Translations

Existing Translations

The OpenClinica community provides translations for multiple locales. Each translation is comprised of a series of properties files with the translated text for the language in question, as well as other localized properties like date formats. Existing translations are accessible at https://www.transifex.com/projects/p/openclinica/.

Only English properties files are installed in OpenClinica by default. The additional translations have been contributed by community members. These translations are not all fully tested and may not have all the strings translated. To determine if the localized files you are interested in are complete, please compare them to the files in CATALINA_HOME/webapps/OpenClinica/Web-INF/classes/org/akaza/openclinica/i18n.

OpenClinica currently has 9 properties files:

- admin.properties
- audit_events.properties
- exceptions.properties
- format.properties
- notes.properties
- page_messages.properties
- terms.properties
- words.properties
- workflow.properties

Starting with version 3.1.3, there is an additional properties file, licensing.properties file. Please do not translate this file.

First, you need to duplicate each of 9 properties files and renaming them by appending the locale string. For example, in case of French, this is how the file list should look like:

- admin.properties
- admin_fr.properties
- audit_events.properties
- audit_events_fr.properties
- exceptions.properties
- exceptions_fr.properties
- format.properties

- format_fr.properties
- notes.properties
- notes_fr.properties
- page_messages.properties
- page_messages_fr.properties
- terms.properties
- terms_fr.properties
- words.properties
- words_fr.properties
- workflow.properties
- workflow_fr.properties
- licensing.properties (3.1.3 or above - do not translate this file)

Pre-3.1.3, please do not translate License infomation under 'footer.license' in words.properties file where it notes "# DO NOT CHANGE ANY OF THE TEXT BELOW".

Installing a Translation

Once you are satisfied with your translation, place the localized properties files in your i18n folder (CATALINA_HOME/webapps/OpenClinica/Web-INF/classes/org/akaza/openclinica/i18n) and restart Tomcat. When a user changes the preferred language in the user's web browser to match the locale in the format_xx.properties file, the user will see the translation into that language.

Please note: even though you have no intention of using the original English strings and wish to use only translated GUI, OpenClinica may break if you modify the original properties files. Instead, you need to create localized versions with '_xx' i18n marker as described above.

Which Language will be displayed in the User Interface (UI)?

OpenClinica determines the locale/language to be displayed based on two conditions: user language preferences and the availability of nine OpenClinica properties files in the desired language.

English (en) language is the default language.

The user's browser language setting determines the language priority for OpenClinica. For locale lookup, OpenClinica adhere to W3C rules which are based on RFC 4647. And also this locale must have all of nine OpenClinica properties files available in the i18n folder.

Browser Language Preferences	Installed Localized Properties files	Displayed Language
1. English [en-us] 2. French [fr] 3. German [de] 4. English [en]	English (default) _fr.properties (French) _en-gb.properties (United Kingdom)	English [en] Even though 'en-us' is not installed, you "progressively truncate" the user's top priority language range until you get a match, which would be en = en.
1. German [de] 2. French [fr] 3. English [en]	English (default) _fr.properties (French) _en-gb.properties (United Kingdom)	French [fr] German [de] is not installed so French [fr] is taking effect that is the 2nd on the list.

1. Chinese [zh]	English (default) _fr.properties (French) _en-gb.properties (United Kingdom)	English [en] Chinese [zh] is not installed so the default, English is taking effect.
1. United Kingdom [en-gb]	English (default) _fr.properties (French) _en-gb.properties (United Kingdom)	United Kingdom [en-gb]

Approved for publication by Cal Collins. Signed on 2014-03-26 4:31PM

Not valid unless obtained from the OpenClinica document management system on the day of use.

7.3 Translate OpenClinica into New Languages or Maintain Existing Translations

Completing Translations

The default (English) files in your i18n folder should be considered a complete list of all the strings that should be translated. You can find existing translated files at the Transifex OpenClinica Project site. The Transifex site includes tools to allow you to manage and conduct translations. Sign up to be a translator for your target language. Once approved you can upload translations or build them online using the Transifex tools.

Visit the i18n forum to share information on what you're translating and ask others for help.

If you find your target language does not match with your OpenClinica version, run a diff to identify the changes so you do not need to start from scratch. Here are some free diff tools. Diff files for 3.1.3 are located here.

In order to complete a translation, you must translate the text to the right of the '=' sign into your language.

```
Default   CRF_attributes = CRF Attributes
es        CRF_attributes = Atributos del CRD
```

Then save the file with your language extension. For example, if you are translating the strings into Spanish, you would add _es to the names of each of the 9 default files and save them as additional files. For example, words.properties would become words_es.properties. Please make sure to keep the original default files in the i18n folder along with the translated files.

Do not change the contents of the default files. Note also there are some strings in the file which should never be translated. These are identified with an appropriate comment in each of the properties files.

Asian Languages

Asian language translations are managed through GitHub rather than Transifex. Transifex does not seem to handle Asian language well. For example, you won't be able to see human readable characters, and downloaded files will add an unnecessary back slash because it treats file encoding as Latin-1 while it should be UTF-8.

To contribute/update Asian language translations, please submit a pull request to the GitHub repository.

