



The Tivoli Storage Manager Client and UTF-8

Using the Linux and UNIX Backup-Archive Client to Backup and Restore Files in Multiple Byte Character Set Environments such as UTF-8

By Peter Symonds
Version 1.0

Copyright Notice

Copyright IBM Corporation 2008. All rights reserved. May only be used pursuant to a Tivoli Systems Software License Agreement, an IBM Software License Agreement, or Addendum for Tivoli Products to IBM Customer or License Agreement. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without prior written permission of IBM Corporation. IBM Corporation grants you limited permission to make hardcopy or other reproductions of any machine-readable documentation for your own use, provided that each such reproduction shall carry the IBM Corporation copyright notice. No other rights under copyright are granted without prior written permission of IBM Corporation. The document is not intended for production and is furnished "as is" without warranty of any kind. All warranties on this document are hereby disclaimed, including the warranties of merchantability and fitness for a particular purpose.

U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corporation.

Trademarks

IBM, the IBM logo, Tivoli, the Tivoli logo, AIX, Cross-Site, NetView, OS/2, Planet Tivoli, RS/6000, Tivoli Certified, Tivoli Enterprise, Tivoli Enterprise Console, Tivoli Ready, and TME are trademarks or registered trademarks of International Business Machines Corporation or Tivoli Systems Inc. in the United States, other countries, or both.

Lotus is a registered trademark of Lotus Development Corporation. Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. UNIX is a registered trademark of The Open Group in the United States and other countries. C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both. PC Direct is a trademark of Ziff Communications Company in the United States, other countries, or both and is used by IBM Corporation under license. ActionMedia, LANdesk, MMX, Pentium, and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. For a complete list of Intel trademarks, see <http://www.intel.com/sites/corporate/trademarx.htm>. SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC. For further information, see <http://www.setco.org/aboutmark.html>. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Notices

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to valid intellectual property or other legally protectable right of Tivoli Systems or IBM, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user. Tivoli Systems or IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, New York 10504-1785,

U.S.A.

About the Tivoli Field Guides

Sponsor

Tivoli Customer Support sponsors the Tivoli Field Guide program.

Authors

Those who write field guides belong to one of these three groups:

- Tivoli Support and Services Engineers who work directly with customers
- Tivoli Customers and Business Partners who have experience using Tivoli software in a production environment
- Tivoli developers, testers, and architects

Audience

The field guides are written for all customers, both new and existing. They are applicable to external audiences including executives, project leads, technical leads, team members, and to internal audiences as well.

Types of Field Guides

Two types of Tivoli Field Guides describe how Tivoli products work and how they are used in real life situations:

- Field Guides for technical issues are designed to address specific technical scenarios or concepts that are often complex to implement or difficult to understand, for example: endpoint mobility, migration, and heartbeat monitoring.
- Field Guides for business issues are designed to address specific business practices that have a high impact on the success or failure of an ESM project, for example: change management, asset Management, and deployment phases.

Purposes

The Field Guide program has two major purposes:

- To empower customers & business partners to succeed with Tivoli software by documenting and sharing product information that provides accurate and timely information on Tivoli products and the business issues that impact an enterprise systems management project
- To leverage the internal knowledge within Tivoli Customer Support and Services and the external knowledge of Tivoli customers and Business Partners

Availability

All completed field guides are available free to registered customers and internal IBM employees at the following Web site:

http://www.ibm.com/software/sysmgmt/products/support/Field_Guides.html

Authors can submit proposals and access papers by e-mail:

mailto:Tivoli_eSupport_Feedback@us.ibm.com

Table of Contents

Sponsor	5
Authors	5
Audience	5
Types of Field Guides	5
Purposes	5
Availability	5
LOCALIZATION IN MULTIPLE LANGUAGE ENVIRONMENTS ...	9
LOCALIZATION OVERVIEW	9
<i>Message Translation</i>	9
<i>Character Set Encoding</i>	10
<i>Character Representation</i>	10
<i>The Locale Setting</i>	11
DISPLAY ISSUES.....	11
<i>The Terminal Character Set Encoding and the Locale</i>	11
<i>The Terminal Encoding Applies to Remote Telnet</i>	13
<i>Some Telnet and SSH Clients Do Not Handle Multiple Byte</i> <i>Characters Correctly</i>	14
FILE NAMES.....	15
<i>Where Do Files Get Their Names?</i>	15
<i>The Encoding of a New File Name is Determined by the Terminal</i> <i>Encoding and the Locale Setting</i>	15
<i>How Can I Discover a File Name's Encoding?</i>	17
FILE SYSTEM ENCODING ENVIRONMENTS	18
FILE NAMES ARE IN THE SAME SINGLE BYTE CHARACTER SET ENCODING AND ALL FILE NAMES ARE IN THE SAME LANGUAGE.....	18
<i>How do I know I'm in this environment?</i>	18
<i>Tivoli Storage Manager Client concerns</i>	18
FILE NAMES ARE IN THE SAME UTF-8 ENCODING AND THE FILE NAMES MAY BE IN MULTIPLE LANGUAGES	19
<i>How Do I Know I'm in this Environme</i>	19
<i>Tivoli Storage Manager Client Concerns</i>	20
FILE NAMES ARE IN DIFFERENT SINGLE BYTE CHARACTER SET ENCODINGS	

AND THE FILE NAMES ARE IN MULTIPLE LANGUAGES.....	24
<i>How Do I Know I'm in this Environment?</i>	24
<i>Tivoli Storage Manager Client Concerns</i>	24
FILE NAMES ARE IN DIFFERENT ENCODINGS, SOME OF WHICH ARE IN A	
DBCS OR UTF-8 ENCODING.....	27
<i>How Do I Know I'm in this Environment?</i>	27
<i>If You Are On a Linux System, You Are Probably in this</i>	
<i>Environment</i>	27
<i>An Environment Where the File Names Have Different Encodings is</i>	
<i>Complicated and Difficult to Administer</i>	28
<i>Tivoli Storage Manager Client concerns</i>	29
<i>Files Can Always Be Backed Up or Restored, But Care</i>	
<i>Must Be Used in entering or displaying the file names</i>	31
 MIGRATING TO A DIFFERENT CHARACTER SET	
ENCODING IS DIFFICULT.....	33
RENAMING FILES FROM ONE CHARACTER SET ENCODING IS A TWO	
STEP OPERATION.....	33
RENAMING A FILE MAY NOT BE SUFFICIENT.....	34
THE GLOBAL ENVIRONMENT MUST BE CHANGED FOR ALL USERS.....	34
 CHARACTER SET ENCODING STANDARDS.....	35
THE ISO 8859 VARIATIONS.....	35
UNICODE AND UTF-8.....	35
JAPANESE EXTENDED UNIX CODE.....	36

The Tivoli Storage Manager Client and UTF-8

Using the Linux and UNIX Backup-Archive Client to Backup and Restore Files in Multiple Byte Character Set Environments such as UTF-8

1. Localization in Multiple Language Environments

This paper discusses the Tivoli Storage Manager for Linux and UNIX Backup -Archive Client operation in environments with UTF-8 and Double Byte Character Set (DBCS) encoded file names. **Deployed correctly, the Tivoli Storage Manager Linux and UNIX Backup -Archive Client can be configured to properly protect all files regardless of their file name or the encoding used to represent the file name.** The purpose of this paper is to help you understand UTF-8 and DBCS issues and the best way to achieve your desired results with Tivoli Storage Manager.

Localization Overview

What are characters used in the file names of the files on your file servers? If the characters are exclusively basic Latin characters (A-Z, a-z, 0-9, etc.), localization is not much of a concern. As soon as you start seeing extended Latin characters (e.g., í, ñ, etc.), Greek, Cyrillic, Arabic, Chinese, or other characters, you need to take a long hard look at localization. If your file server is serving clients in Amsterdam, Athens, Beijing, or Berlin, chances are you have left the comfortable world of basic Latin and should read on.

Localization is a general term for the process of customizing the display and keyboard characteristics of a computer for a particular country or location. Localization involves message translation, character encoding, character representation, keyboard usage, and date, time and number formats.

For instance, the German word for "date" is "Datum", the Russian word for date is "дата", and the Japanese characters for date are "日付". Notice that both the translation and the characters used for the translation differ between the different languages.

Message Translation

Software products that have been localized, such as the Tivoli Storage Manager Client, provide message catalogs that have been translated into multiple languages, and the users of localized products have the option to specify the language used during product interaction.

The Tivoli Storage Manager Client provides message catalogs in fourteen languages: English, Czech, French, German, Hungarian, Italian, Polish, Portuguese, Russian, Spanish, Japanese, Korean, Simplified Chinese and Traditional Chinese.

The Tivoli Storage Manager client automatically detects the language specified by the locale and displays all messages in that language. For example, in a French operating system where the LANG environment variable is "LANG=fr_FR", the Tivoli Storage Manager Client attempts to use the French message catalog. If the Tivoli Storage Manager Client cannot load the French

message catalog, or if the client is running on an unsupported language, such as French/Canada (LANG=fr_CA) (fr for ISO 8859-1 French and CA for Canada), the Tivoli Storage Manager Client defaults to the United States English message catalog and all messages will appear in English.

Character Set Encoding

File names on a computer system are typically strings of characters that can be entered on the keyboard and displayed on the computer monitor. Each character displayed on the screen is encoded in a binary representation on the file system. The mapping between the character representation displayed on the computer monitor and its binary representation in the file system is defined by a **character encoding** standard. There are a number of standards. For example, the ISO 8859 standard maps the character 'A' as the eight bit byte 0x41.

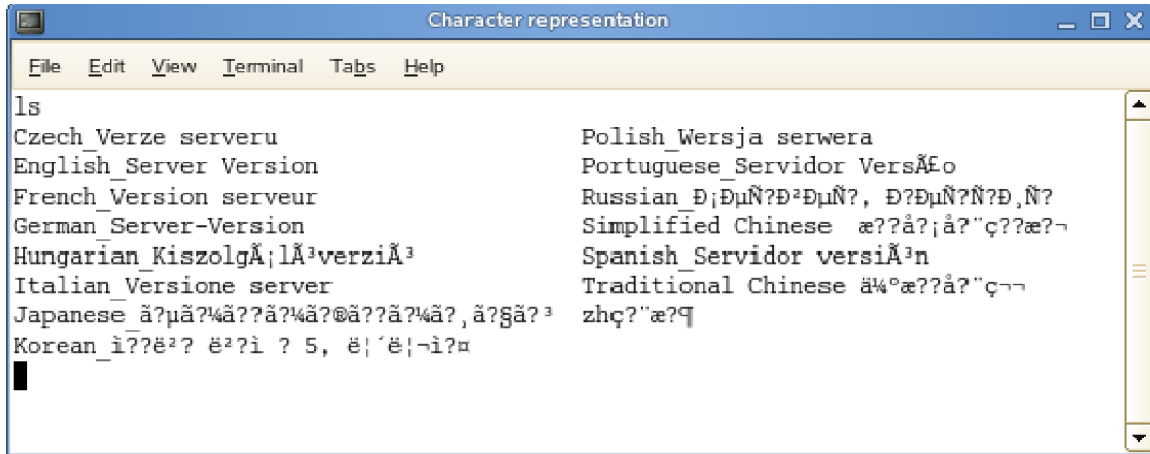
The basic Latin characters (A-Z, a-z, 0-9, etc.) compose the seven bit ASCII characters (the high order bit of the eight bit byte for these characters is always zero), and they have the same encoding in all supported locales, ISO 8859, DBCS and UTF -8. If your file names are composed exclusively of basic Latin characters, the file names will always appear the same regardless of the locale.

