Tivoli software

IBM

Tivoli Storage Manager and Localization

Localization Considerations Using the Linux and UNIX Backup - Archive Client

By Peter Symonds and J.P. (Jim) Smith

Version 1.0

Copyright Notice

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

Tivoli Storage Manager and Localization

Localization Considerations Using the Linux and UNIX Backup - Archive Client

1. Introduction: The Localization Problem and Storage Management

Who in the world would have a computer file named "*fյle.txt*"? Or how about a file named *"file.txt"*? The second letter in the second name is a Latin small letter "l" with an acute accent. The first name is actually a rendering of the same name in the Windows 1252 code page. You don't have to be an expert in character encoding to see that the two names are different. What you do need to understand is that each file name is recorded in the file system as a string of bits. More importantly, you want to be assured that if you needed to recover a file, your storage management product would correctly restore the name as the same exact sequence of bits.

This paper discusses the Tivoli Storage Manager for Linux and UNIX Backup -Archive Client in the context of localization and addresses the three central issues that should most concern a storage administrator in this context: backup, recovery, and error reporting. Deployed correctly, the Backup-Archive Client will provide desired protection. ***The Tivoli Storage Manager Linux and UNIX Backup-Archive Client Version 5.4 and above can be configured to properly protect all files regardless of their file name or the encoding used to represent the file name.*** This paper is to help you understand localization issues and the best way to achieve your desired results with Tivoli Storage Manager.

Do I Really Need to Worry about Localization?

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 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. I f 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.

If you have the Tivoli Storage Manager Backup -Archive Client deployed and have encountered this error during a backup operation, you should definitely review this document:

```
ANS1228E Sending of object /mixed/f£ failed
ANS4042E Object name /mixed/file£ contains one or more unrecognized
characters and is not valid
```

Localization 101

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 frequently involves message translation, character and font representation, keyboard usage and date, time and number formats.

Message Translation

Software products that have been internationalized, such as the Tivoli Storage Manager Client, provide message catalogs that have been translated into multiple languages. Users of internationalized products have the option to specify the language used during product interaction. The locale environment variables are used to specify the language to be used.

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

Character Representation

As stated in the introduction, 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 bina ry 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. If your file names are composed exclusively of these characters, the file names will always appear the same regardless of the locale. However, characters that have a binary encoding that is greater than 0x80 have different mappings in 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.

This document discusses files with names defined according to the following standards.

1. The single byte ASCII standards ISO 8859-1 through ISC8859-16
2. The Japanese double byte character set (DBCS) standard ja_JP.eucJP.
3. The Korean double byte character set standard ko_KR.
4. The Chinese double byte character set standards zh_CN and zh_TW
5. The Unicode multiple byte standard for UTF-8.

For more information about locale see Appendix B.

2. About the Organization of this Document

This document is intended to provide guidance when running the Tivoli Storage Manager Client in the four common character encoding environments. For each of these environments, we will discuss basic localization considerations for using the Tivoli Storage Manager Backup -Archive Client:

1. ***Summary*** will give a quick summary of the Backup -Archive client considerations
2. ***How do I know if I'm in this environment*** will discuss how you can determine which one of the basic character encoding environments is running on your computers.
3. ***Backup processing*** will discuss considerations for processing backup and archive operations, both at the logical volume level and a single-file or single-directory level.
4. ***Restore processing*** will discuss considerations for querying files stored on the Tivoli Storage Manager Server and restoring and retrieving the files using the different interfaces provided by the Tivoli Storage Manager Backup-Archive client.
5. ***Include-exclude processing*** will discuss considerations for properly including or excluding file names from backup processing
6. ***Messages*** will discuss considerations for properly viewing messages produced by the Tivoli Storage Manager Backup-Archive client
7. ***Examples*** are also included to illustrate the points above in each section.

For more information about the file system configuration established to demonstrate these four environments, please see Appendix A.

3. What Is My Character Encoding?

Most UNIX file systems are insensitive to the character encoding of a file name. File names are arbitrary strings of binary bytes that can vary in value between 0x01 to 0xff with one exception; no file name can contain an imbedded directory delimiter, the '/'character.

The character encoding of a file name is an interpretation of the binary bytes making up the file name that defines how the file name is displayed on a graphics device. The character e ncoding of a file name is determined by the application used to create the file name. If the shell is used to create the file using a command such as **touch**, the encoding of the file name is determined by the locale of the shell. For example, issuing touch f£ in the ISO8857-1 locale will create a file name with the binary encoding 0x66A3. The same command in a UTF -8 locale will create a file name with the binary encoding 0x66C2A3.

If the file is created by an application program, such as the file manager, the application program will determine the encoding of the file created. In Linux systems such as SUSE 10, the file manager application typically creates the file name with a UTF -8 encoding.

For more information about locales and UNIX shell see Appendix F.

4. Scenario 1: ISO 8859-1 and Other Single Byte Character Encodings

A single byte character set encoding locale, such as the ISO 8859-1 encoding of the Latin alphabet provides the simplest character encoding environment. All file names are assumed to have the same single-byte character set encoding and the locale matches the file name encoding. In such an environment the file names are displayed correctly and the characters can easily be entered by the keyboard.

This environment is fully supported by the Tivoli Storage Manager client.

Note that ISO 8859 has other encodings which are implicitly covered in this section, for example ISO 8859-2 encoding for Central and Eastern European Languages and ISO 8859 -5 for Slavic languages based on the Cyrillic alphabet.

Summary

In environments that use single byte encodings exclusively, there should be no localization concerns using the Tivoli Storage Manager Backup-Archive client. All aspects of backup, recovery and messaging should behave as expected.

How do I know I'm in this environment?

The **locale** command will return the name of a single byte character set locale such as en_US, de_DE, or one of the ISO 8859 variants. Commands that deal with file names, such as **ls** will correctly display all file names.

Language	Locale
English	en_US
Czech	cs_CZ
French	fr_FR
German	de_DE
Hungarian	hu_HU
Italian	it_IT
Polish	pl_PL
Portuguese	pt_BR
Russian	ru_RU
Spanish	es_ES

Backup processing

There are no special considerations for backup processing in this environment. File names can always be entered on the Backup-Archive command-line client and all files meeting the selection criteria will be backed up.