Character Encoding

If you are translating into a language that uses characters not found in Western Europe (specifically, outside the ISO-Latin-1 character set) you will have to save those characters as Unicode escape sequences. For example "" would be "u20AC". To convert your translated text to Unicode escape sequences (Hex value), enter them into a converter such as brahan_converter and copy the strings. Be sure to save your original unconverted text in a comment, because the escape sequences are not editable.

Contributing a Translation

If you complete or improve one of the existing translations or create a new one, please consider sharing this back with the community! It's easy, sign up the Transifex OpenClinica Project site and post your files there.

Localization of Date Formats and Calendar Widget

OpenClinica supports displaying date and time in localized formats via the date format settings in the format.properties file. By default dates are formatted in DD-MMM-YYYY in English, where 'MMM' is an alphabetical abbreviation for the month.

Other date formats may be used by changing the locale-specific date format, with the alphabetical abbreviation or using numeric formats. For instance, you may want to use DD/MM/YYYY in your format_en-gb.properties for the Great Britain locale (note: this particular format is provided as an example only and is NOT recommend due to its ambiguity with common American date formats).

OpenClinica dates are never localized in ODM XML files, web services calls, or other programmatic data formats. In these cases OpenClinica uses the universal ISO 8601 format to represent all dates, date/times, and partial dates. Regardless of the user's locale, OpenClinica also saves all dates in its database in the ISO 8601 format. See OpenClinica Date Format Specifications for more detail.

When localizing OpenClinica, you must ensure that date formats specified in the format.properties file for a given locale are also supported by the OpenClinica javascript calendar widget. This is especially important when using the 'MMM' alphabetical month name as part of the localized date format. Please refer to examples below.

By default, OpenClinica provides a calendar widget in English (en). A file named calendar-en.js is located at ${Catalina.home}webappsyour-openclinica-instanceincludesnew_callang.

OpenClinica supports localization of the calendar widget into other languages. Let's use French (fr) as example. We'll assume French properties files are installed with format_fr.properties having a

date format of DD-MMM-YYYY.

In this French scenario, if French is the chosen web browser locale, but there is no calendar-fr.js available, the default English calendar widget will be used. Since server-side date validation expects a French-formatted date, the English date string provided by the calendar widget may fail server-side date format validation. The user would be required to manually change the English date to a French date. For example, the English calendar widget will populate the field with 01-May-2012 but the DD-MMM-YYYY expected for the French locale would be 01-Mai-2012. 01-May-2012 has to be changed to 01-Mai-2012 to pass the validation.

To ensure the calendar widget is compatible with the localized date format, create a calendar-fr.js file which contains the translated contents from calendar-en.js file (follow the instructions in the .js file). Please refer to the examples below for more detail. Then, place calendar-fr.js file into ${Catalina.home}webappsyour-openclinica-instanceincludesnew_callang and restart Tomcat.

Based on the locale 'fr' being used, server-side validation of month names/abbreviations are automatically expected in French by the OpenClinica. The calendar widget translation MUST use the same month names/abbreviations as those expected by Java. Java 6 currently supports this list of languages for the java.text package, and the translations are provided by projects like CLDR from the Unicode Consortium. To ensure your month names/abbreviations exactly match those expected by the Java application, you can download the CLDR translations from here and find the list of months for your language in core/common/main.xml. Please note that Asian languages do not seem to follow this practice. For example, the browser default for the MMM abbreviation for Japanese and Korean are a number without leading zero when less than the month of October. This indicates both languages do not use strings for an abbreviated month.

Examples of Localized Date Formats

To ensure localized date formats work properly, you will need to modify the following items in the format.properties file located under tomcat/webapps/OpenClinica/WEB-INF/classes/org/akaza/openclinica/:

- date_format =
  - is used to build the error message
- date_format_string =
  - is used to display format by UI level
- date_time_format_string =
  - is used to format where time is applicable such as Scheduling Event
  - Note that OpenClinica allows only 24 hour format, and AM/PM markers will not be used
- date_format_calendar=
  - is used to format the date string that is sent from calendar widget when applicable

date_format, date_format_string, date_time_format_string

- Date
  - dd: 2 digits, i.e., 01, 02, 03...
  - d: digit(s) without leading zero, i.e., 1, 2, 3... 10, 11...
- Month
  - MM: month in digits. i.e., 01, 02, 03...
  - MMM: locale default abbreviated month names
    - Mar (English)
    - mars (French)

        - Mr (Deutsch)
        - 3 (Japanese, Korean)
        - ??(Chinese)
        - .
        - .
    - MMMMM: locale default full month names
        - December (English)
        - dcembre (French)
        - .
        - .
- Year
    - yyyy: Year with century
    - yy: Year without century

date_format Examples

Upper case M is used for month, while lower case is used for day and year for java class. However, upper case for all three are used to build error messages.

