

Casabac Unicode Support

Full Unicode support was added into the Casabac GUI Server with build 25_20040105. Before ISO 8859-1 was used for encoding and decoding HTML pages and your system's default encoding was used for storing characters in files (e.g. multi language files).

This document contains both background information and upgrade information.

The Decoding/Encoding Problem

Encoding/Decoding means the conversion from human-readable character data into bytes - and back. There is a certain history behind - and as consequence a variety of standards. And: there is one standard covering "all you need" - Unicode.

One Byte per Character Encoding

This is the very traditional way of translating characters into bytes. There is one byte per character. Famous standards are:

- ASCII
- EBCDIC

The standards were designed having the language "English" and having memory restrictions in mind. But of course there are limits: there are much more than 256 characters available throughout the world.

As consequence "enhancements" were made. Several character sets were defined, in which the number area 0-127 was shared with ASCII and the characters in area 127-255 were interpreted dependent on a character set definition. There are for example a couple of character sets "ISO-8859-*" which are standardized:

- ISO 8859-1 for Western-Europe languages
- ISO 8859-2 for Eastern-Europe languages
- ISO 8859-3 for Southern-Europe languages
- etc.

There are also standards defined by vendors as CP 1252 - the character used in Microsoft Windows based systems.

Well, now certain characters - as are contained in the Cyrillic alphabet - were made available. But: with every text file that is transferred from bytes into characters you first have to know the character set in order to properly decode the bytes.

The limitations are:

- A one-byte-encoding does not cover alphabets like the Chinese one with thousands of characters.
- The fact that a certain text only is understandable with knowing the right character set leads up into some mess - especially in an environment like the Internet in which "multi language" and "multi character set" pages are something very usual.
- Mixing of two character sets inside one document is a problem.

Unicode

Unicode is designed to solve all these problems. Each character is encoded into two bytes of data. As consequence the range of potentially available characters is much wider than with single byte encoding. Every single character that is available via Unicode has exactly one unique representation as double byte value.

Modern programming languages/ environments like Java are based on Unicode - i.e. each character internally is represented by two bytes.

Unicode is a "nearly perfect" solution from programming point of view because it is a clear and simple. But:

- The representation of one character as two bytes requires two times more resources than before. Knowing that most characters stored in computer systems are in the "old ASCII range" this means a waste of resources.
- All low level text editors are not usable anymore because they are typically based on ASCII encoding.
- And: though the available encoding area is expanded due to the usage of two bytes from 0 to 64k it is predictable that also this will not cover all characters existing throughout the world.

Unicode - UTF-8

UTF-8 is an extension of the Unicode standard: it defines a mapping between the Unicode encoding into a special encoding requiring a dynamic number of bytes:

- Characters in the area 0-127 are represented as one byte
- All other characters above are encoded into two or more bytes.

UTF-8 does not define a character-to-code-mapping but defines a mathematical rule how to convert data that is originally kept in two bytes into one, two or three bytes - and back.

UTF-8 has become the standard for text representation for all Unicode aware pages in the Internet. All up to date browsers support UTF-8 encoding. All up to date programming environments offer simple possibilities to convert character data into UTF-8 bytes.

Unicode UTF-8 in Java-/Web-Environment

Java and Unicode UTF-8

As mentioned in a previous chapter Java internally is Unicode enabled, i.e. every character stored in the data type "char" internally is represented as two bytes of data. Once you have a certain character data available inside your Java program you do not have to take care of encoding and decoding issues.

But: you have to take care when converting bytes of information into characters or if you convert characters into bytes of information. Typical example:

- When reading a text file from the file system you have to make sure that your program uses the right character set to convert the bytes into characters.
- The same is true for writing a text file.

The Java class library offers the classes "InputStreamReader" and "OutputStreamWriter" in order to read/write character based data into byte streams. In addition every class "String" supports a getBytes()-method. If using these classes in default mode - i.e. without explicitly specifying the character set to use - then the system's default character set is used. In Windows based systems this typically is "Cp1252" (Microsoft Codepage 1252).

The Web and Unicode UTF-8

The Web is a medium to display and input information that contains characters. There are the following aspects:

- In an http response (containing as body part the HTML page) there is header information about the character set used for encoding the contained HTML page. This information is the default information and typically is generated by the web server.

Example: when offering pages as part of a web application then you can define the header information via the mime-mapping part of the web.xml of your web-application:

```
<web-app>
  ...
  ...
  ...
  </servlet-mapping>

  <mime-mapping>
    <extension>html</extension>
    <mime-type>text/html; charset=UTF-8</mime-type>
  </mime-mapping>
</web-app>
```

- In an html page itself there is a header information declaring the encoding that is used for the page:

```
<html>
  <head>
    <meta http-equiv='Content-Type' content='text/html; charset=UTF-8'>
    <title></title>
  </head>
  ...
  ...
  ...
  ...
```

The browser receives both information and interprets the byte stream coming as response accordingly.