Characters that have a binary encoding that is greater than the seven bit ASCII characters have different mappings in the different character encoding standards. For example, the binary encoding 0xA3 in the ISO 8859-1 standard represents the character '£'. The same binary encoding represents the character 'ł' in the ISO 8859-3 standard.

Character Representation

The representation of individual characters in your terminal window is a function of the terminal window character set encoding configuration and the fonts used. If the terminal window character set encoding and fonts used are appropriate to the characters to be represented, the characters will be displayed as expected. However, if the terminal window character set encoding and font are not appropriate for the actual characters, then the display will be incorrect. It should be noted that the basic Latin characters (A-Z, a-z, 0-9, etc.) have the same encoding in all supported character set encodings so these characters are always displayed correctly.

File names are not displayed correctly when the terminal window character set encoding and the file name character set encoding do not match. The example below shows what happens when the encoding of the file names and the terminal do not match. The file names in this example are all encoded in UTF-8 and the terminal is running in an English ISO-8859-15 locale. Only the basic Latin characters (A-Z, a-z, 0-9, etc.) in the file names are displayed correctly.



The Locale Setting

The locale is not a file name characteristic; instead, it defines an execution environment. The LANG environment variable is used to set the locale. The fully qualified LANG environment variable has two parts. The first part determines the language used for messages, and the second part determines how the binary encoded individual characters are treated. For example, if LANG=en_US.ISO8859-1, the English language message catalogs are used, and all characters are expected to be in the ISO 8859-1 character set. If LANG= DE_DE.UTF-8, the German language message catalogs are used and all characters are expected to be in the UTF-8 character set.

The locale setting determines how programs running in that environment treat the individual characters in the file name and the message catalogs used when writing out messages. Changing the locale has no effect on the binary encoding of the file names, it just changes how the file name characters are processed by programs executing in that locale and the languages used for messages.

Display Issues

The Terminal Character Set Encoding and the Locale

The terminal, or terminal emulator to be precise, is a window for communicating with a computer system. Commands can be entered in the terminal window, and the results of the commands can be viewed in the same window.

In a modern computer system supporting multiple languages, the terminal must be able to handle strings with different character set encodings. Terminals, such as the Gnome terminal that is used for the examples in this document, have a pull down menu that can be used to set the character set encoding of the terminal. The encoding configuration chosen in the menu determines how the characters are displayed in the window and how characters entered on the command line are

encoded into a file's name or contents. If the terminal character set encoding is set incorrectly, file names composed of characters other than the basic Latin characters may not be handled correctly.

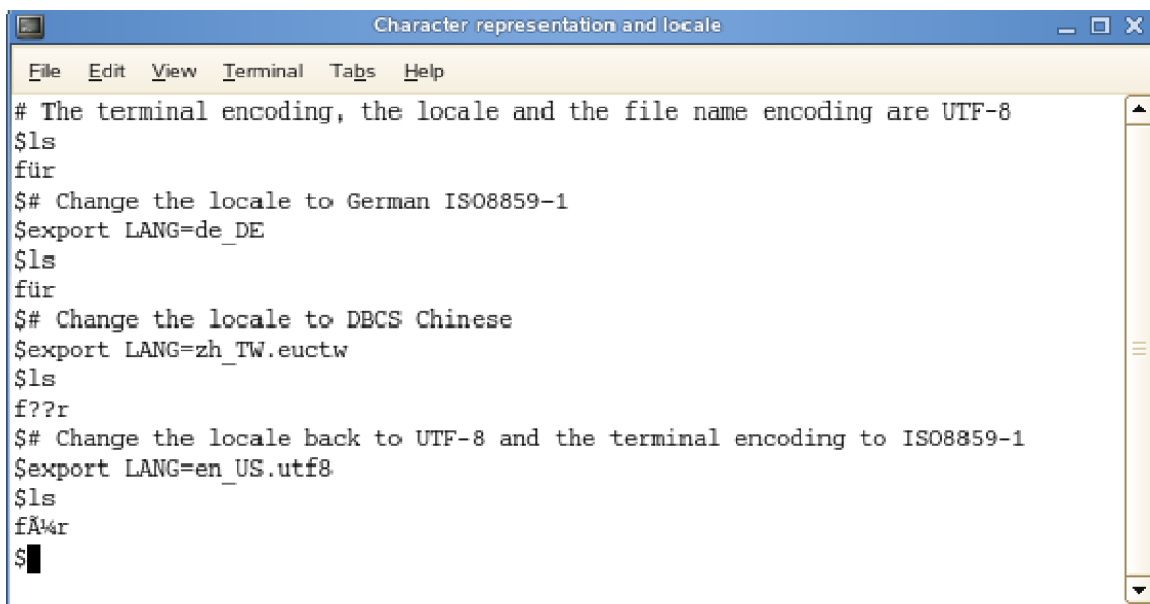
The locale setting determines how programs running in that environment handle the individual characters. A program, such as the `ls` command, uses the locale to determine how to parse the characters in the file names before sending them to the terminal window to be displayed. This interaction between the terminal character set encoding and the locale occurs because the terminal window does not display characters all by itself.

The first few lines in the example below shows how the characters displayed in the terminal window differ when the locale is changed and the terminal encoding is kept constant. The terminal encoding and the locale were initially set to UTF-8. Notice that all of the characters, including the German character "ü" in the file name displayed by the `ls` command are displayed correctly.

If the locale is changed to `de_DE`, which indicates that the German language message catalogs are to be used and all characters are expected to be in the ISO8859-1 character set and the terminal encoding is left as UTF-8, the display of the file name still looks correct. This is because the UTF-8 character "ü" is also a valid ISO8859-1 character. However, if the locale is changed to the Chinese DBCS `zh_TW` locale, the "ü" is no longer displayed correctly because the character "ü" is not defined in the `zh_TW.euctw` character mappings.

If the locale is changed back to the UTF-8 locale and the terminal encoding is changed to ISO8859-1, the UTF-8 multiple byte character "ü" is now displayed incorrectly as "A1/4". This is because the multiple byte UTF-8 character "ü" maps into two single byte ISO8859-1 characters.

In general, both the terminal encoding and the locale must have the same character set definitions for the display to be correct.

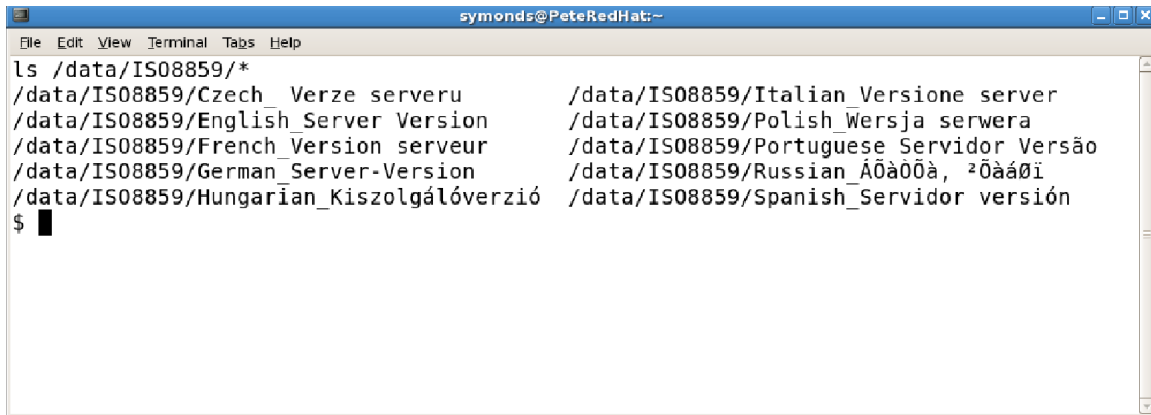


```
Character representation and locale
File Edit View Terminal Tabs Help
# The terminal encoding, the locale and the file name encoding are UTF-8
$ls
für
$# Change the locale to German ISO8859-1
$export LANG=de_DE
$ls
für
$# Change the locale to DBCS Chinese
$export LANG=zh_TW.euctw
$ls
f??r
$# Change the locale back to UTF-8 and the terminal encoding to ISO8859-1
$export LANG=en_US.utf8
$ls
fA1/4r
$
```

The Terminal Encoding Applies to Remote Telnet and SSH Sessions

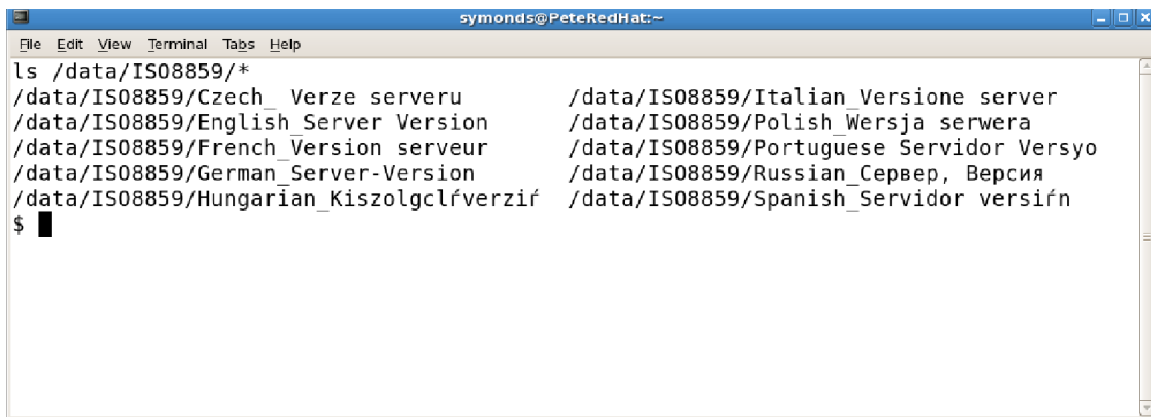
The terminal encoding on the local system applies to both the display of names on the local system and to the display of names received from a telnet or ssh session connected to a remote system. This appears counter intuitive, but when you realize that the telnet or ssh session is actually a program (just like the `ls` command) running in the terminal window, it becomes clear why the terminal encoding on the local systems applies to characters received from a remote system.