US English (Default): 01-Jan-2012

- date_format = DD-MMM-YYYY
- date_format_string = dd-MMM-yyyy
- date_time_format_string = dd-MMM-yyyy HH:mm:ss

Great Britain English and Spanish: 30/10/2012

- date_format = DD/MM/YYYY
- date_format_string = dd/MM/yyyy
- date_time_format_string = dd/MM/yyyy HH:mm:ss

French: 01-fvr.-2012

- date_format = DD-MMM-YYYY
- date_format_string = dd-MMM-yyyy
- date_time_format_string = dd-MMM-yyyy HH:mm:ss

Deutsch: 20.10.2012

- date_format = MM.DD.YYYY
- date_format_string = dd.MM.yyyy
- date_time_format_string = dd.MM.yyyy HH:mm:ss

Japanese: 2012/1/1

- date_format = YYYY/MMM/D
- date_format_string = yyyy/MMM/d
- date_time_format_string = yyyy/MMM/d HH:mm:ss

Japanese and Chinese: 2012?1?1? (Unicode Hex value needs to be escaped with single quote)

- date_format = YYYY'u5e74'MMM'u6708'd'u65e5'
- date_format_string = yyyy'u5e74'MMM'u6708'd'u65e5'

- date_time_format_string = yyyy'u5e74'MMM'u6708'd'u65e5' HH:mm:ss

Chinese: 2012-01-01

- date_format = YYYY-MM-DD
- date_format_string = yyyy-MM-dd
- date_time_format_string = yyyy-MM-dd HH:mm:ss

Chinese: 2012-??-1

- date_format = YYYY-MMM-D
- date_format_string = yyyy-MMM-d
- date_time_format_string = yyyy-MMM-d HH:mm:ss

Korean: 2012.1.1

- date_format = YYYY.MMM.D
- date_format_string = yyyy.MMM.d
- date_time_format_string = yyyy.MMM.d HH:mm:ss

Korean: 2012? 1? 1? (Unicode Hex value needs to be escaped with single quote)

- date_format = YYYY'uB144'MMM'uC6D4'd'uC77C'
- date_format_string = yyyy'uB144'MMM'uC6D4'd'uC77C'
- date_time_format_string = yyyy'uB144'MMM'uC6D4'd'uC77C' HH:mm:ss

CJK: 2012.01.01

- date_format = YYYY.MM.DD
- date_format_string = yyyy.MM.dd
- date_time_format_string = yyyy.MM.dd HH:mm:ss

date_format_calendar

List of parameters:

- %b: Prints the abbreviated **month name**. Range: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec. Example Output: Jan
- %B: Prints the full month name. Range: January, February, March, April, May, June, July, August, September, October, November, December. Example Output: January
- %d: Prints the **day** of the month with the leading zero. Range: 01 to 31. Example Output: 01
- %e: Prints the **day** of the month without the leading zero: Range: 1 to 31. Example Output: 1
- %m: Prints the numeric **month** with the leading zero. Range: 01 to 12. Example Output: 01
- %o: Prints the numeric representation of the **month** without the leading zero. Range: 1 - 12. Example Output: 1
  (Not supported OpenClinica version 3.1.2 or earlier)
- %y: Prints the **year** without the century with the leading zero. Range: 00 - 99. Example: 01
- %Y: Prints the **year** with the century. Example: 2001

date_format_calendar Examples

US English (Default): 01-Jan-2012

- date_format_ = %d-%b-%Y

Japanese and Chinese: 2012?1?1? (Hex escape not needed)

- date_format_ = %Yu5e74%ou6708%eu65e5
  (%o not supported OpenClinica 3.1.2 or earlier)

CJK: 2012.01.01

- date_format_ = %Y.%m-%d

Calendar Widgets

- When you translate Calendar Widgets found at ${Catalina.home}webappsyour-openclinica-instanceincludesnew_callang, you do not need to convert to Unicode escape sequence (Hex value). Translated strings pass without corruption as long as your tomcat connector is configured properly as explained before.

  Example in French:

  // short month names
  Calendar._SMN = new Array
  ("janv.",
  "fvr.",
  "mars",
  "avr.",
  "mai",
  "juin",
  "juil.",
  "aot",
  "sept.",
  "oct.",
  "nov.",
  "dc.");
- Save As the calendar-en.js file with appropriate locale marker, i.e., calendar-fr.js. This file name has to be called from the format-{lang}.properties file under "jscalendar_language_file =".

Localization of Image files

As of OpenClinica 3.1, icons and other image files do not contain any text and do not have to be localized. However it is possible to create localized images in the desired language. Under the folder ${Catalina.home}webappsyour-openclinica-instanceimages, makes a new folder with the name as desired locale code, e.g. "es" for Spanish (es). In this new folder collects all translated image files.

Notes for Developers

1. When you need Locale, please call the method org.akaza.openclinica.i18n.core.LocaleResolver.getLocale (HttpServletRequest request).
org.akaza.openclinica.i18n.core.LocaleResolver class locates in Core module. Its resolveLocale(HttpServletRequest request) method implements OpenClinica business logic of how to determine locale (please refer to Which Language to Be Displayed on User Interface (UI)).

2. If you need Date/Time format in UI, please call the methods for pattern string or SimpleDateFormat in the class org.akaza.openclinica.i18n.util.I18nFormatUtil.

If you need Date/Time format in database, please call methods for date format pattern string in the class org.akaza.openclinica.bean.core.ApplicationConstants.