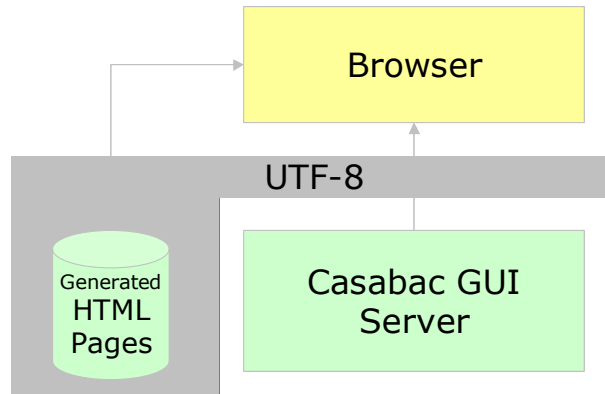
Unicode UTF-8 and Casabac

Casabac uses UTF-8 for Encoding HTML pages and Decoding Requests

Casabac is a GUI Server serving web browsers - and as consequence transfers character information from the browser to the server and back.

Casabac consistently uses UTF-8 encoding from build 25_20030105 on when talking to the web browser. This means:

- All HTML-pages (including the internally used data pages) are encoded in UTF-8 and as consequence can contain any characters available via Unicode.
- Requests coming from HTML pages are decoded using UTF-8 into Java characters.



As consequence Casabac is open to serve all languages supported by the Unicode character set.

Textual Information stored in Casabac

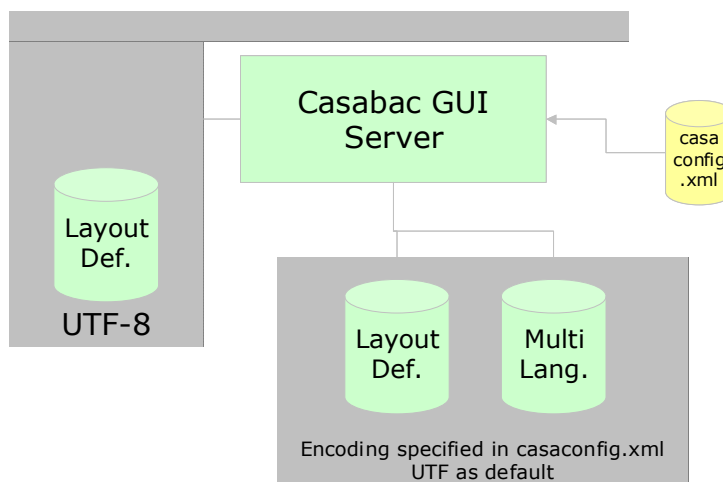
Casabac explicitly stores textual information in the following objects:

- Layout Definitions (e.g. you define the name of a label)
- Multi Language Files (e.g. you define the name of a label in multiple languages)

Since build 25_20040105 you can explicitly configure the character set that Casabac uses to store and read these files. In the file `casconfig.xml` there is a parameter **"textencoding"** that can be used accordingly.

```
<casconfig ...
    textencoding="UTF-8"
  ...>
</casconfig>
```

Casabac itself is delivered on "UTF-8" but can also be used (due to compatibility) with different character sets - e.g. if you application files are not UTF-8 based.



If not explicitly defined in `casconfig.xml` then "UTF-8" is used for encoding and decoding.

Compatibility with previous Releases

Previous releases used your server's default encoding for reading and writing textual information (layout definitions, multi language files).

When starting Casabac (>= build 25_20040105) and having used Casabac before then you will see inside the log the following information:

```

20040105/120552/653 (I) (CasabacGS) Connector:
=====
20040105/120552/653 (I) (CasabacGS) Connector: Configuration:
20040105/120552/653 (I) (CasabacGS) Connector: StartMonitoringThread = true
20040105/120552/653 (I) (CasabacGS) Connector: RequestClientHost      = false
20040105/120552/653 (I) (CasabacGS) Connector: casabac.home = ...
20040105/120552/653 (I) (CasabacGS) Connector: casabac.log = ...
20040105/120552/653 (I) (CasabacGS) Connector:
=====
20040105/120552/653 (I) (CasabacGS) Connector: webApp temporary directory = ...
20040105/120552/653 (I) (CasabacGS) Connector:
=====
20040105/120552/653 (I) (CasabacGS) Connector: System's default text encoding is Cp1252
20040105/120552/653 (I) (CasabacGS) Connector: text encoding used by casabac is UTF-8
20040105/120552/653 (I) (CasabacGS) Connector:
=====
20040105/120552/653 (I) (CasabacGS) Connector:
20040105/120552/653 (I) (CasabacGS) Connector: CCCC
20040105/120552/653 (I) (CasabacGS) Connector: C          b
20040105/120552/653 (I) (CasabacGS) Connector: C      aaa  sss   aaa  bbbb   aaa  cccc
20040105/120552/653 (I) (CasabacGS) Connector: C      a  a  s   a  a  b   b  a   a  c
20040105/120552/653 (I) (CasabacGS) Connector: C      a  a  sss  a  a  b   b  a   a  c
20040105/120552/653 (I) (CasabacGS) Connector: C      a  a      s  a  a  b   b  a   a  c
20040105/120552/653 (I) (CasabacGS) Connector: CCCC  aaaa  sss   aaaa  bbbb   aaaa  cccc
20040105/120552/653 (I) (CasabacGS) Connector:

```

The system's default encoding is character set "Cp1252" (may be different in your system, Cp1252 is a character set defined by Microsoft). Casabac itself is using "UTF-8" (this is the configuration in casaconfig.xml).

What does this mean from compatibility point of view? Casabac now will access Layout Definitions and Multi Language Files by using character set "UTF-8". But: if having used previous versions of Casabac then your files were written using character set "Cp1252" - a one byte character set. As consequence characters with code greater than 127 will mess the encoding and decoding process, typically resulting in "strange characters" appearing in your browser.

What to do? There are two possibilities:

- **"Quick one"**: change the character set of your casaconfig.xml to your system's default character set (e.g. "Cp1252" in Windows based systems) - then there no change needs to be done in your files:

```

<casaconfig ...
  textencoding="cp1252"
  ...>
</casaconfig>

```

If having taken over the updated web.xml then remove the <MIME-MAPPING> part at the end.

- **"Better one"**: use "UTF-8" because of its advantages and convert your files from the previous format to UTF-8. There is a special tool coming with Casabac that makes this conversion very simple! Please read the next chapter for more information.

What is the reason for qualifying the second possibility to be better than the first one? - The first one is limited to your system's default character set - there is no way to use e.g. Japanese or Chinese characters. - On a mid to long term time range we strongly recommend to convert to UTF-8 - in order to not only be "multi language enabled" but also "multi character enabled".

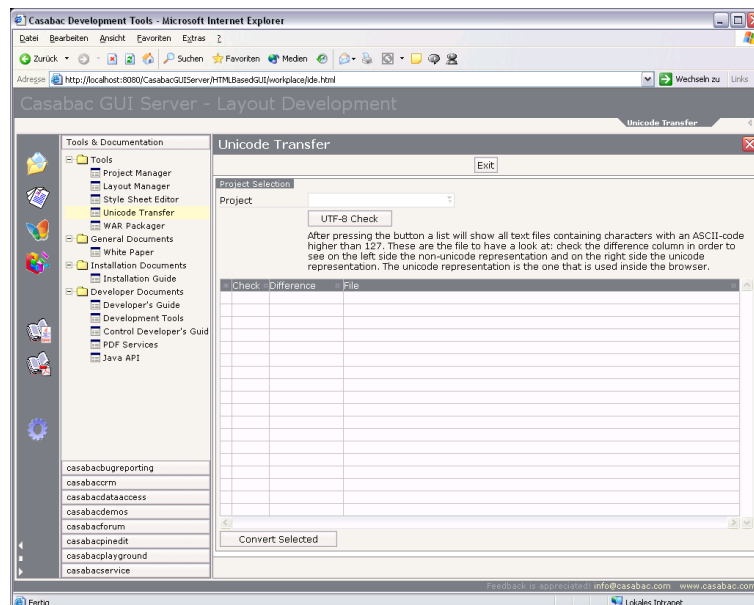
Transferring non-UTF-8 Files to UTF-8

When having used Casabac before build 25_20040105 Layout Definitions and Multi Language Files were stored using your system's default encoding.