In the example below, some single byte character set file names composed of characters from multiple different ISO8859 standards are shown. When the terminal encoding is set to ISO 8859 - 1 the display is as follows. Notice that the Russian file names are obviously incorrect.



```
symonds@PeteRedHat:~  
File Edit View Terminal Tabs Help  
ls /data/ISO8859/*  
/data/ISO8859/Czech_Verze serveru      /data/ISO8859/Italian_Versione server  
/data/ISO8859/English_Server Version  /data/ISO8859/Polish_Wersja serwera  
/data/ISO8859/French_Version serveurur /data/ISO8859/Portuguese Servidor Versão  
/data/ISO8859/German_Server-Version    /data/ISO8859/Russian_Ã0à00à, ²0àá0ï  
/data/ISO8859/Hungarian_Kiszolgalóverzió /data/ISO8859/Spanish_Servidor versión  
$ █
```

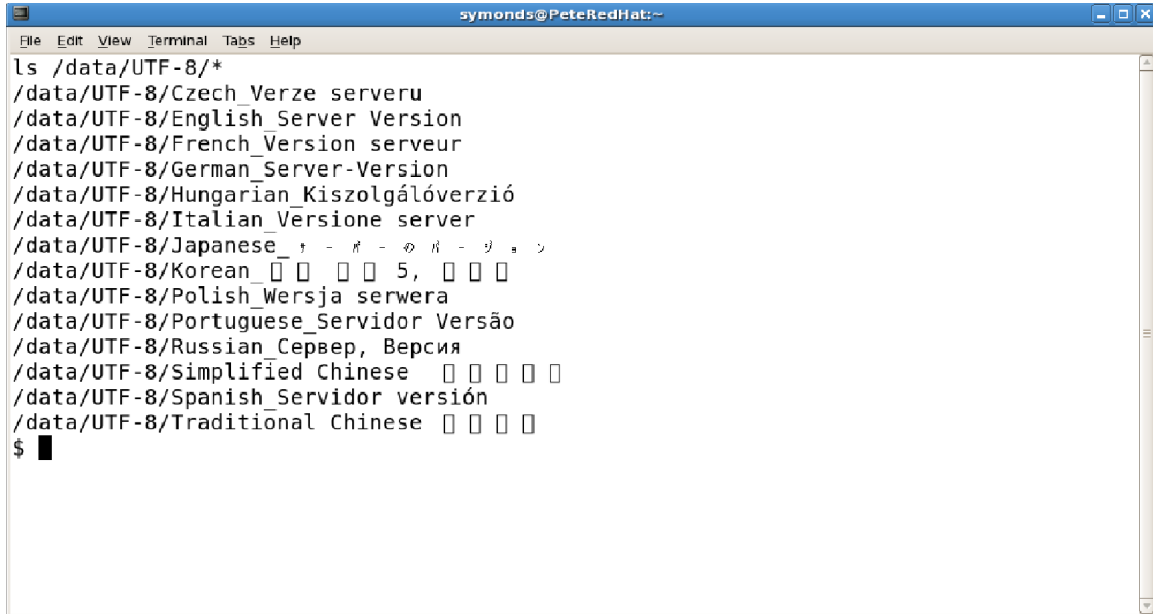
If the terminal encoding is changed to ISO 8859-5, the Russian ISO 8859-5 characters are now displayed correctly, but some of the other file names appear differently. If you look closely, you will see that the Spanish, Hungarian and Portuguese file names are not the same as when the encoding was set to ISO 8859-1.



```
symonds@PeteRedHat:~  
File Edit View Terminal Tabs Help  
ls /data/ISO8859/*  
/data/ISO8859/Czech_Verze serveru      /data/ISO8859/Italian_Versione server  
/data/ISO8859/English_Server Version  /data/ISO8859/Polish_Wersja serwera  
/data/ISO8859/French_Version serveurur /data/ISO8859/Portuguese Servidor Versyo  
/data/ISO8859/German_Server-Version    /data/ISO8859/Russian_Сервер, Версия  
/data/ISO8859/Hungarian_Kiszolglcl'verzif /data/ISO8859/Spanish_Servidor versifn  
$ █
```

Some Telnet and SSH Clients Do Not Handle Multiple Byte Characters Correctly

Some versions of telnet and ssh clients are not UTF-8 enabled and do not handle multiple byte characters correctly. If you get a display similar to the one shown below, try to update to a telnet or ssh client that supports multiple byte characters.



```
symonds@PeteRedHat:~  
File Edit View Terminal Tabs Help  
ls /data/UTF-8/*  
/data/UTF-8/Czech Verze serveru  
/data/UTF-8/English_Server Version  
/data/UTF-8/French_Version serveurur  
/data/UTF-8/German_Server-Version  
/data/UTF-8/Hungarian_Kiszolgálóverzió  
/data/UTF-8/Italian_Versione server  
/data/UTF-8/Japanese_ - - - -  
/data/UTF-8/Korean_ 5,   
/data/UTF-8/Polish_Wersja serwera  
/data/UTF-8/Portuguese_Servidor Versão  
/data/UTF-8/Russian_Сервер, Версия  
/data/UTF-8/Simplified Chinese   
/data/UTF-8/Spanish_Servidor versión  
/data/UTF-8/Traditional Chinese   
$
```

File Names

Where Do Files Get Their Names?

Files are typically created either by a command issued in a terminal emulator's window or by providing a name in a full screen editor's window. The character set encoding configuration of the terminal or editor and the locale determine how all characters, including file names, are encoded in the resultant file on the disk. If the encoding configuration and locale are UTF-8, all characters are encoded as UTF-8 characters; if the encoding configuration and locale are ISO 8859-1, all characters are encoded as ISO 8859-1 characters. Changing the character set encoding used to create a new file is as simple as changing the terminal or full screen editor's encoding configuration and the locale.

The Encoding of a New File Name Is Determined by the Terminal Encoding and the Locale Setting

The example below demonstrates how easy it is to create file names with different character set encodings by simply changing the terminal emulator encoding. Initially, the locale terminal encoding and locale were set to the DBCS traditional Chinese (zh_TW.euctw) encoding and the file Chin1名 was created. The **ls** command was then used to display the contents of the directory.

Then both the terminal encoding and the locale were switched to UTF-8 and a second file Chin2名 was created. With both the terminal encoding and the locale as UTF-8, the UTF-8 file Chin2名 displayed correctly, but the DBCS file Chin1名 displayed incorrectly. The directory now has two files, a zh_TW.euctw file Chin1名 and a UTF-8 file Chin2名.

Next, the locale was changed to zh_TW.euctw and the terminal encoding was left as UTF-8. A third file, Chin3名 was created with mixed UTF-8/zh_TW.euctw encoding. The **ls** command was used again to display the contents of the directory. The mixed encoding file Chin3名 are not displayed correctly in this or any environment, and neither the zh_TW.euctw file, Chin1名 nor the UTF-8 file Chin2名 are displayed correctly in the mixed locale/encoding environment.

A similar mixed locale environment was established, with the terminal encoding set to zh_TW.euctw and the locale set to UTF-8. A fourth file, Chin4名 was created with mixed UTF-8/zh_TW.euctw environment. The **ls** command was again used to display the contents of the directory. The mixed encoding files Chin3名 and Chin4名 cannot be displayed correctly in this or any environment, and neither the zh_TW.euctw file, Chin1名 or the UTF-8 file Chin2名 are displayed correctly in the mixed locale/encoding environment.

If the locale and the terminal encoding are switched back to DBCS traditional Chinese (zh_TW.euctw), the file Chin1名 is displayed correctly and the UTF-8 file Chin4名 is displayed with an additional garbage character due to the extra bytes in the UTF-8 encoding.

It is important to keep both the locale and the terminal encoding in sync. If files are created when the locale and the encoding are not in synchronization, the encoding of the files will be invalid.

```

Locale and file name encoding
File Edit View Terminal Tabs Help
# Start in traditional Chinese locale and encoding
$vi Chin1名
$ls
Chin1名
$# Switch to a UTF-8 locale and encoding
$export LANG=zh_TW.utf8
$vi Chin2名
$ls
Chin1?? Chin2名
$# Switched to a mixed locale/encoding LANG=zh_TW.euctw/UTF-8 encoding
$export LANG=zh_TW.euctw
$vi Chin3名
$lls
bash: lls: command not found
$ls
Chin1[] Chin2???? Chin3?????
$# Switch to a mixed locale/encoding LANG=UTF-8 and traditional Chinese encoding
$export LANG=zh_TW.utf8
$vi Chin4名
$lls
bash: lls: command not found
$ls
Chin1?? Chin2[] [] [] Chin3[] [] []? Chin4????
$# Switch locale to traditional Chinese
$export LANG=zh_TW.euctw
$ls
Chin1名 Chin2???? Chin3?????? Chin4名?
$

```

The `od` command was used to display the hexadecimal encoding of the file names. Notice that the first four characters of each file name “Chin” are identical 0X’6843 6e69’. The next two bytes of each name (with the bytes in little endian) are c731, e532, e533, c734, corresponding to Chin1名, Chin2名, Chin3名, and Chin4名. However the remaining bytes in the file names corresponding to the character “名” are all different. The file Chin1名 has a valid DBCS encoding and the file Chin2名 has a valid UTF-8 encoding. The character encodings of the mixed UTF-8/zh_TW.euctw file names Chin3名 and Chin4名 are invalid.

```

Locale and file name encoding continued
File Edit View Terminal Tabs Help
# Show the file name encodings with od -x
$ls Chin1* | od -x
0000000 6843 6e69 c731 0ad8
0000010
$ls Chin2* | od -x
0000000 6843 6e69 e532 8d90 000a
0000010
$ls Chin3* | od -x
0000000 6843 6e69 e533 8d90 90e5 000a
0000010
$ls Chin4* | od -x
0000000 6843 6e69 c734 c7d8 000a
0000010
$

```

New files can also come into existence via UNIX commands such as **mv** or **cp**. However, these commands cannot change the character set encoding of the file name because both the initial file name and the new file name must be specified on the same command line with the same terminal emulator encoding.

How Can I Discover a File Name's Encoding?