Note that there are sixteen separate ISO 8859 locales. File names containing special characters, such as '¢' may have different encodings in the different ISO 8859 locales. For example, the file with the name “file3¢” in the ISO 8859-1 locale would display as “file3Ђ” when displayed in an ISO 8859-5 locale. The Tivoli Storage Manager client will backup and restore this file correctly in either locale because the actual binary encoding of existing file names does not change if the locale is changed. However switching between the two different locales could cause users confusion as to which file is being accessed.

Restore processing

There are no special considerations for restore processing in this environment. File names can always be entered on the Tivoli Storage Manager Client command line and all files meeting the selection criteria will be restored up.

Include-exclude processing

There are no special considerations for include-exclude processing in this environment.

Tivoli Storage Manager Client include-exclude processing is fully supported in this environment. All file names can be specified and they will be processed correctly.

The following two dsm.sys file statements illustrate how the Tivoli Storage Manager Client include/exclude processing can be configured.

The Tivoli Storage Manager Client processes the statements sequentially. The first statement “exclude /fileData/* excludes all files in the directory “/fileData”. The second statement “include /fileData/file* overrides the exclude statement and allows all files beginning with the characters “file” in the directory /fileData to be included in the backup processing.

```
exclude /fileData/*
include /fileData/file*
```

Messages

There are no special considerations for displaying messages produced by the Tivoli Storage Manager Backup-Archive client in this environment.

The Tivoli Storage Manager client automatically detects the language specified by the system locale and displays 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/locale combination, 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.

Examples

The following examples show the Tivoli Storage Manager Backup-Archive Client in an ISO 8859-1 environment.

*Example 1: The **ls** command listing of the file names*

```
>ls -l /testData/en_US_files
total 16
-rw-r--r--  1 symonds users 21 2007-10-02 10:07 file1
-rw-r--r--  1 symonds users 22 2007-10-02 10:07 file2
-rw-r--r--  1 symonds users 16 2007-10-02 10:07 file3¢
-rw-r--r--  1 symonds users 13 2007-10-02 10:07 file4£
```

Example 2: Include-exclude statements

The include-exclude statements in the dsm.sys configuration file are also in the ISO 8859-1 locale.

```
exclude /testData/en_US_files/*
include /testData/en_US_files/file1
include /testData/en_US_files/file2
include /testData/en_US_files/file3¢
include /testData/en_US_files/file4£
```

Example 3: Execution of a selective backup operation

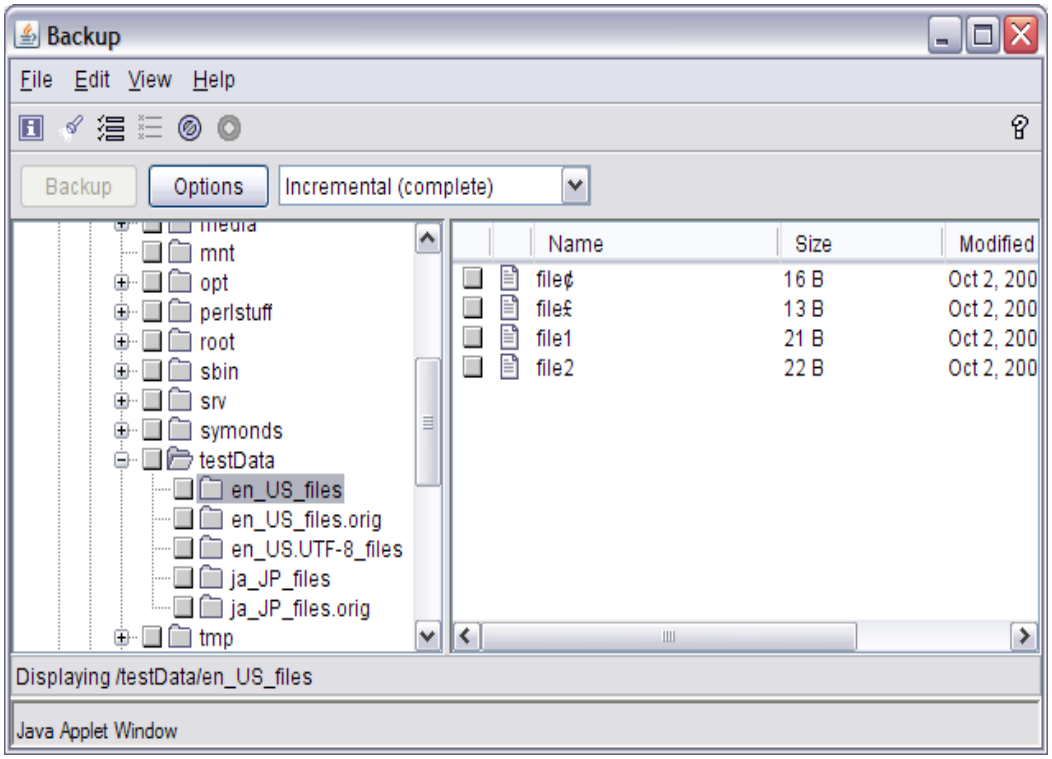
```
>dsmc  sel /testData/en_US_files/*
Selective Backup function invoked.
Directory-->             144 /testData/en_US_files [Sent]
Directory-->             120 /testData [Sent]
Normal File-->            21 /testData/en_US_files/file1 [Sent]
Normal File-->            22 /testData/en_US_files/file2 [Sent]
Normal File-->            16 /testData/en_US_files/file3¢ [Sent]
Normal File-->            13 /testData/en_US_files/file4£ [Sent]
Selective Backup processing of '/testData/en_US_files/*' finished without failure
```

Example 4: Execution of a restore operation.

In this single byte environment all of the files can be restored and the names of files to be restored can always be explicitly specified.

```
>dsmc rest /testData/en_US_files/file¢
Restoring              16 /testData/en_US_files/file¢ [Done].
```

The Tivoli Storage manager GUI displays all single byte en_US encoded files correctly when the dsmcad is started in the en_US single byte locale.