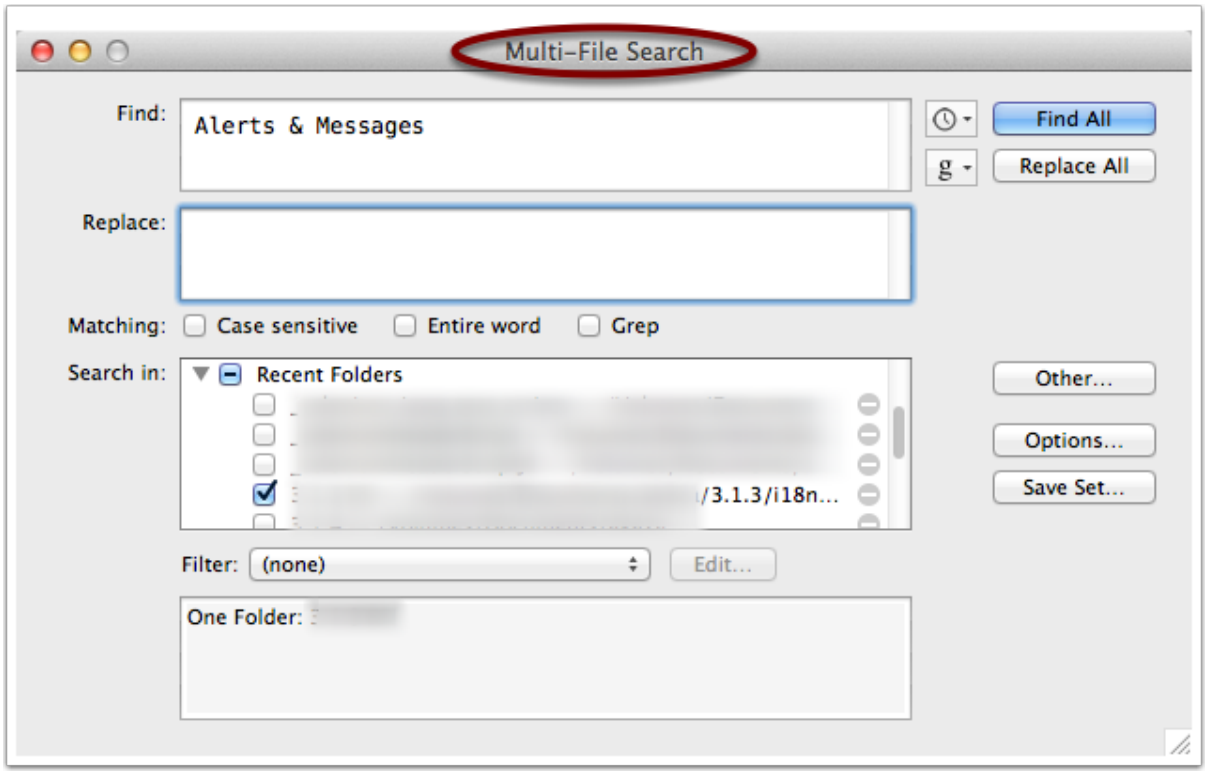
Approved for publication by Cal Collins. Signed on 2015-02-06 8:42AM

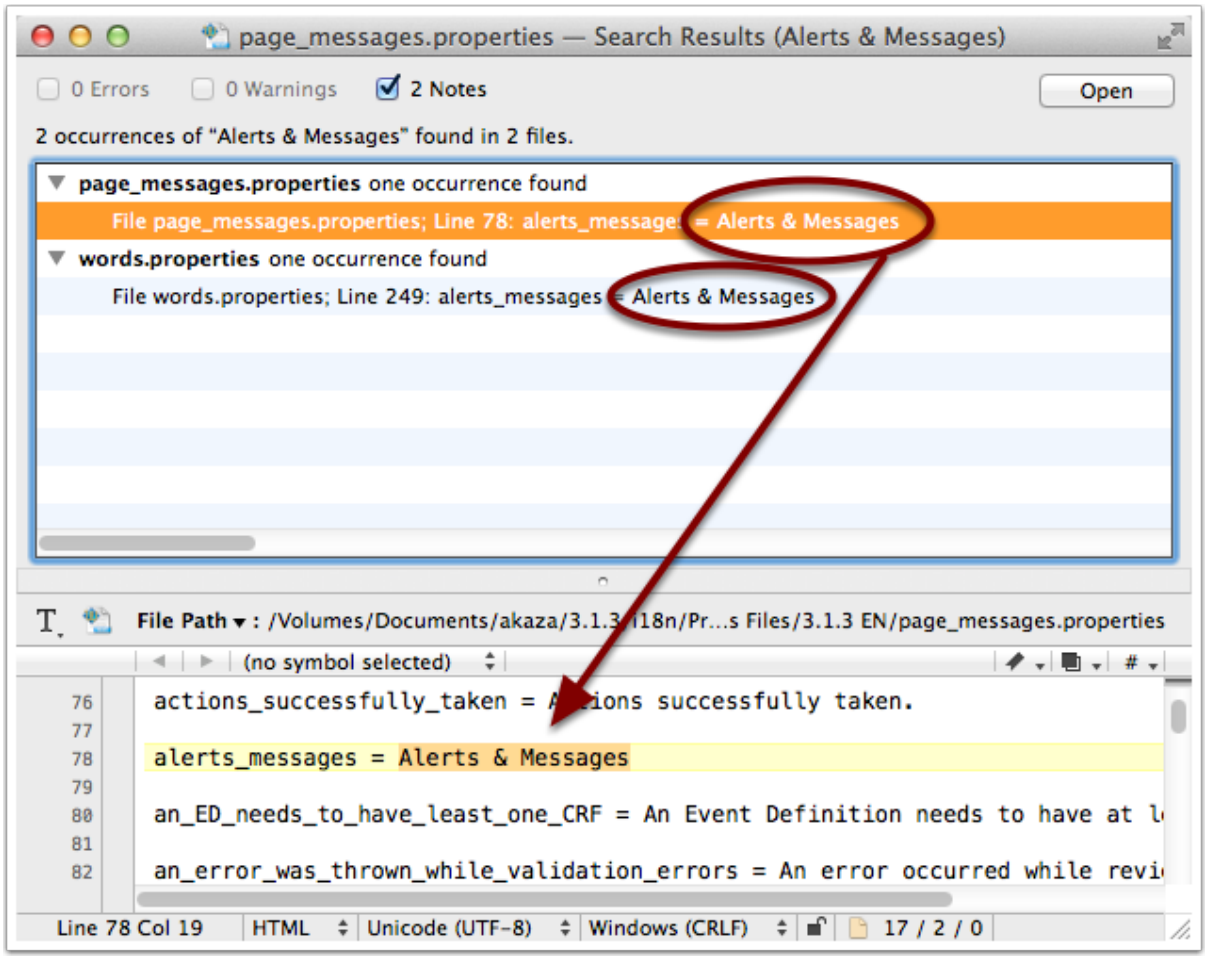
Not valid unless obtained from the OpenClinica document management system on the day of use.