The simplest way to discover a file name's encoding is to display the file name multiple times with the **ls** command while changing the terminal encoding and locale settings. The terminal encoding and locale setting that displays the name correctly corresponds to the file name's character set encoding.

Using the **ls** command to discover a file name's encoding requires that both the locale and the terminal encoding settings are changed in synchronization.

It should be noted that file names solely composed of the basic Latin characters (A-Z, a-z, 0-9, etc.) have the same encoding in all character set encodings, so such file names could be considered to have multiple possible character set encodings.

If it is not possible to use a configurable terminal emulator to discover the file name character set encoding, either because a configurable terminal emulator is not available or because no setting displays the name correctly, then the **od** command can be used to examine the binary encoding of the file name.

```
ls Ru*  
Russian_??????, ??????
```

```
ls Ru* | od -x  
7572 7373 6169 5f6e d5c1 d2e0 e0d5 202c  
d5b2 e1e0 efd8 000a
```

2. File System Encoding Environments

Because file names can be in multiple languages and there are at least two different character set encoding standards for each language (UTF-8 and a non-UTF-8 encoding standard), there are many possible file system character set encoding environments. Older legacy operating systems such as AIX and Solaris are typically ISO 8859 single byte character set encoding environments. New operating systems such as Linux are typically UTF-8 character set encoding environments.

This document discusses the operation of the TSM client in the following four environments.

1. All file names are in the same single byte character set encoding and all file names are in the same language.
2. All file names are in the same UTF-8 encoding; the file names may be in multiple languages.
3. File names are in different single byte character set encodings and the file names are in multiple languages.
4. File names have different multiple byte encodings, such as DBCS and UTF-8 encoding.

The Japanese, Chinese and Korean DBCS environments are not specifically covered because they are equivalent to the first case if all file names are in the same language. If files are in more than one language, this is equivalent to the fourth case.

1. File Names Are In the Same Single Byte Character Set Encoding and All File Names Are In the Same Language

This is the simplest environment. Localization concerns are minimal: messages are always in the same language, file names are always correctly displayed if the terminal encoding and locale are set correctly, and names can always be entered on the command line.

How Do I Know I'm in this Environment?

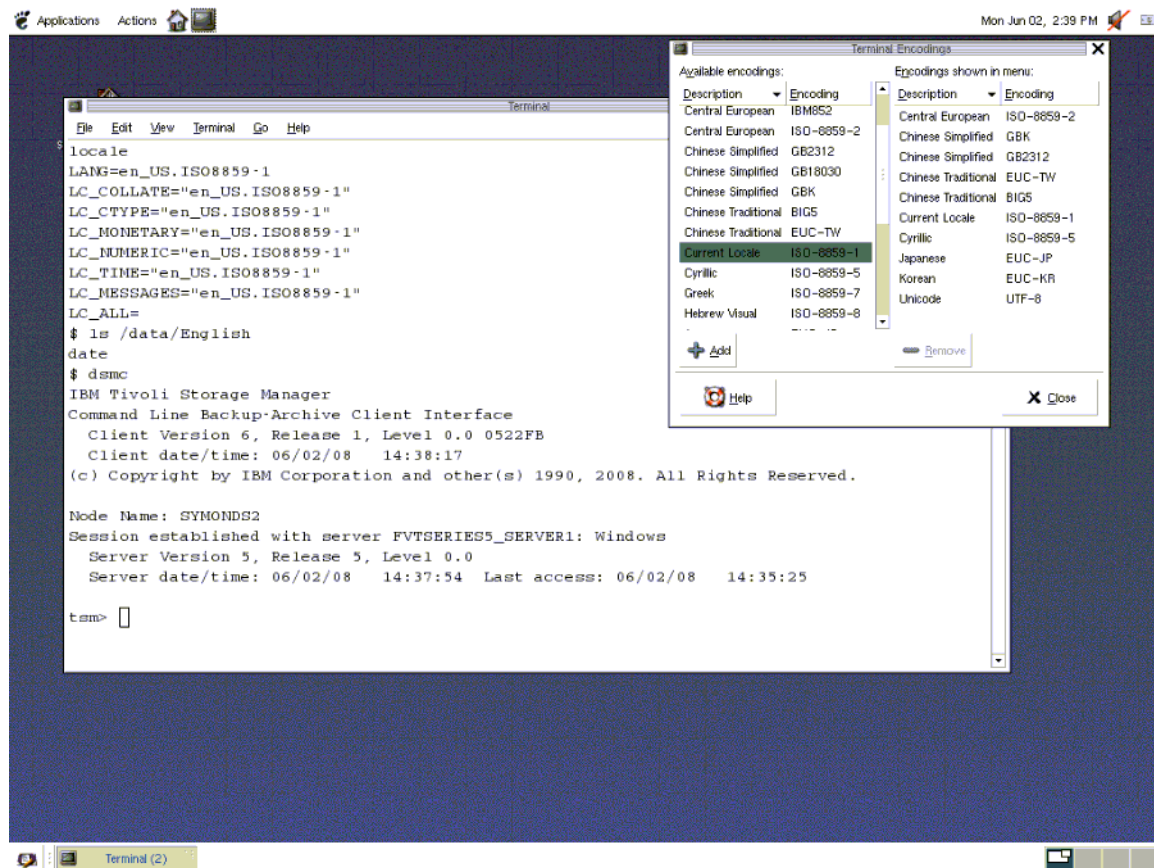
The most direct method to discover this environment is to query the locale with the command **locale**. If the locale is a single byte ISO 8859 locale and all file names display correctly on the command line and in the Tivoli Storage Manager client GUI and all names are in the same language, then you are in this environment. There is no support for Japanese, Chinese or Korean names in this environment.

Tivoli Storage Manager Client Concerns

There are no special considerations for using the Tivoli Storage Manager client to backup and restore files in this environment. All file names will be displayed correctly if the terminal character set encoding is set correctly and all file names can be entered on the Backup -Archive client command-line client. Requests to backup or restore a set of files using the wild card "*" specification will backup or restore all files meeting the selection criteria.

In the example below, the value of the locale environment variable is "LANG=en_US.ISO8859-1", and the Gnome terminal is configured to use ISO 8859-1 encodings. An ISO 8859-1 encoded file with the name "date" was created in this example.

The **ls** command correctly displays the English ISO 8859-1 file name and the TSM client messages are in English.



2. File Names Are In the Same UTF-8 Encoding and the File Names May Be in Multiple Languages

This is the most powerful environment and it is the only environment in which there are no restrictions on files with names in multiple languages. The UTF-8 environment is unique because all file names, regardless of the locale, have the same UTF-8 character set encoding, and all characters can always be entered and displayed if the terminal is configured properly. This is the only environment in which it is possible to simultaneously display and backup files with names composed of characters from multiple languages.

How Do I Know I'm in this Environment?

The most direct method to discover this environment is to query the locale with the command **locale**. If the locale is a UTF-8 locale then you may be in this environment. The pure UTF-8 environment is similar to single byte character set single language case in that all file names display correctly on the command line and in the Tivoli Storage Manager client GUI. However the names can be in multiple languages and Japanese, Chinese and Korean names are supported.

The biggest difference between this environment and the single byte character set single language environment is that multiple languages, including the Japanese, Chinese and Korean languages, are supported.

If the Tivoli Storage Manager client running in the UTF-8 locale can successfully backup all of the files in the system and no files are skipped on the backup, then you are in this environment.

If some files are skipped when running a Tivoli Storage Manager client backup and you see the message shown below, then you are not in this environment; instead, you are in environment 4, "File names have different multiple byte encodings, such as DBCS and UTF-8 encoding"

ANS4042E Object name '/testData/en_US_files/file3?' contains one or more unrecognized characters and is not valid

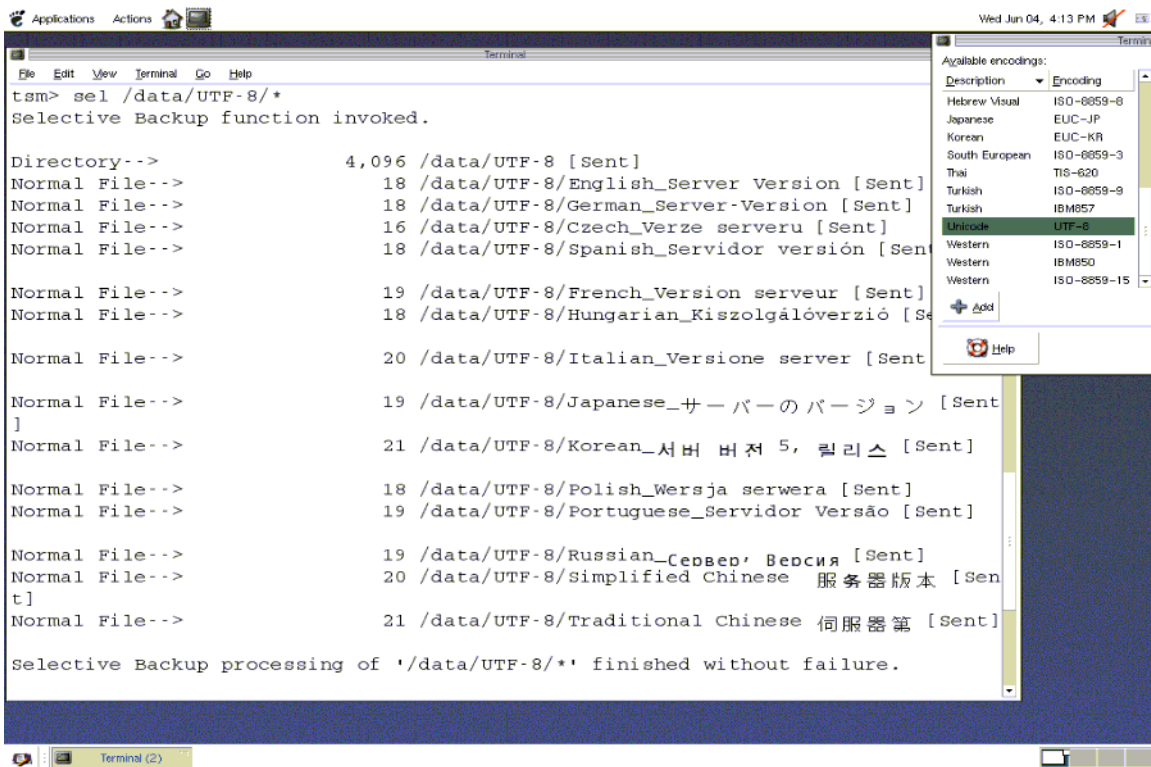
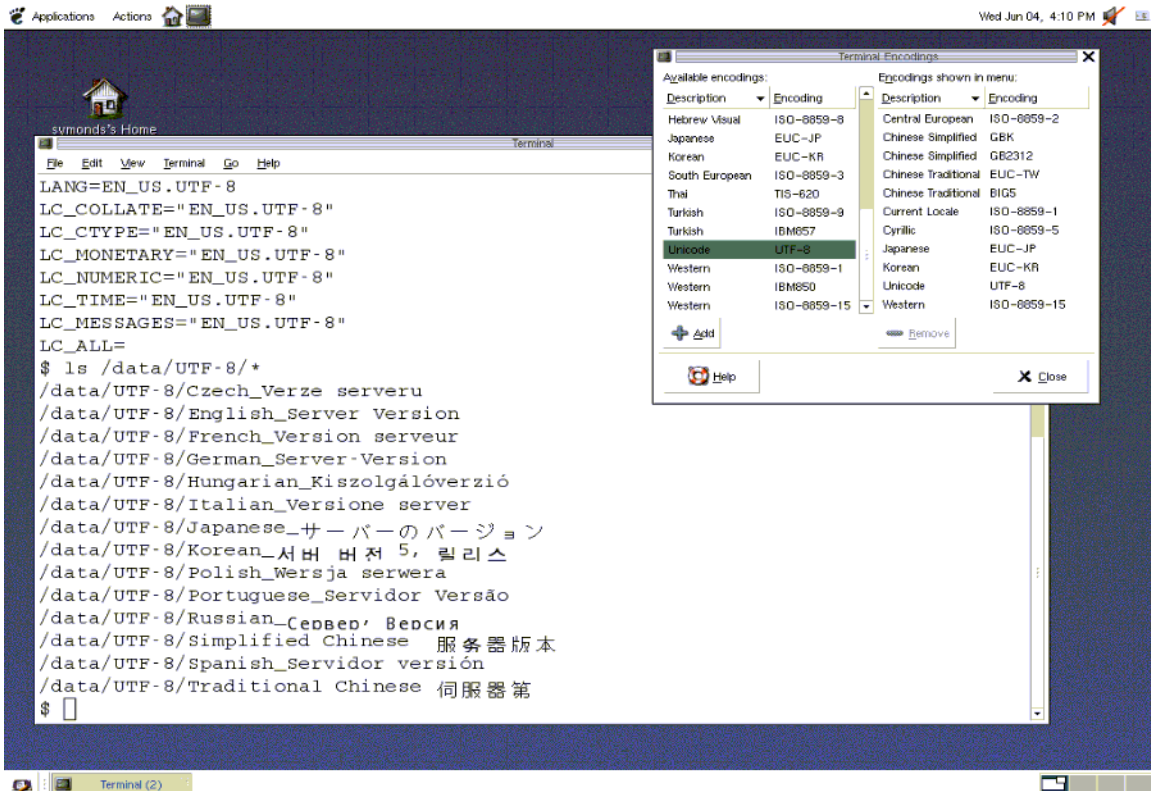
Tivoli Storage Manager Client Concerns