Backup
File Edit View Help
Backup
Options
Incremental (complete)
media
mnt
opt
perlstuff
root
sbin
srv
symonds
testData
en_US_files
en_US_files.orig
en_US.UTF-8_files
ja_JP_files
ja_JP_files.orig
tmp
Name
Size
Modified
file¢
16 B
Oct 2, 200
file£
13 B
Oct 2, 200
file1
21 B
Oct 2, 200
file2
22 B
Oct 2, 200
Displaying /testData/en_US_files
Java Applet Window

5. Scenario 2: Double- and Multi-byte Character Set Encodings

An environment with double byte character set encoding , such the Japanese ja_JP.eucJP locale or the Chinese zh_CN locale, provides a more complicated environment. The characters which comprise the file names can consist of one or more bytes .

In this example, the files were created in the ja_JP.eucJP locale. All file names have the same ja_JP.eucJP encoding, and the locale matches the file name encoding. In such a n environment, all of the file names are displayed correctly and the characters ca n easily be entered by the keyboard.

Summary

In environments that use double- and multi-byte encodings, there are minor localization concerns when entering file specifications for backup and restore operations on the command -line client. All other aspects of backup, recovery and messaging should behave as expected.

How do I know I'm in this environment?

The **locale** command will return the name of a double byte character set locale such as or ja_JP.eucJP. Commands that deal with file names, such a s **ls** will correctly display all file names .

Language	Locale
Simplified Chinese	zh_CN
Traditional Chinese	zh_TW
Japanese	ja_JP
Korean	ko_KR

Backup processing

Tivoli Storage Manager Client backup processing is supported with some restrictions in this environment. All files meeting the selection criteria will be backed up when running in a consistent multiple byte character set environment.

File names can always be entered on the dsmc command line if "editor no" is specified in the dsm.sys configuration file. If "editor no" is not specified in the dsm.sys configuration file, Chinese, Japanese or Korean pictorial characters can not be entered.

Restore processing

Tivoli Storage Manager Client restore processing in this environment is supported with the sa me restrictions as backup processing; "editor no" must be specified in the dsm.sys configuration file to enter file names containing double byte character set encoded characters. If "editor no" is

specified, file names can always be entered on the Tivoli S torage Manager Client command line , and all files meeting the selection criteria will be restored .

Include-exclude processing

Tivoli Storage Manager Client include -exclude processing is fully supported in this environment. All file names can be specifie d and they will be processed correctly .

Messages

The Tivoli Storage Manager Client simplified Chinese, traditional Chinese, Japanese and Korean message catalogs are fully supported this environment .

Examples

The following examples show the Tivoli Storag e Manager Client in a ja_JP.eucJP environment.

*Example 1: The **ls** command listing of the test file names*

```
>ls -l /testData/ja_JP_files/*
-rw-r--r--  1 symonds users 19 2007-10-02 10:08 file1
-rw-r--r--  1 symonds users 24 2007-10-02 10:08 file2
-rw-r--r--  1 symonds users 26 2007-10-02 10:08 file3無
-rw-r--r--  1 symonds users 25 2007-10-02 10:08 file4右
```

Example 2: Editor no and Include-Exclude statements

The include-exclude statements in the dsm.sys configuration file in this example are also in the double byte ja_JP.eucJP character set.

Notice the statement ***editor no***. This statement is required when running the Tivoli Storage Manager client in a double byte character set locale such as ja_JP.eucJP. If this statement is not added to the dsm.sys file, single b yte ascii characters such as 'f' can be entered on the Tivoli Storage Manager client command line, but double byte ch aracters such as the Japanese character '右' will not be processed correctly.

```
editor no
exclude /testData/ja_JP_files/*
include /testData/testData/ja_JP_files /file1
include /testData/testData/ja_JP_files /file2
include /testData/testData/ja_JP_files /file3無
include /testData/testData/ja_JP_files /file4右
```

Example 3: Execution of a selective backup operation

```
>dsmc sel /testData/ja_JP_files/*
Selective Backup function invoked.
Directory-->         144 /testData/ja_JP_files [Sent]
Normal File-->        19 /testData/ja_JP_files/file1 [Sent]
Normal File-->        24 /testData/ja_JP_files/file2 [Sent]
```

```
Normal File-->         26 /testData/ja_JP_files/file3無 [Sent]
Normal File-->         25 /testData/ja_JP_files/file4右 [Sent]
Selective Backup processing of '/testData/ja_JP_files/*'  finished without failure.
```

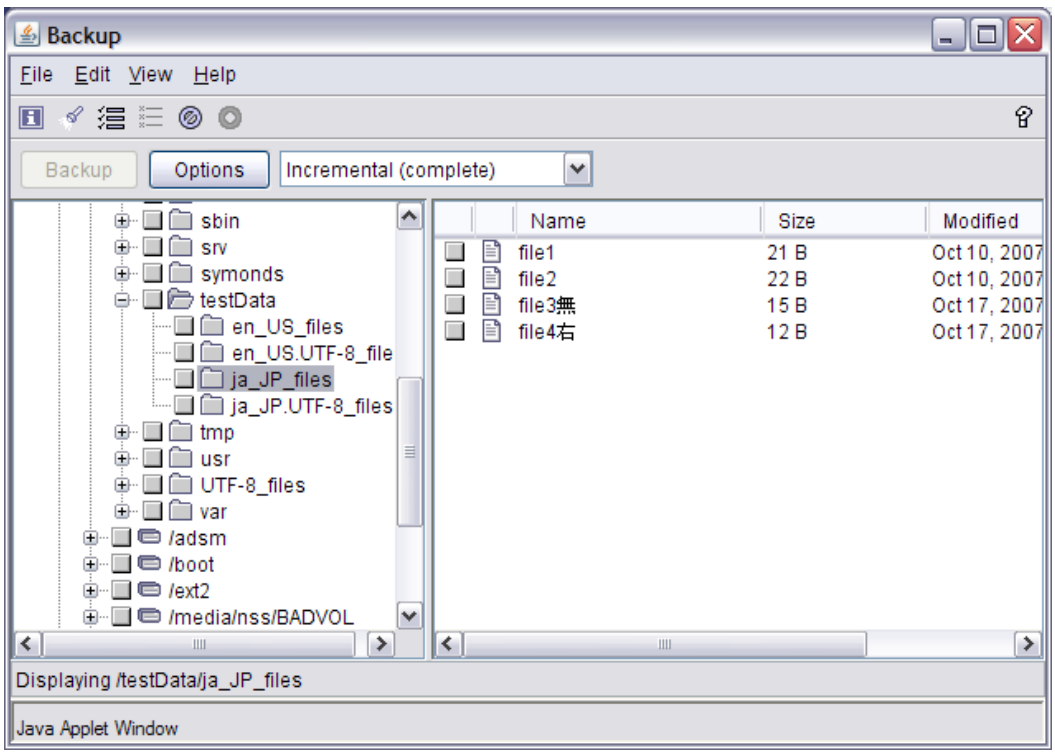
Example 4: Execution of a restore operation.