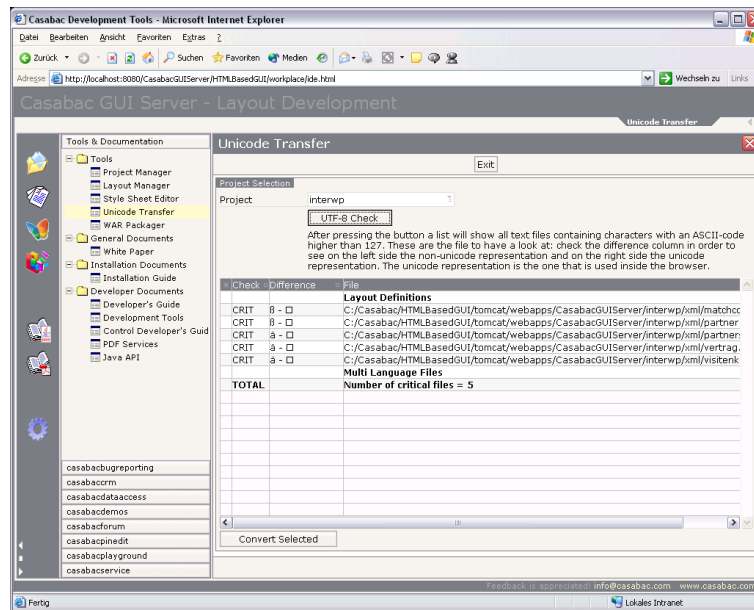
Unicode UTF-8 and the typical default character set of your system share the one byte encoding between 0-127. I.e. if you so far only have maintained characters in this area (the typical English ones...) then there's nothing to do (...but better check!). If you are e.g. a German user of Casabac than you may have input some characters above 127, e.g. the Euro character "€", or "Ä"/"Ö"/"Ü" characters. In this case your files keeping the textual information need to be transferred from your system's default encoding to Unicode UTF-8.

Unicode Transfer Tool

When starting the Casabac Development Workplace you can find a tool "Unicode Transfer":

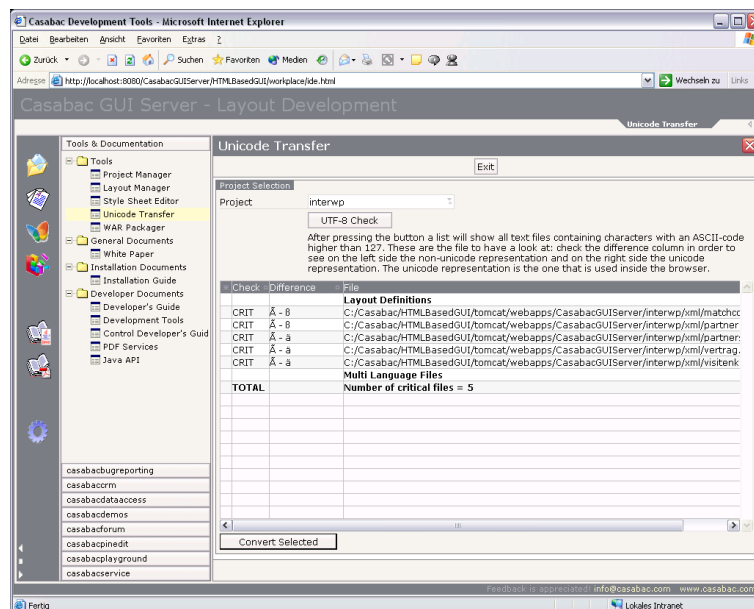


Select your project(s) and press the "UTF-8 Check" button. The system will scan all Layout Definitions and Multi Language Files and will test if they contain characters above code 127:



In the example the system finds 5 layout definitions. In the "Difference" column the system shows the first character above 127 which was found in the document - on the left you see the representation using your system's default encoding, on the right the Unicode UTF-8 representation. In the example you see that the characters on the left are "proper ones" and on the right are "non proper ones".

After having selected all the five lines (use click/shift-click for ranges selections) and having pressed "Convert Selected" the screen looks as follows:



Now the UTF-8 representation looks "proper" and the representation using your system's default character set looks "strange" - the files were converted successfully.

Please make sure that the files are writable when using the tool!

Pay attention: do only run the conversion one time! Doing the same conversion a second time will again convert - and mix up your characters higher than code 127.

WEB.XML Settings if using Tomcat Servlet Engine

Some servlet containers like Tomcat require an explicit definition in the web.xml file in order to generate the correct http header information.

```
<web-app>
  ...
  ...
  ...
  ...
  </servlet-mapping>
  <mime-mapping>
    <extension>html</extension>
    <mime-type>text/html; charset=UTF-8</mime-type>
  </mime-mapping>
</web-app>
```

Encoding/Decoding Considerations for Your Application

Accessing Character Information

You have to be aware of encoding/decoding within your application at any point you access character data and explicitly or implicitly convert it into Java characters:

- Reading/writing files
- Reading/writing data from/to database

There are no "extra Casabac issues" you have to be aware of - Casabac talks to you via a Java API (your adapters) and as consequence there is Unicode usage by Java on both sides.

Directly Manipulating Casabac Files

When using the standard Casabac delivery, that is based on UTF-8 from build 25_20040104 on then please pay attention: if you directly maintain Casabac files (e.g. you maintain layout definitions via a textual editor, and not via Casabac's Layout Painter) then be aware of that the file is in UTF-8 format. Characters higher below code 127 are no problem, but characters above are. If your editor is using ISO 8859-1 (or something else) then the data you input (e.g. a Euro symbol "€") will not be correctly displayed in UTF-8.

Make sure that you use a text editor that is UTF-8 enabled!