Just like the first environment, there are no special considerations for using the Tivoli Storage Manager client to backup and restore files in this environment. The principle difference is that the locale is UTF-8, not a single byte character set ISO 8859 variant. All file names will be displayed correctly and all file names can be entered on the Backup -Archive client command-line client. Requests to backup or restore a set of files using the wild card "*" specification will backup or restore all files meeting the selection criteria.

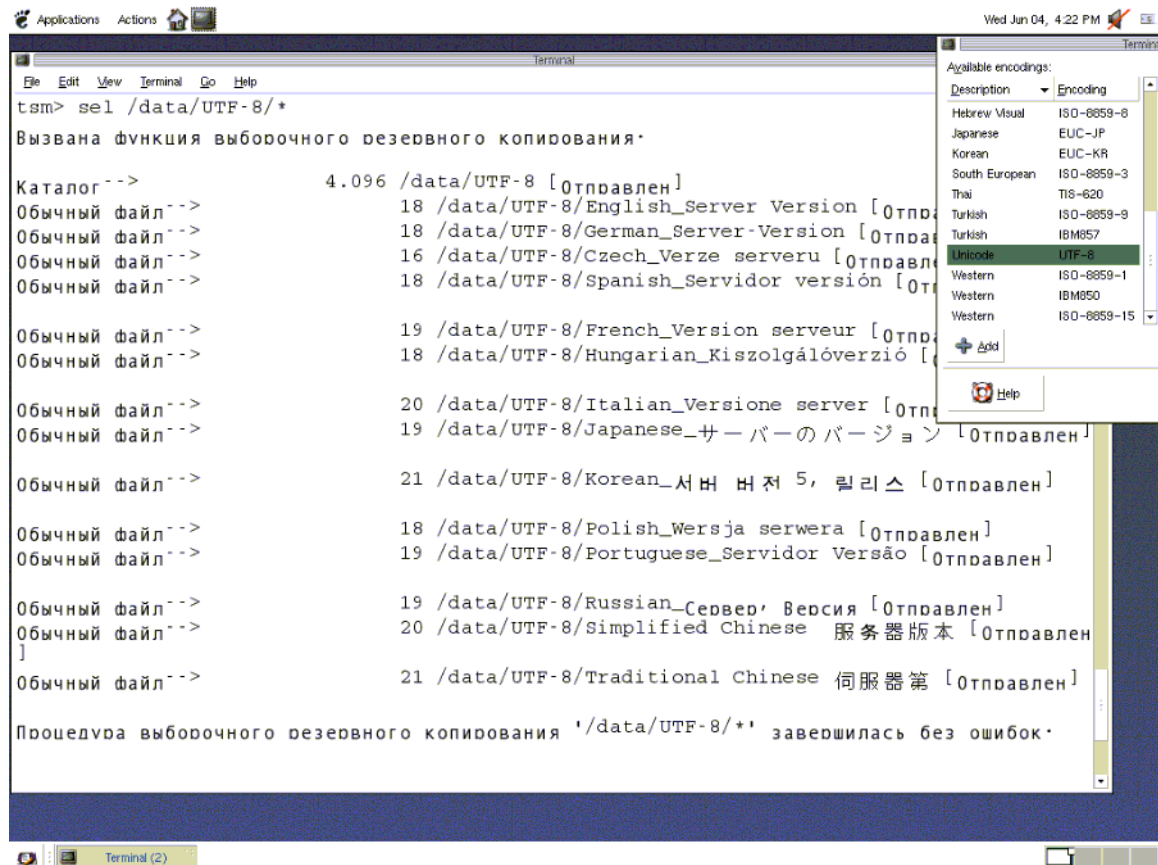
This is a very powerful environment, but it is easy to add file names that contain invalid UTF-8 characters and contaminate the pure UTF-8 environment. Running a selective backup in the UTF-8 locale is one way to discover if the environment is contaminated and to identify all invalid names.

In the following UTF-8 examples, fourteen files with names corresponding to the words for "Server Version" in the fourteen different languages supported by the Tivoli Storage Manager client were created.

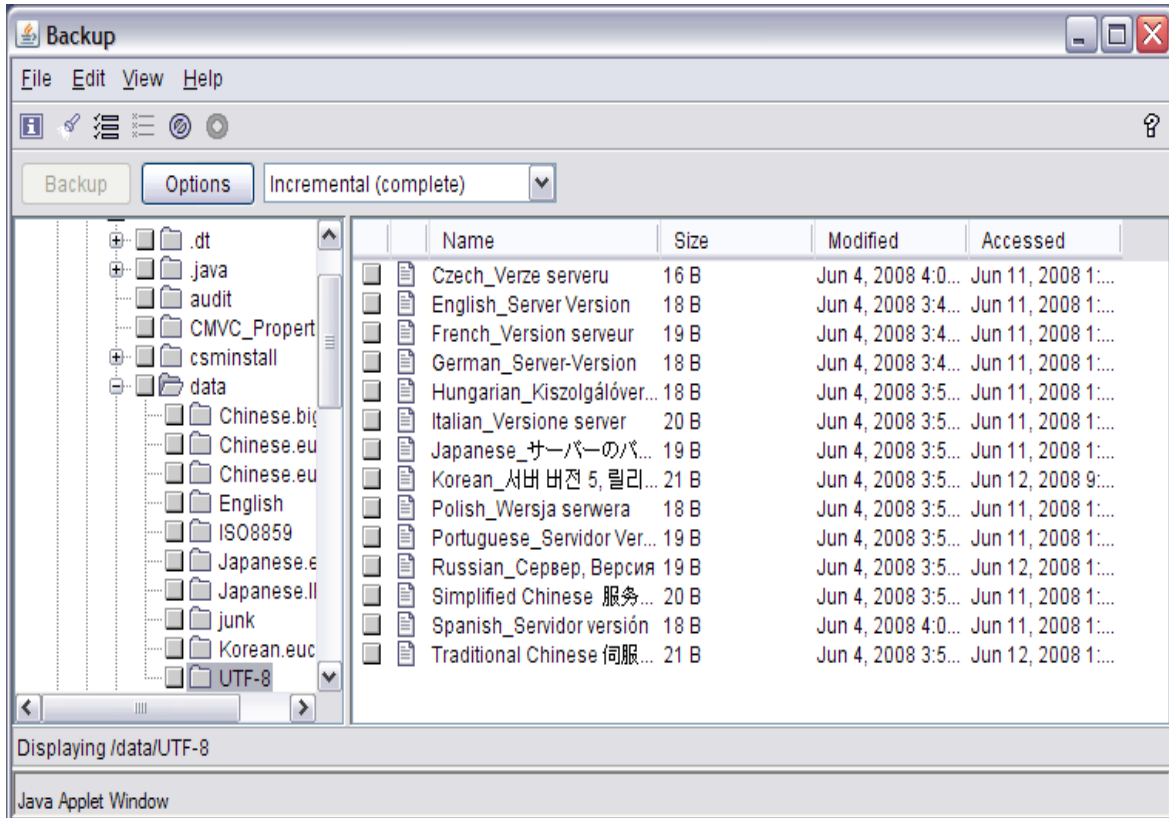
In the example below, the locale is set to English UTF-8 and the terminal encoding is set to UTF-8. The `ls` command correctly displays all fourteen UTF-8 encoded file names correctly, the TSM client messages are in English, and all of the selected files are backed up.



If the locale is changed to RU_RU.UTF-8, the TSM client messages are in Russian, all file names are still displayed correctly, and all of the selected files are backed up.



The Tivoli Storage Manager client GUI can also be used to backup and restore UTF-8 files. In the example below, the Tivoli Storage Manager GUI is running in the English UTF-8 locale and all fourteen UTF-8 file names are displayed correctly.