In a Tivoli Storage Manager Backup-Archive Client supported DBCS environment such as ja_JP.eucJP, the names of the files to be restored can be explicitly specified if “editor no” is specified in the dsm.sys file.

```
>dsmc rest /testData/ja_JP_files/file4右
Restoring              26 /testData/ja_JP_files/file4右 [Done]
```

Example 5: Tivoli Storage Manager GUI

The Tivoli Storage Manager GUI displays all file names correctly when the dsmcad is started in the same ja_JP.eucJP double byte character set locale as t he encoding of the file names.



6. Scenario 3: UTF-8 Encoding

An environment with UTF-8 character set encoding is similar to the double byte character environment described in the previous scenario; the characters which comprise the file names can consist of one or more bytes. However UTF-8 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.

In this scenario, all file names have the same UTF-8 encoding and a UTF-8 locale has been set. In such an environment, all of the file names are displayed correctly and the characters can easily be entered by the keyboard.

Summary

In environments that use UTF-8 encodings exclusively, there are minor localization concerns when entering file specifications for backup and restore operations on the command-line client. Messages produced by the Tivoli Storage Manager Backup-Archive client will not always be readable when the program is executed in a UTF-8 locale. With all other aspects of backup, recovery should behave as expected.

How do I know I'm in this environment?

The **locale** command will return the name of locale with ".UTF-8" appended. Some examples:

Language	**Locale**
English	en_US.UTF-8
Czech	cs_CZ.UTF-8
Japanese	ja_JP.UTF-8
Korean	ko_KR.UTF-8

The **locale** command will return the name of a multiple byte character set locale such as en_US.UTF-8, de_DE.UTF-8 or ja_JP.UTF-8. Commands that deal with file names, such as **ls,** will correctly display all file names.

Backup processing

The en_US.UTF-8 locale can always be used for file backup and restore processing because UTF-8 encoding is a "universal" character set encoding. Every UTF-8 encoded character has a unique UTF-8 encoding that is independent of the country-specific locale. When running in the en_US.UTF-8 locale, not only are all English characters processed correctly by the Tivoli Storage Manager client, but Chinese, Japanese, Korean and all other UTF-8 characters are also processed correctly. The same would be true when running in a de_DE.UTF-8, ja_JP.UTF-8 or any other UTF-8 locale.

Tivoli Storage Manager Client backup processing is supported with some restrictions in this environment. All files meeting the selection criteria will be backed up when running in a consistent UTF-8 character set environment.

File names can always be entered on the Tivoli Storage Manager Client command line if "**editor no**" is specified in the dsm.sys configuration file. If "**editor no**" is not specified in the dsm.sys configuration file, multiple byte characters such as '¢' (or any Chinese, Japanese or Korean pictorial character) can not be entered.

Restore processing

Tivoli Storage Manager Client restore processing is supported with the same restrictions on entering file names containing multiple byte character set encoded characters discussed in the above section on "Backup processing".

Include-exclude processing

Tivoli Storage Manager Client include-exclude processing is fully supported in this environment. All file names can be specified, and they will be processed correctly.

Messages

The Tivoli Storage Manager Client does not provide any UTF-8 message catalogs. When running in a UTF-8 locale, the equivalent Tivoli Storage Manager Client will choose the equivalent single byte character set or DBCS message catalog. If no equivalent Tivoli Storage Manager Client message catalog is found, the English (United States) language pack will be used.

However, if the equivalent locale is a DBCS language, such as Japane se, the Tivoli Storage Manager client DBCS message text will be displayed incorrectly in this locale because DBCS character encoding and UTF-8 character encoding are incompatible. For this reason, the en_US.UTF-8 locale should always be used to provide valid Tivoli Storage Manager Client single byte character set messages when processing file names encoded in a Japanese, Chinese or Korean UTF-8 character set encoding. The en_US.UTF-8 locale is used because messages in the English message catalog are displayed correctly in all locales.

Tivoli Storage Manager Client messages are supported with restrictions that are locale-dependent.

UTF-8 European locales such as de_DE.UTF-8

When running in a European UTF-8 locale such as de_DE.UTF-8, the corresponding Tivoli Storage Manager Client single byte character set message catalog will be used.

In this UTF-8 locale, all file names will be displayed correctly, but message text containing multiple byte characters such the German character 'ä' may not be displaye d correctly.

UTF-8 Asian locales such as ja_JP.UTF-8

It is not advisable to run the Tivoli Storage Manager Client in the Japanese, Chinese or Korean UTF-8 locales. Instead switch to a UTF-8 locale that has a corresponding Tivoli Storage Manager Client provided single byte character set message catalog such as en_US.UTF-8.

Examples

The following examples show the Tivoli Storage Manager Backup-Archive client in a UTF-8 environment.