7.4 Recommended steps to translate strings

OpenClinica properties files may contain some legacy strings which should have been removed. For this reason, translating page by page on the front end, instead of translating string by string in the properties files, is recommended.

1. Create a new local directory for your work space
2. Copy all the original properties files, except licensing.properties, into the new directory
3. Rename each of the 9 files so the language extension is appended as explained in section 11.2, e.g., words.properties ? words_ja.properties (make sure you are not modifying the original files)
4. Open the page you wish to start translating, e.g., /OpenClinica/MainMenu
5. Chose a word and/or sentence you wish to translate, e.g., "Alerts & Messages"
6. Search the string(s) chosen in Step 2 throughout the work directory you created in the Step 1
    - Some text editors such as TextWrangler (Mac)/NotePad++ (Wind) makes this step easy





Once you found the string you wish to translate

1. Translate the word(s)

   alerts_messages = ????????

2. Duplicate the line

   alerts_messages = ????????

   alerts_messages = ????????

3. Comment out the first line

   #alerts_messages = ????????

   alerts_messages = ????????

4. Now convert the translated string into Unicode value using a tool such as this.

   #alerts_messages = ????????

alerts_messages = u8b66u544au3068u30e1u30c3u30bbu30fcu30b8

It's important to leave the readable string in the commented line, in case you need to modify/update the string later.

This page is not approved for publication.

7.5 OpenClinica Data Extract File Format

OpenClinica Data Extract File Format

When data contain non-ASCII characters, you may encounter character viewing issues on extracted files. Here is the rundown:

- CDISC ODM XML 1.3 Full with OpenClinica extensions
  - Converts into Decimal values with Character Entity marker (&#), i.e., １２３４５
- CDISC ODM XML 1.3 Clinical Data with OpenClinica extensions
  - Converts into Decimal values with Character Entity marker
- CDISC ODM XML 1.3 Clinical Data
  - Displays as expected - see note below if Windows
- CDISC ODM XML 1.2 Clinical Data with OpenClinica extensions
  - Displays as expected - see note below if Windows
- CDISC ODM XML 1.2 Clinical Data
  - Displays as expected - see notes below if Windows
- View as HTML
  - Displays as expected - see notes below
- Excel Spreadsheet
  - Tab delimited text file. This can be displayed with workaround
- Tab-delimited Text
  - Displays as expected - see notes below if Windows
  - If opening with Microsoft Excel, see this workaround
- SPSS data and syntax
  - Tab delimited text file. This can be displayed - see notes below if Windows
- Datamart in a downloadable format
  - Currently not fully compatible - see notes below
- Datamart
  - Currently not fully compatible - see notes below
- Discrepancy Notes CSV Export
  - Converts into Hex values with Unicode Escape marker (u), i.e., uff11uff12uff13uff14uff15
- Discrepancy Notes PDF Export
  - Currently not compatible 17230

File Encoding Issue

If you encounter issues viewing UTF-8 characters where they are expected to display correctly, you may need to specify the encoding.

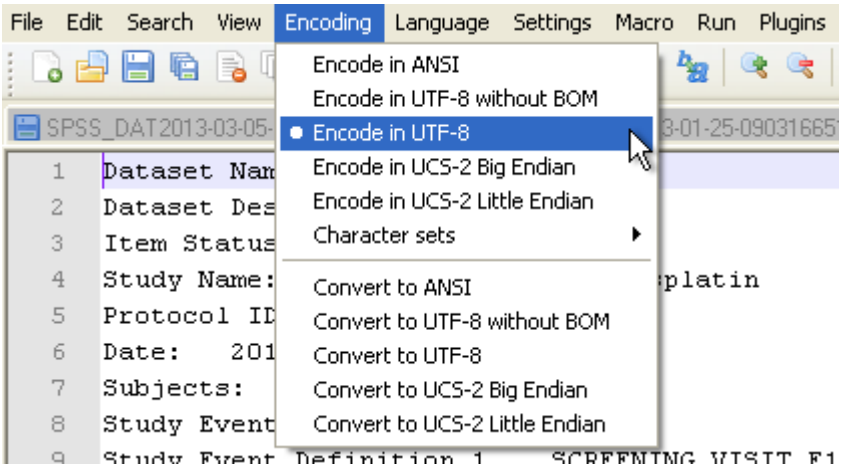
For example, if you open OpenClinica exported **HTML** file and see garbled corrupted characters, you need to set your browser encoding to UTF-8 to view those characters correctly.