3. File Names Are in Different Single Byte Character Set Encodings and the File Names Are in Multiple Languages

There are sixteen separate ISO 8859 single byte character set encoding definitions, ISO 8859 -1 to ISO 8859-16. File names containing characters other than the basic ASCII seven bit Latin characters (A-Z, a-z, 0-9, etc.) may display differently in different single byte ISO 8859 locales. For example, the file with the name “file3φ” in the ISO 8859 -1 locale would display as “file3ϕ” when displayed in an ISO 8859-5 locale.

How Do I Know I'm in this Environment?

The most direct method to discover this environment is to query the locale with the command **locale**. If the locale is a single byte ISO 8859 locale and file names are in different languages, then you are probably in this environment. Not all file names display correctly on the command line and in the Tivoli Storage Manager client GUI and there is no support for Japanese, Chinese or Korean names when running in this environment.

Tivoli Storage Manager Client Concerns

All files can be backed up or restored by the Tivoli Storage Manager client with no fear of seeing the ANS4042E error message and skipping files during a backup. Requests to backup or restore a set of files using the wild card “*” specification will backup or restore all file s meeting the selection criteria.

The Tivoli Storage Manager client can run in any of the single byte character set ISO 8859 locales and backup and restore all files correctly . However, switching between the two different locales could cause users confusion as to which file is being accessed.

Restoration of files can be complicated in this environment because you may not be able to specify the name of the file to be restored if the locale and the terminal encoding are not the same as that of the file to be restored.

Files with single byte character set encoded names that cannot be entered or displayed correctly in the current locale can be backed up or restored by the TSM Backup -Archive command line client if a wild card "*" specification is used.

The screenshot shows a terminal window with the following output:

```

tsm> sel /data/ISO8859/*
Selective Backup function invoked.

Directory-->          4,096 /data/ISO8859 [Sent]
Normal File-->       22 /data/ISO8859/Polish_Wersja_serwera [Sent]
Normal File-->       23 /data/ISO8859/English_Server Version [Sent]
Normal File-->       17 /data/ISO8859/Czech_Verze_serveru [Sent]
Normal File-->       22 /data/ISO8859/Hungarian_Kiszolgaloverzio [Sent]
Normal File-->       19 /data/ISO8859/German_Server-Version [Sent]
Normal File-->       24 /data/ISO8859/Spanish_Servidor version [Sent]
Normal File-->       21 /data/ISO8859/French_Version_serveur [Sent]
Normal File-->       20 /data/ISO8859/Italian_Versions_server [Sent]
Normal File-->       23 /data/ISO8859/Russian_Àâãäå,²Öääöï [Sent]
Selective Backup processing of '/data/ISO8859/*' finished without failure.

Total number of objects inspected:      10
Total number of objects backed up:     10
Total number of objects updated:        0
Total number of objects rebound:       0
Total number of objects deleted:        0
Total number of objects expired:        0
Total number of objects failed:         0
Total number of bytes transferred:     479 B
Data transfer time:                     0.00 sec
Network data transfer rate:             5,439.22 KB/sec
Aggregate data transfer rate:           0.42 KB/sec
Objects compressed by:                  0%
Elapsed processing time:                00:00:01
tsm>
  
```

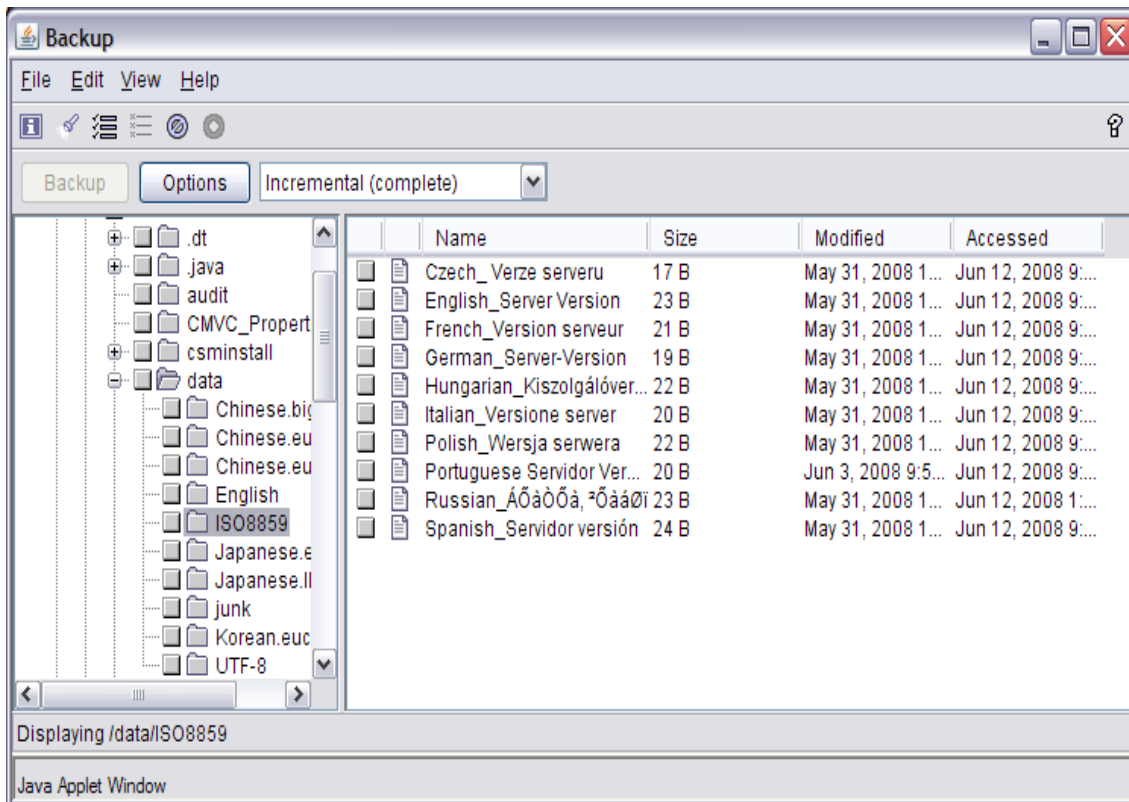
Overlaid on the terminal is a 'Terminal Encodings' dialog box with the following content:

Available encodings:		Encodings shown in menu:	
Description	Encoding	Description	Encoding
Baltic	ISO-8859-4	Central European	ISO-8859-2
Central European	IBM852	Chinese Simplified	GBK
Central European	ISO-8859-2	Chinese Simplified	GB2312
Chinese Simplified	GB2312	Chinese Traditional	EUC-TW
Chinese Simplified	GB18030	Chinese Traditional	BIG5
Chinese Simplified	GBK	Current Locale	ISO-8859-1
Chinese Traditional	BIG5	Cyrillic	ISO-8859-5
Chinese Traditional	EUC-TW	Japanese	EUC-JP
Current Locale	ISO-8859-1	Korean	EUC-KR
Cyrillic	ISO-8859-5	Unicode	UTF-8
Greek	ISO-8859-7		

The dialog box includes 'Add' and 'Remove' buttons at the bottom, and a 'Close' button in the bottom right corner.

The Tivoli Storage Manager client GUI running in a single byte character set locale, such as en_US.ISO8859-1, shows all of the file names, but file names containing characters defined in other locales may not be displayed correctly.

This is a display problem, not a functional problem. If one or more of the files with the incorrectly displayed names is selected to be backed up or restored, the backup or restore will process the file correctly.



4. File Names are in Different Encodings, Some of Which Are in a DBCS or UTF-8 Encoding

This environment is similar to the previous environment in that there are files with names in multiple languages and multiple character set encodings. The difference is that some of the file names are encoded in incompatible multiple byte character set encoding such as Japanese DBCS or UTF-8.

How Do I Know I'm in this Environment?

If you support two or more DBCS languages such as Japanese, Chinese or Korean, you are in this environment.

If you support a DBCS language and any other language besides English, you are in this environment.

If you encounter the ANS4042E error message when running a backup with the Tivoli Storage Manager client, you are in this environment.

If the Tivoli Storage Manager client GUI does not show some file names, you are in this environment.

It is not always easy to determine if you are in this environment because if you have DBCS or UTF-8 file names, the addition of just one file with a name that is incompatible with the DBCS or UTF-8 locale will put you in this environment.

If You Are on a Linux System, You Are Probably in this Environment

Linux systems normally default to UTF-8 file name. A Linux UTF-8 system may start out as a UTF-8 system, but it is very easy to create a file with a name that is incompatible with UTF-8 and put you in this environment. For instance:

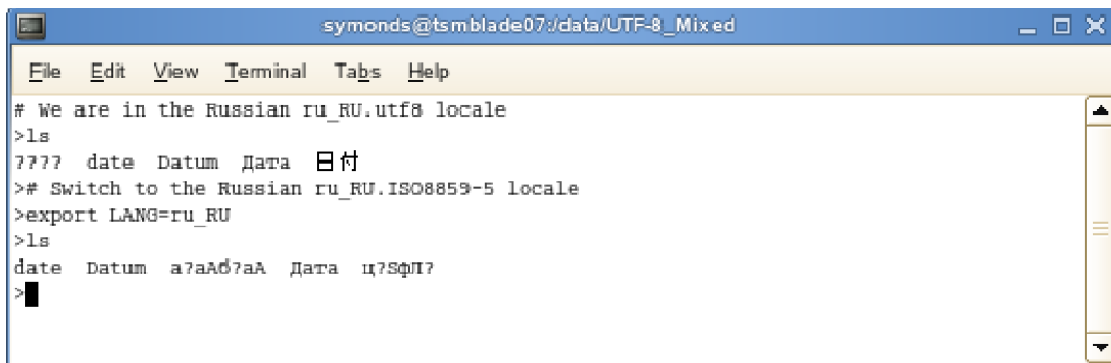
1. If the terminal emulator is changed to an encoding other than UTF-8 and a file is created with any characters other than the basic Latin characters (A-Z, a-z, 0-9, etc.), you are now in this environment.
2. If a file is copied from a remote non-UTF-8 system and the file name is composed of any characters other than the basic Latin characters, you are now in this environment.

An Environment Where the File Names Have Different Encodings Is Complicated and Difficult to Administer

In the example below, four files with names corresponding to the English, German, Russian and Japanese word for “date” were created with UTF -8 encoding, and an additional file named “Дата” was created with the Russian ISO8859-5 encoding. The addition of the Russian ISO -8859-5 file puts the system in a mixed locale environment where it is not possible to correctly display both the ISO 8859-5 encoded file and the UTF-8 encoded file in a single window.