*Example 1: The **ls** command listing of the file names*

Notice that the UTF-8 file names (file1, file2, file3¢, file4 £) look similar to the previous single byte ISO 8859 locale. However, because the locale is now a multiple byte UTF-8 locale, the binary encoding of the file names is different. Characters in the range of hex 01 to he x 7f are the same in both environments. However, the characters '¢' and '£' have different encodings in the UTF-8 locale; in this UTF-8 locale, they now take up two bytes.

```
>ls -l /testData/en_US.UTF-8_files
total 16
-rw-r--r--  1 symonds users 19 2007-10-02 10:08 file1
-rw-r--r--  1 symonds users 24 2007-10-02 10:08 file2
-rw-r--r--  1 symonds users 26 2007-10-02 10:08 file3¢
-rw-r--r--  1 symonds users 25 2007-10-02 10:08 file4£
-rw-r--r--  1 symonds users 21 2007-10-02 10:08 file5Œ

>ls -l /testData/ja_JP.UTF-8_files
total 16
-rw-r--r--  1 symonds users 21 2007-10-17 15:08 file1
-rw-r--r--  1 symonds users 22 2007-10-17 15:08 file2
-rw-r--r--  1 symonds users 29 2007-10-17 14:53 file3無
-rw-r--r--  1 symonds users 29 2007-10-17 14:54 file4右
```

Example 2: Include-Exclude statements

The include-exclude statements in the dsm.sys configuration file in this example are also in the multiple byte en_US.UTF-8 character set.

Notice the additional statement ***editor no***. This statement is required when running the Tivoli Storage Manager client in a UTF-8 multiple byte character locale or a Tivoli St orage Manager client supported multiple byte cha racter set such as ja_JP.eucJP If this statement is not added to the dsm.sys file, single byte characters such as 'f' can be entered on the Tivoli Storage Manager client command line, but multiple byte UTF-8 characters such as '¢', and Japanese, Chinese or Korean characters such as '語' will not be processed correctly.

```
editor no
exclude /testData/en_US.UTF-8_files/*
include /testData/en_US.UTF-8_files/file1
include /testData/en_US.UTF-8_files/file2
include /testData/en_US.UTF-8_files/file3¢
include /testData/en_US.UTF-8_files/file4£
include /testData/en_US.UTF-8_files/file5Œ
exclude /testData/ja_JP.UTF-8_files/*
include /testData/testData/ja_JP.UTF-8_files /file1
include /testData/testData/ja_JP.UTF-8_files /file2
include /testData/testData/ja_JP.UTF-8_files /file3無
include /testData/testData/ja_JP.UTF-8_files /file4右
```

Example 3: Execution of a selective backup operation of both Japanese and English UTF-8 encoded file names

```
>dsmc sel /testData/en_US.UTF-8_files/* /testData/ja_JP.UTF-8_files/*
Selective Backup function invoked.
Directory-->                144 /testData/ja_JP.UTF-8_files [Sent]
Normal File-->               21 /testData/ja_JP.UTF-8_files/file1 [Sent]
Normal File-->               22 /testData/ja_JP.UTF-8_files/file2 [Sent]
Normal File-->               29 /testData/ja_JP.UTF-8_files/file3無 [Sent]
Normal File-->               29 /testData/ja_JP.UTF-8_files/file4右 [Sent]
Selective Backup processing of '/testData/ja_JP.UTF-8_files/*' finished without
failure.


Directory-->                168 /testData/en_US.UTF-8_files [Sent]
Normal File-->               19 /testData/en_US.UTF-8_files/file1 [Sent]
Normal File-->               24 /testData/en_US.UTF-8_files/file2 [Sent]
Normal File-->               26 /testData/en_US.UTF-8_files/file¢ [Sent]
Normal File-->               25 /testData/en_US.UTF-8_files/file£ [Sent]
Normal File-->               21 /testData/en_US.UTF-8_files/file€ [Sent]
Selective Backup processing of '/testData/en_US.UTF-8_files/*' finished without
failure.
```

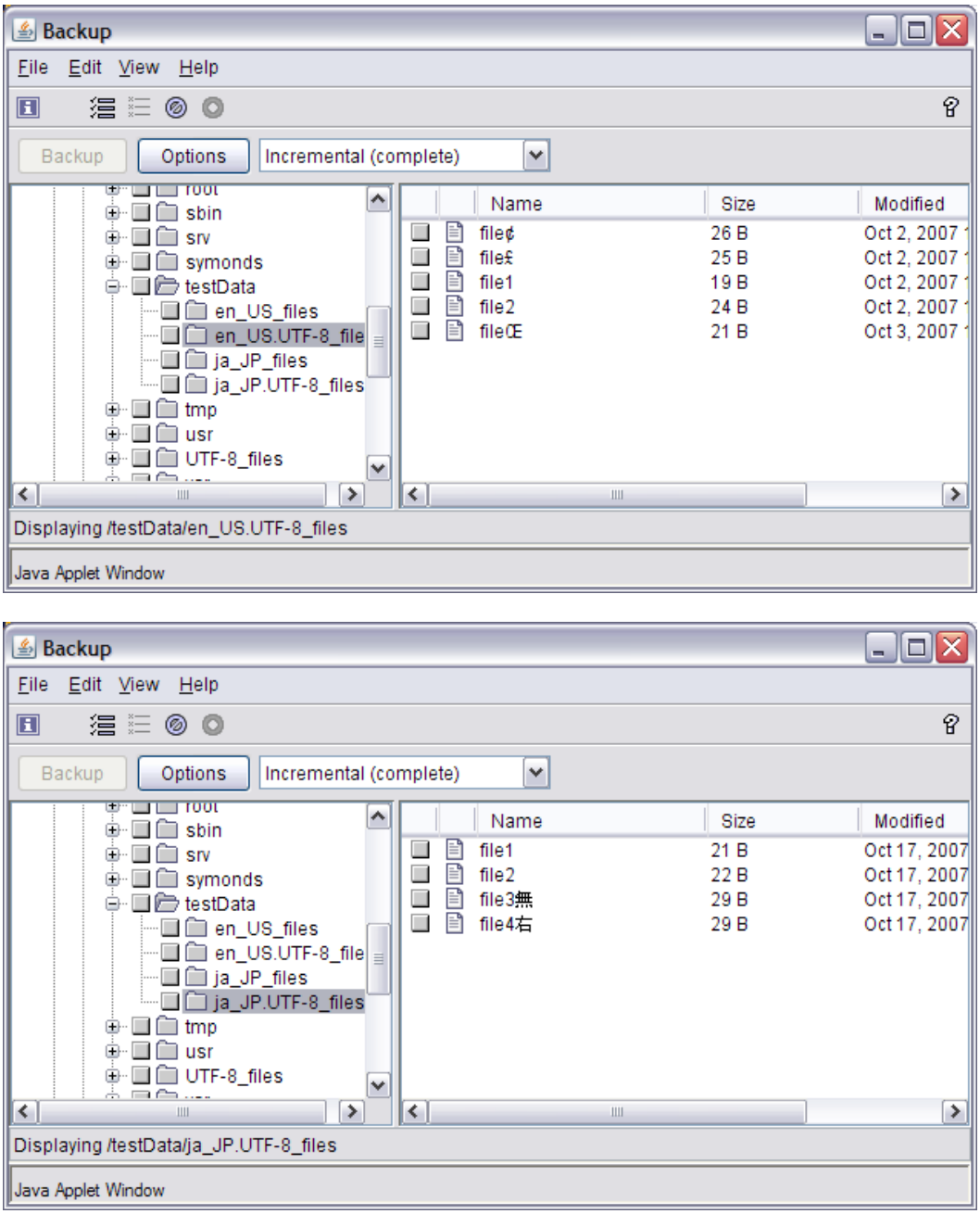
Example 4: Execution of a restore operation.