Opening files on Windows machines

The native file encoding on Linux, Unix, Mac OS and popular databases such as SQL and Oracle is UTF-8 (Big Endian); however, Microsoft Windows' native file encoding is UTF-16LE (Little Endian). Depending on your text editor, this can become an issue because Java runs on OpenClinica server are UTF-8, not UTF-16LE.

If you open a file that contains non-ASCII characters and the file itself does not declare the encoding at the file binary header, the OS will try to determine with which encoding the file is written. Non-Windows OSes have an UTF-8 character map library in its OS level to determine the character map when opening the file, while Windows does not.

If you see garbled UTF-8 characters in ODM 1.3, ODM 1.2-Ext, ODM 1.2, Tab-delimited Text, and SPSS .dat files, you may need to **SaveAs** with the file encoding specified to UTF-8. Popular text editor such as Notepad++ (Win) and TextWrangler (Mac) will enforce encoding declaration at the file binary header level.



BOM Option

BOM (Byte Order Mark) can be critical on Windows environment. Unicode on Linux, Unix, Mac OS and popular databases such as SQL and Oracle is UTF-8, which is Big Endian byte order by default; Windows chooses UTF-16 Little Endian byte order.

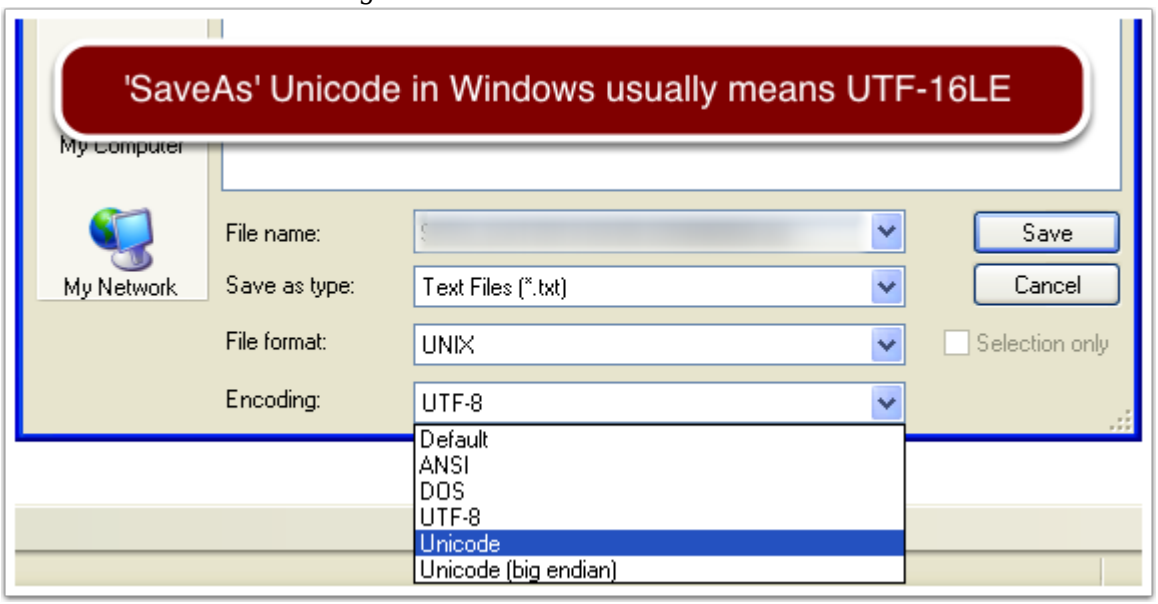
Your text editor should give you the option to SaveAs 'UTF-8 with BOM' and 'UTF-8 without BOM'. In our experience, this is somewhat hit or miss. Logically, it should work better with BOM but sometimes it seems to confuse Windows. You may need to experiment with the option of 'with' and 'without' BOM to find which option works on your Windows environment.