The locale and terminal encoding are initially in the Russian UTF -8 locale and all of the UTF-8 files are displayed correctly, but the Russian ISO -8859-5 encoded file name is not displayed correctly (In this case it is displayed as “? ???”).

When the terminal encoding and the locale are changed to the Russian ISO -8859-5 encoding, the single byte ISO8859-1 characters in the English and German file names (“date” and “Datum”) and the Russian ISO8859-5 encoded file “Дата” are displayed correctly but the Russian and Japanese UTF-8 encoded file names are completely garbled.



```
symonds@tsmblade07:/data/UTF-8_Mixed
File Edit View Terminal Tabs Help
# We are in the Russian ru_RU.utf8 locale
>ls
???? date Datum Дата 日付
># Switch to the Russian ru_RU.ISO8859-5 locale
>export LANG=ru_RU
>ls
date Datum a?aaб?aa Дата ц?9фП?
>
```

Tivoli Storage Manager Client Concerns

Deployed correctly, the Tivoli Storage Manager Linux and UNIX Backup -Archive Client can be configured to properly protect all files regardless of their file name or the encoding used to represent the file name.

If you are running in this environment, you should always run your scheduled backups using a single byte character set locale so that no files are skipped due to the file name containing characters that are not defined in the current locale. When you restore files, you should attempt to run in the same locale as the name of the file or files to be restored. However, any file that has been backed up can always be restored regardless of the locale.

When running in multiple byte character set locale, such as the DBCS or UTF -8, files with names composed of characters that are not valid in the locale cannot be entered on the TSM client command line and they may be skipped when running a backup where a wild card "" specification is used. If files are skipped, the following error message will be written.*

ANS4042E Object name '/testData/en_US_files/file3?' contains one or more un-recognized characters and is not valid.

The screenshot shows a terminal window with the following output:

```
Selective Backup function invoked.

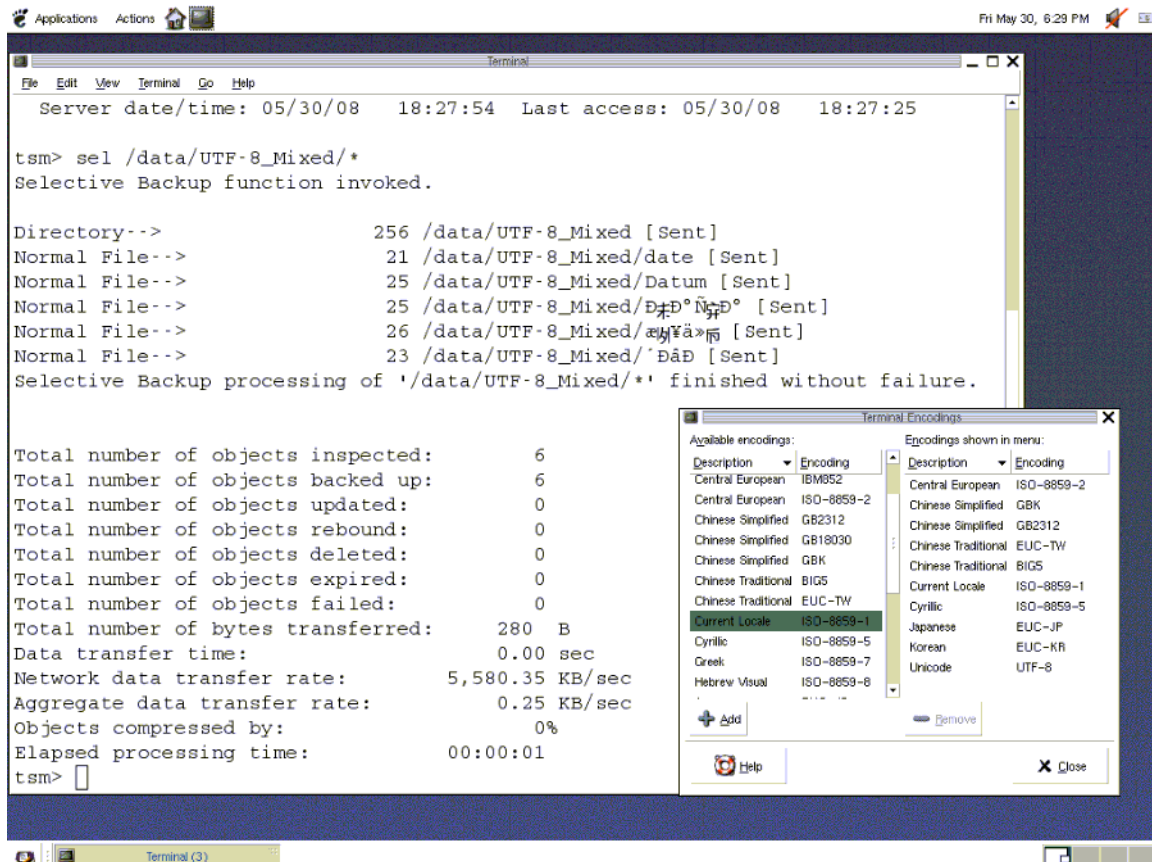
ANS1228E Sending of object '/data/UTF-8_Mixed/' failed
ANS4042E Object name '/data/UTF-8_Mixed/' contains one or more unrecognised
characters and is not valid.
Directory-->          256 /data/UTF-8_Mixed [Sent]
Normal File-->        21 /data/UTF-8_Mixed/date [Sent]
Normal File-->        25 /data/UTF-8_Mixed/Datum [Sent]
Normal File-->        25 /data/UTF-8_Mixed/дара [Sent]
Normal File-->        26 /data/UTF-8_Mixed/日付 [Sent]
ANS1804E Selective Backup processing of '/data/UTF-8_Mixed/*' finished with fail
ures.

Total number of objects inspected:      6
Total number of objects backed up:      5
Total number of objects updated:        0
Total number of objects rebound:        0
Total number of objects deleted:         0
Total number of objects expired:         0
Total number of objects failed:         1
Total number of bytes transferred:      225 B
Data transfer time:                      0.00 sec
Network data transfer rate:              5,938.55 KB/sec
Aggregate data transfer rate:            0.20 KB/sec
Objects compressed by:                   0%
Elapsed processing time:                 00:00:01
tsm>
```

Overlaid on the terminal is the 'Terminal Encodings' dialog box. It has two panes: 'Available encodings' and 'Encodings shown in menu'. The 'Available encodings' pane lists various encodings with their descriptions and encoding names. 'Unicode UTF-8' is highlighted. The 'Encodings shown in menu' pane lists encodings currently active in the terminal's menu, including 'Unicode UTF-8'.

It is possible to use the Tivoli Storage Manager client to backup all of the files in a mixed environment if the locale is set to a single byte locale such as the English en_US.ISO8859-1 locale. However, files whose names contain characters that are invalid in the chosen single byte locale will not be displayed correctly and can not be entered on the command line. File names will only be displayed correctly and can be entered on the Backup-Archive command-line client if the locale of the TSM client is the same as the character encoding of the file name and the terminal encoding is configured properly.

The Tivoli Storage Manager client include/exclude processing is also locale sensitive. Care must be taken to ensure that the include/exclude statements in the dsm.sys file match the encoding of the names of the files to be included or excluded. One way to check this is to **cat** the dsm.sys file while running in the locale corresponding to the files names to be included or excluded. If the cat display does not match the expected file names, the encoding in the dsm.sys file is incorrect.

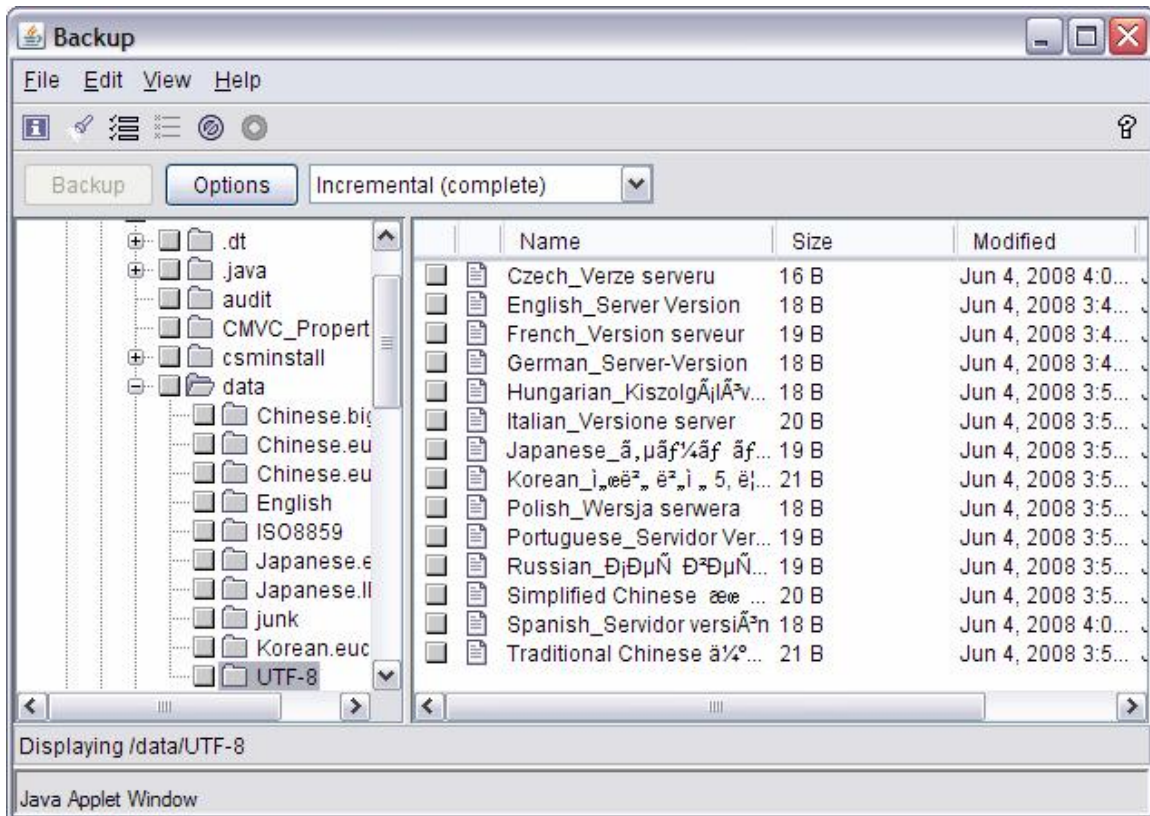


Files Can Always Be Backed Up or Restored, but Care Must Be Used in Entering or Displaying the File Names

The Tivoli Storage Manager client GUI will display all of the file names when running in a single byte character set locale such as ISO8859-1. However, file names containing characters that are not valid in the locale may not be displayed correctly. .