In a multiple byte UTF-8 environment, the names of the files to be restored can be explicitly specified if ***editor no*** is specified in the dsm.sys file.

```
>dsmc rest /testData/en_US.UTF-8_files/file3¢
Restoring              26 /testData/en_US.UTF-8_files/file3¢ [Done]
```

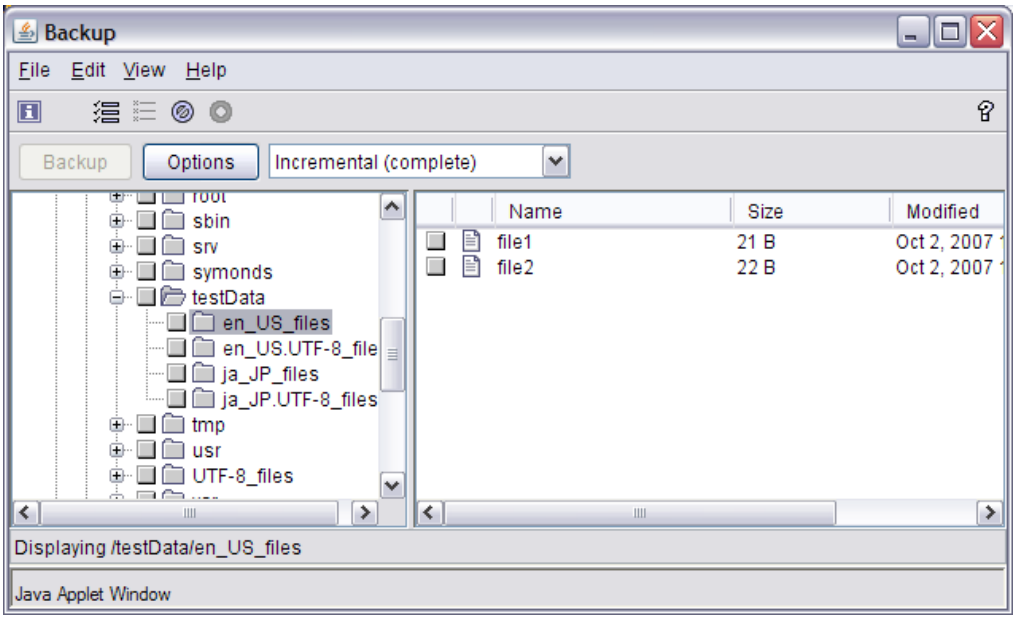
Example 5: Tivoli Storage Manager GUI

The Tivoli Storage Manager GUI displays all UTF-8 file names correctly when the dsmcad is started in a UTF-8 locale. In this example, the dsmcad was started in en_US.UTF-8. The examples below show both English UTF-8 files and UTF-8 Japanese files:



Example 6: Tivoli Storage Manager GUI and files names not encoded in UTF-8

Note that files containing invalid characters or multiple byte encoded characters that are not encoded in UTF-8 are not shown in the display.


Backup
File Edit View Help
Backup
Options
Incremental (complete)
sbin
srv
symonds
testData
en_US_files
en_US.UTF-8_file
ja_JP_files
ja_JP.UTF-8_files
tmp
usr
UTF-8_files
Name
Size
Modified
file1
21 B
Oct 2, 2007
file2
22 B
Oct 2, 2007
Displaying /testData/en_US_files
Java Applet Window

7. Scenario 4a: Mixed Encodings – Executing in a Multiple-byte Locale

The operating system allows for file names with different encodings in the same file sy stem or directory. If your environment does not standardize a single locale for creation of all files and directories (for example, if there are mixed UTF-8 and multi-byte encodings), backing up and restoring files is more complicated.

It is important to understand that actual binary encoding of file names on the Tivoli St orage Manager server are independent of the locale used to backup or restore the files. The encoding of file names on the Tivoli Storage Manager server are always identical to the file names on the file system. A mismatch between the binary encoding of the file names and the locale used to run the Tivoli Storage Manager Client may cause files to be skipped during backup operations: however, the locale of the Tivoli Storage Manager Client has no effect on the binary encoding of the file names on the Tivoli Storage Manager server.

Summary

In environments that have mixed encodings (for example, both UTF-8 and multi-byte encodings), you can chose to execute the Tivoli Storage Manager Backup-Archive Client in either a single-byte locale or a multi-byte (including UTF-8) locale. This summary is for executing the client in a multi-byte (including UTF-8) locale. There are significant restrictions for backup and restore operations, include-exclude processing and messages produced by the client. The main advantage for running in a multi-byte locale in this environment is the ability for the client GUI to display file names more accurately.

If you cannot guarantee that all files and all directories are in the same locale, you should always run your scheduled backups using a single byte character s et locale so that no files are skipped due to the file name containing characters that are not defined in the current locale.

How do I know I'm in this environment?

It is not always easy to determine that you are running in this environment. The **locale** command will return the name of a multiple byte character set locale such as en_US.UTF -8, de_DE.UTF-8 or ja_JP.eucJP. Commands that deal with file names, such as **ls** will not always display all file names correctly.