Excel issue

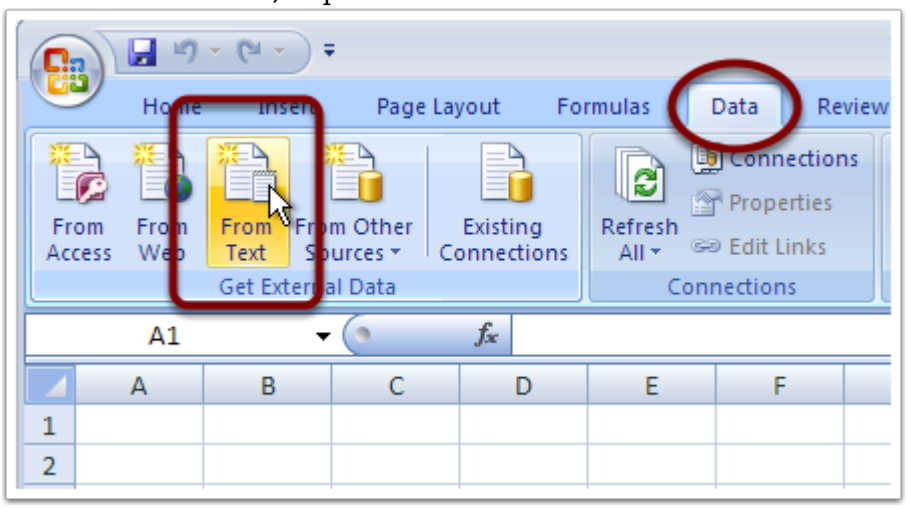
Not only does Excel not offer an encoding option, it doesn't seems to understand UTF-8 encoding. Even on a Mac OS platform, where UTF-8 is the native encoding, Excel cannot display non-ASCII characters unless file is encoded in UTF-16LE.