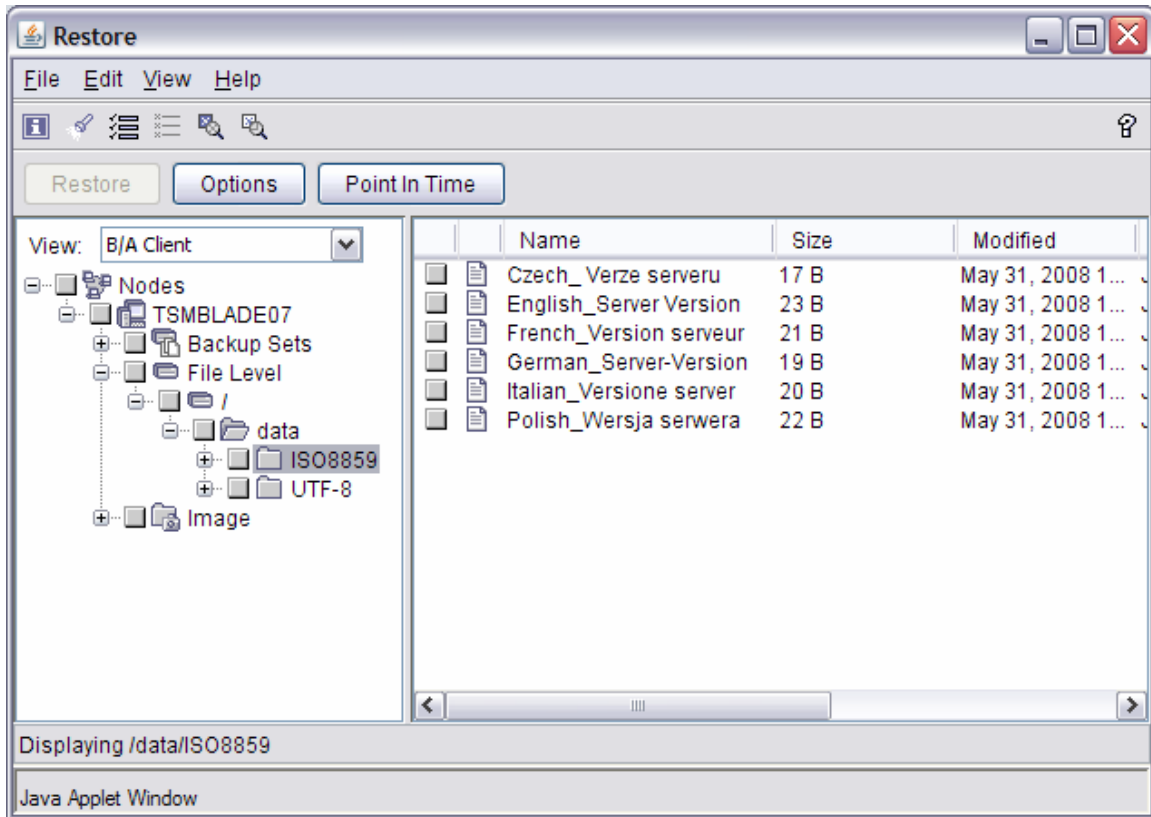
The example below shows the display when the Tivoli Storage Manager GUI is started in the English ISO 8859-1 locale and the directory containing the ISO 8859 files is displayed. All ten files are shown, but file names containing characters that are not defined in the current locale are not displayed correctly.

This is a display problem, not a functional problem. If one or more of the files with the incorrectly displayed names is selected to be backed up or restored, the backup or restore will process the selected files correctly.



If the Tivoli Storage Manager client GUI is started in a UTF -8 locale, all of the files with UTF -8 encoded names will be displayed correctly, but file names composed of characters that are not valid in the UTF -8 locale **will be completely skipped in the display!**

The directory in the display below actually has ten files, Czech, English, French, German, Hungarian, Italian, Polish, Portuguese, Russian and Spanish named files. The UTF -8 display skipped the Hungarian, Portuguese, Russian and Spanish named files because the file names contained characters that were invalid in the UTF -8 locale of the Tivoli Storage Manager GUI.



3. Migrating to a Different Character Set Encoding is Difficult

It is possible to convert an ISO 8859 or DBCS character set encoded environment to an all UTF-8 environment, but the conversion process is complex. Whenever possible, it is better to start with a clean UTF-8 only environment.

Converting files from an ISO8859 or DBCS character set encoding to a UTF-8 encoding is not a simple operation. The process requires several steps that have considerable implications on your environment

Renaming Files from One Character Set Encoding is a Two Step Operation

It is not possible to rename files from one character set encoding to another with a simple rename command. The rename command is running in a single locale and all of the inputs to that command are in that locale. Attempting to rename the ISO8859-1 file für to the UTF-8 equivalent would result in an error with no change to the file's name.

```
mv für für
```

```
mv: 'für' and 'für' are the same file
```

The simplest way to rename a file from one character set encoding to another is to first rename the offending file to an intermediate name using just the basic Latin characters. The basic Latin characters (A-Z, a-z, 0-9, etc.) compose the seven bit ASCII characters (the high order bit of the eight bit byte for these characters is always zero), and they have the same encoding in all supported locales. Since the basic Latin characters are valid in all locales, you can switch to the target locale and perform a second rename of the basic Latin name to the locale specific name.

The terminal's copy and paste function can be used to ease this operation. First, copy the initial name to the terminal's clipboard. Then, paste the file name into first operand on the first rename command. Next, switch the terminal's encoding to the target locale and paste the same name into the second operand on the second rename command. This reuse of the same copied name works because the terminal emulator treats the pasted character on the rename command as new input that is encoded in the current terminal encoding.

Another way to change a file's name to a different character set encoding is to use a program such as iconv to convert the name from one encoding to another. There is no official utility for performing such conversions; however, there is perl convmv utility available on the Internet that provides the ability to rename files from one locale to another:

```
convmv -f ISO8859-1 -t UTF-8 für
```

Renaming a File May Not Be Sufficient

Renaming the file may not be sufficient when converting to a UTF-8 environment. Once the file name has been changed, the enhanced display and backup and restore capabilities of UTF-8 named files is available. However, the data inside the file is still in its original encoding. This may

or not be a problem, depending on how the data in the file is processed. If the data is processed in a locale independent manner, then no change to the file data encoding is required. However, if the processing is dependent upon the locale, then the character set of the data in the file must also be changed.

For example, if a search program is used to look for an instance of a word in a file, the search program would fail if the character set encoding of the name was changed but not the data inside the file. In the example shown below, the program *search* takes two arguments, a file name and the string to search for.

```
search für groß
```

If this program is run in an ISO8859 locale, it would search for a different character string than if it were run in a UTF-8 locale. In this case, both the character set encoding of the file name and the character set encoding of the file data must be changed to UTF-8 for the search program to execute successfully.

The file data can be converted from its original character set encoding to UTF-8 character set encoding with the *iconv* utility as long as all of the data in the file is character data. If the file contains a mixture of character data and binary data, there is no general purpose conversion utility. Files containing a mixture of character data and binary data can only be converted by programs that understand the mixed format of the file.

In the case of the file "für", which consists of character data, the *iconv* utility can be used to convert the data to the UTF-8 character set encoding.

```
iconv -f ISO8859-1 -t UTF-8 für
```

The Global Environment Must Be Changed for All Users

Once the ISO8859-1 file "für" has been renamed to a UTF-8 file, it is a different file. No Tivoli Storage Manager backups exist for this file, and the file name can not be displayed or entered properly while the terminal is in the original ISO8859-1 encoding. The terminal encoding must be changed for every user of the original file to make this rename operation transparent.

Changing a user's default terminal encoding is a global change that affects all of the files accessed by the user. Files whose names are in the same character set encoding as the terminal can be displayed, but files whose names are not in the same character set encoding as the terminal emulator can no longer be displayed correctly unless the terminal encoding is changed to match that specific file name encoding.

4. Character Set Encoding Standards

The ISO 8859 Variations

ISO 8859-1 is a standard character encoding of the Latin alphabet. Byte values 0 to 31 (decimal) are not assigned in this standard, byte values 32 to 127 correspond to standard Latin characters (i.e., the English alphabet and numeric digits), and byte values 128 to 159 are also not assigned.

This means that in a single byte ISO 8859 locale, any byte with a value from 32 to 127 or from 160 to 255 can be displayed as a character in an ISO 8859 locale. Characters outside of that range cannot be displayed. Attempts to display these characters will normally be replaced with the '?' character.

It should be obvious that 256 characters are inadequate to encode all of the languages covered in the ISO 8859 standard: English, French, German, Polish, Croatian, Latvian, Russian, Greek, Arabic, Hebrew, etc.

Because a single standard for the 8 bit ASCII characters cannot encode the characters for all of these languages, there are 16 variations of this encoding standard, ISO 8859-1 to ISO 8859-16. Each of these 16 variants has different definitions for the characters corresponding to 160 to 255. The existence of 16 different variations of the ISO 8859 standard means that the same binary encoded file name may appear different in different locales. For instance, the character 'x'A2' is the character 'ç' in ISO 8859-1; however, it is the character 'Ђ' in ISO 8859-5 and 'ą' is ISO 8859-16.

Unicode and UTF-8

The Unicode standard was defined to resolve the complexities of multiple code pages. The Unicode standard is a character set coding system designed to support all of the modern languages and many of the ancient languages in a single encoding scheme. The Unicode standard is continually evolving as new characters and code points are added.

The original Unicode standard defined a 16-bit encoding for approximately 16,000 different characters. This original standard has been extended to support more than 16,000 characters and to support systems such as UNIX that define strings as sequences of eight bit bytes.

The latest Unicode standard defines three different encodings for each character in the standard:

- The UTF-8 encoding for each character is composed of one or more eight bit bytes. UTF-8 is commonly found on UNIX systems.
- The UTF-16 encoding for each character is composed of one or more 16 bit values. UTF-16 is used on Microsoft Windows systems.
- The UTF-32 encoding for each character is composed of one 32 bit value for each character.

All three of these encodings are equivalent in that any one of the three encodings can be converted into either of the other two using a computer algorithm. There is no need for a table lookup to convert from one to another.

For more information on the Unicode standard, refer to the [Unicode Standard](#).

For an excellent discussion on the three different encodings of Unicode, refer to [“Forms of Unicode”](#).

Extended Unix Code

Extended Unix Code (EUC) is a multiple byte character encoding system used primarily for Japanese, Korean and simplified Chinese. This encoding scheme allows the easy mixing of 7 -bit ASCII and multiple byte 8-bit Japanese characters.