If the backup of files in a file system occasionally fails with the following message, then the Tivoli Storage Manager Client is running in the wrong multiple byte character set l ocale for the specified file name.

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

In this case, set the locale to either a UTF-8 locale, or to one of the supported DBCS l ocals. If that fixes the problem, and errors are no longer detected when backing up the file systems, then you are running in a pure UTF-8 or DBCS environment.

If some backups continue fail with the above message, while other files are backed up successfully, then you are probably running in an environment where the file system names are encoded in two or more incompatible multiple b yte character set encodings.

Another way to detect an environment where file system names are encoded in two or more incompatible multiple byte character set encodings is to start two or more instances of the Tivoli Storage Manager Client GUI (dsmcad). Start one dsmcad in the single byte character set locale en_US, and start the second dsmcad in the suspected multiple byte locale of the file system. If some of the file system names are garbled in the dsmcad started in the single byte en_US locale, and all of the file names are correctly displayed in the Tivoli Storage Manager Client GUI, then you are probably running in a pure UTF-8 or DBCS environment. If some file system names are displayed correctly when viewed from the Tivoli Storage Manager Client GUI, while other file names are missing from the display, then you are probably running in an environment where the file system names are encoded in two or more incompatible multiple byte character set encodings.

Backup processing

Tivoli Storage Manager Client backup processing is supported with significant restrictions in this environment. Files may be skipped when running a backup in a multiple byte character set locale on a system containing file names composed of characters that are not valid in the multiple byte character set locale.

If you can not guarantee that all files and all directories will be created with the same locale, 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.

All multiple byte characters can be entered if "**editor no**" is specified in the dsm.sys configuration file, and all single byte ISO 8859 characters in the range of hexadecimal '00' to '7F' can be entered. (Single byte ISO 8859 characters in the range of hexadecimal '00' to '7F' are part of both the UTF-8 definition and the Tivoli Storage Manager Client supported DBCS locales.).

Single byte ISO 8859 characters in the range of hexadecimal 'A0' to 'FF' can not be entered on the command or backed up when running in a multiple byte character set environment because they are not valid UTF-8 or DBCS characters, and any attempt to process a file containing these special characters when running in a multiple byte locale will fail with the message

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

Restore processing

Tivoli Storage Manager Client restore processing is supported with significant restrictions in this environment. The same file name restrictions discussed in the section of "Backup processing" also apply to restore processing. And, of course, the files that could not be backed up because they contained invalid characters in their names cannot be restored.

Include-exclude processing

Tivoli Storage Manager Client include-exclude processing is supported with significant restrictions. File names containing invalid UTF-8 or DBCS characters will be ignored. Only valid multiple byte character set file names will be processed correctly.

Messages

Tivoli Storage manager messages are supported with restrictions that are locale dependent.

UTF-8 European locales such as de_DE.UTF-8

When running in a European UTF-8 locale such as de_DE.UTF-8, the corresponding Tivoli Storage Manager Client single byte character set message catalog will be used. In this locale, all file names will be displayed correctly, but message text containing multiple byte characters such the German character 'ä' may not be displayed correctly.

Tivoli Storage Manager Client supported DBCS locales such as ja_JP.eucJP

The Tivoli Storage Manager Client message catalogs fully support this environment.

UTF-8 Asian locales such as ja_JP.UTF-8

It is not advisable to run the Tivoli Storage Manager Client in the Japanese, Chinese or Korean UTF-8 locales. Instead switch to a UTF-8 locale that has a corresponding Tivoli Storage Manager Client provided single byte character set message catalog such as en_US.UTF-8. The Tivoli Storage Manager Client processing of UTF-8 file names is valid in any UTF-8 locale, and by running in a locale where the single byte character set message catalog is available, the Tivoli Storage Manager Client messages will be displayed properly.

Examples

The following example shows the Tivoli Storage Manager Client in a UTF-8 locale where some files have valid UF-8 names and some files have ISO 8859 character names that are not valid in the UTF-8 locale.

The files in the directory /testData/en_US_files were created in a single byte en_US locale, and the files in the directory /testdata/en_US.UTF-8_files were created in a multiple byte en_US.UTF-8 locale.

Notice that some of the single byte en_US file names are not displayed correctly when running in the en_US.UTF-8 locale, and the invalid characters are displayed as '?'.

*Example 1: The **ls** command listing of the file names*

```
>ls -l /testData/en_US.UTF-8_files/
total 16
-rw-r--r--  1 symonds users 19 2007-10-02 10:08 file1
-rw-r--r--  1 symonds users 24 2007-10-02 10:08 file2
-rw-r--r--  1 symonds users 26 2007-10-02 10:08 file3¢
-rw-r--r--  1 symonds users 25 2007-10-02 10:08 file4£
-rw-r--r--  1 symonds users 21 2007-10-02 10:08 file5Œ
> ls -l /testData/en_US_files
total 16
-rw-r--r--  1 symonds users 21 2007-10-02 10:07 file1
-rw-r--r--  1 symonds users 22 2007-10-02 10:07 file2
-rw-r--r--  1 symonds users 16 2007-10-02 10:07 file3?
```

```
-rw-r--r--  1 symonds users 13 2007-10-02 10:07 file4?
```

Example 2: Include-exclude statements

In the example below, notice when the dsm.sys configuration file contains a mixture of incompatible encodings. In this case, the vi editor display switches to a single byte locale and shows all characters as single byte characters even though it was invoked in a multiple byte locale. In the case of the multiple byte character, 'Œ', the multiple byte encoding is does not correspond to two displayable single byte characters, so the file name in the vi editor display is garbled.

When the dsm.sys file is processed by the Tivoli Storage Manager client in the multiple byte , all the files names containing valid multiple byte characters are processed correctly and the files are included in the backup.