Workaround 1

1. Open the .xls file with a text editor of your choice
2. SaveAs UTF-16LE encoding

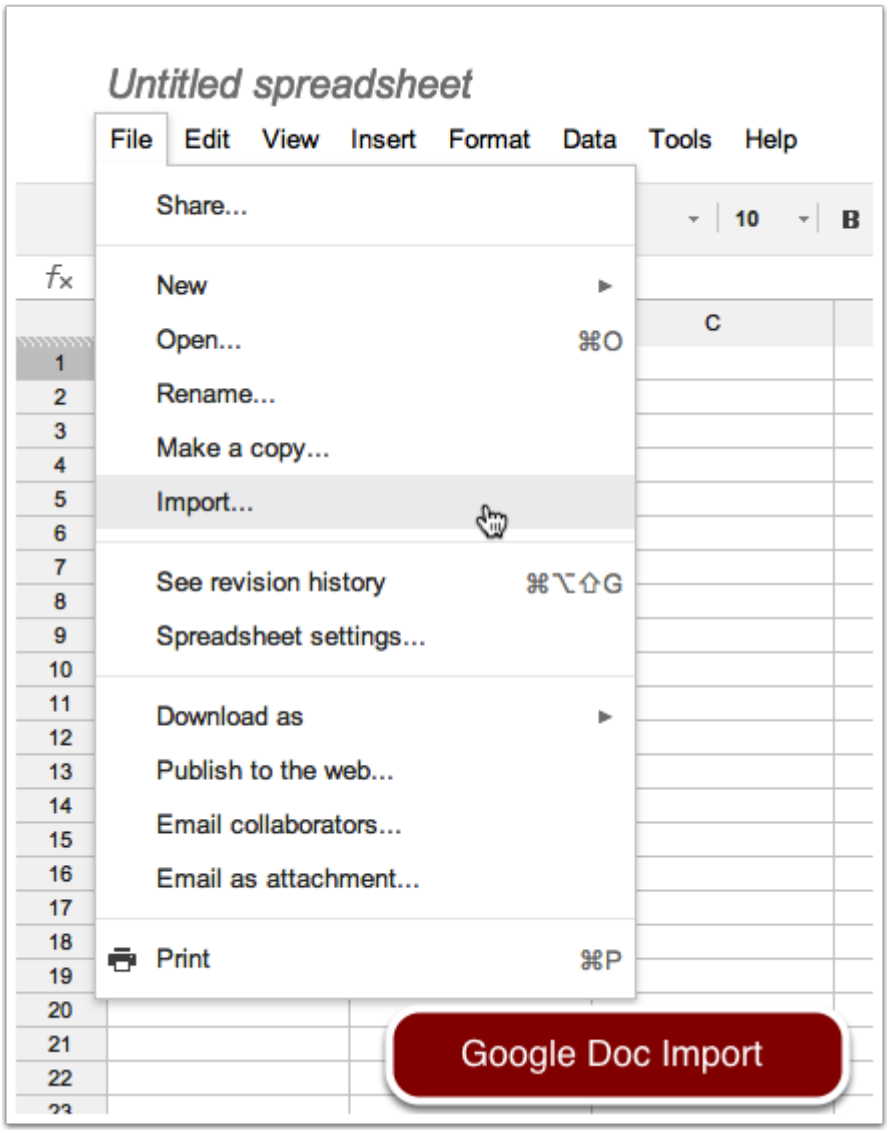


3. Open with Excel application
    - If .tsv instead of .xls, import the data into a new Excel file

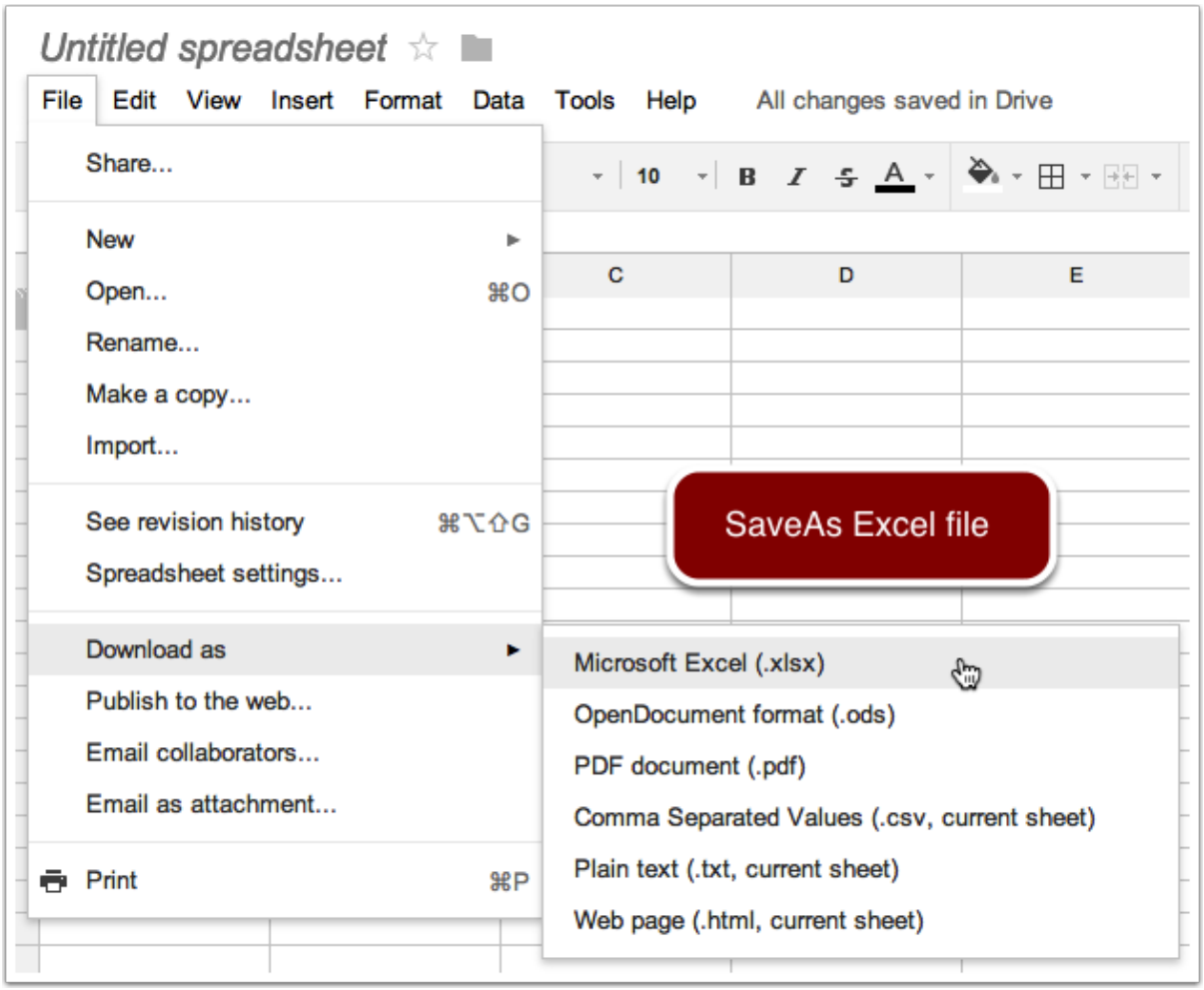


Workaround 2

1. Import to Google Doc Spreadsheet
    - Google Doc is native UTF-8 and correctly identifies UTF-8 encoded files



2. Select 'Download As' ? Microsoft Excel
    - Google Doc successfully embeds the encoding declaration binary header when the file is re-saved to your local directory



3. Open with Excel application

Troubleshooting

- If you see white boxes, e.g., ??????????
  - This is an indication of font problem. Your OS may not have Unicode mapped fonts.
  - This is a typical issue with Windows XP
  - You need to obtain Unicode font and install on your Windows
    - Arial Unicode Font is Microsoft default Unicode font
- If you see one or more white boxes in a recognizable i18n string, e.g., ????
  - This usually means incomplete Unicode font is assigned, often seen when the properly encoded file is opened with Excel
  - Select All and reassign known working Unicode font

Data Mart Issues

Currently, OpenClinica Data Mart function converts non-ASCII characters used for Table Names and Column Names into underscore character to avoid possible database issues.

It was designed this way for occasional non-ASCII character appearances among ASCII characters in a string, such as European word with accented characters. It was never meant for 100% non-ASCII string such as Asian languages.

If 100% non-ASCII string, all the entries become a series of underscore characters, which ends up with duplicated Table/Column names. We are hoping to resolve this issue as soon as possible.

On the other hand, data will not be affected by this. You can have Unicode characters in data string, and Data Mart will work as expected.

In summary:

- Any string that becomes a Table Name needs to be ASCII such as CRF name and Item Group name.
- Any string that becomes a Column names needs to be ASCII such as Multi-select Response Text and Item Name.
- Study Name becomes a series of underscore characters if non-ASCII

If a series of underscore characters become duplicated entries, Data Mart in a Downloadable Format output file will error when importing to Postgres. On the other hand, Data Mart extract operation silently stops during the operation without error message, leaving the data output incomplete (17249).

Data Mart in a Downloadable Format on Windows

Even if your .sql output file from Data Mart in a Downloadable Format does not contain any offensive underscore characters, remember Windows may require you to modify the file encoding as discussed above. This is not an issue when Postgres/pgAdmin III is running on Mac OS and/or Linux OS.