However, the file names containing characters that are not valid in the multiple byte ch aracter set locale such as (file3¢ and file4£ in the directory /testData/en_US_files), will not be included in the backup by the Tivoli Storage Manager client.

```
editor no
exclude /testData/en_US.UTF-8_files/*
exclude /testData/en_US_files/*
include /testData/en_US.UTF-8_files/file1
include /testData/en_US.UTF-8_files/file2
include /testData/en_US.UTF-8_files/file3Â¢
include /testData/en_US.UTF-8_files/file4Â£
include /testData/en_US.UTF-8_files/file5Å~R
include /testData/en_US_files/file1
include /testData/en_US_files/file2
include /testData/en_US_files/file3¢
include /testData/en_US_files/file4£
```

Example 3: Execution of a selective backup operation

If you try to backup files with names containing characters that are not valid in the UTF -8 locale, errors occur, and the Tivoli Storage Manager client will not the backup those files.

```
>dsmc sel /testData/* -su=yes
Selective Backup function invoked.


ANS1228E Sending of object '/testData/en_US_files/file' failed
ANS4042E Object name '/testData/en_US_files/file' contains one or more
unrecognised characters and is not valid.
ANS1228E Sending of object '/testData/en_US_files/file' failed
ANS4042E Object name '/testData/en_US_files/file' contains one or more
unrecognised characters and is not valid.
ANS1228E Sending of object '/testData/ja_JP_files/file4' failed
ANS4042E Object name '/testData/ja_JP_files/file4' contains one or more
unrecognised characters and is not valid.
Directory-->                192 /testData [Sent]
Directory-->                168 /testData/en_US.UTF-8_files [Sent]
Directory-->                144 /testData/en_US_files [Sent]
```

```
Directory-->               144 /testData/ja_JP.UTF-8_files [Sent]
Directory-->               144 /testData/ja_JP_files [Sent]
Normal File-->              19 /testData/en_US.UTF-8_files/file1 [Sent]
Normal File-->              24 /testData/en_US.UTF-8_files/file2 [Sent]
Normal File-->              26 /testData/en_US.UTF-8_files/file¢ [Sent]
Normal File-->              25 /testData/en_US.UTF-8_files/file£ [Sent]
Normal File-->              21 /testData/en_US.UTF-8_files/file€ [Sent]
Normal File-->              21 /testData/en_US_files/file1 [Sent]
Normal File-->              22 /testData/en_US_files/file2 [Sent]
Normal File-->              21 /testData/ja_JP.UTF-8_files/file1 [Sent]
Normal File-->              22 /testData/ja_JP.UTF-8_files/file2 [Sent]
Normal File-->              29 /testData/ja_JP.UTF-8_files/file3? [Sent]
Normal File-->              29 /testData/ja_JP.UTF-8_files/file4? [Sent]
Normal File-->              21 /testData/ja_JP_files/file1 [Sent]
Normal File-->              22 /testData/ja_JP_files/file2 [Sent]
Normal File-->              15 /testData/ja_JP_files/file3 [Sent]
ANS1804E Selective Backup processing of '/testData/*' finished with failures.
```

Example 4: Execution of a restore operation.

In a multiple byte locale environment, files whose names do not have a valid multiple byte encoding cannot be restored using the Tivoli Storage Manager Client single file restore command because the file name cannot be specified on the command line.

dsmc rest /testdata/en_US_files/file3?

The Tivoli Storage Manager Client must be run in the locale corresponding to the encoding of the file name for the file name to be correctly entered. However, all of the files can be restored by using a wild card qualified no-query restore as shown in the example below.

```
dsmc>  rest /testData/en_US/*
Restore function invoked.
Restoring             144 /testData/en_US_files [Done]
Restoring              21 /testData/en_US_files/file1 [Done]
Restoring              22 /testData/en_US_files/file2 [Done]
Restoring              16 /testData/en_US_files/file? [Done]
Restoring              13 /testData/en_US_files/file? [Done]
Restore processing finished.
```

This example of a no-query restore illustrates an interesting difference between backup processing and restore processing. The equivalent backup command,

dsmc sel "/testdata/en_US_files/*"

will fail to backup the files containing characters in their name that are invalid in the multiple byte character set locale. That is because include/exclude processing is performed as part of the Tivoli Storage Manager Client backup logic. If a file name contains characters that are invalid in the current locale, the include/exclude processing always excludes that file.

The Tivoli Storage Manager Client restore operation does not involve any include/exclude processing, so files names containing invalid characters can always be restored when performing a no-query restore.

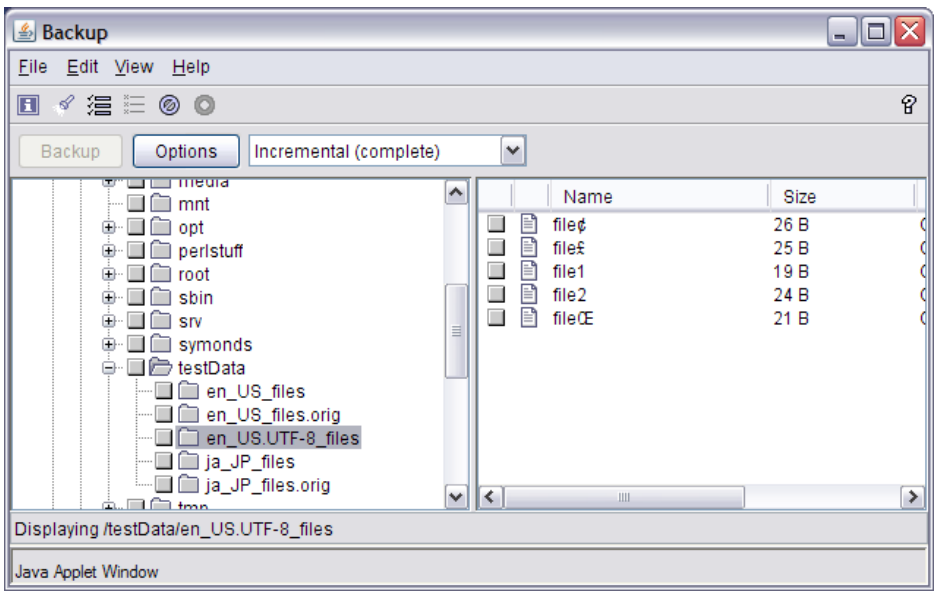
Example 5: Execution of a restore operation with a file list

The filelist option cannot be used to restore individual files whose names are not valid in the multiple byte character set locale.

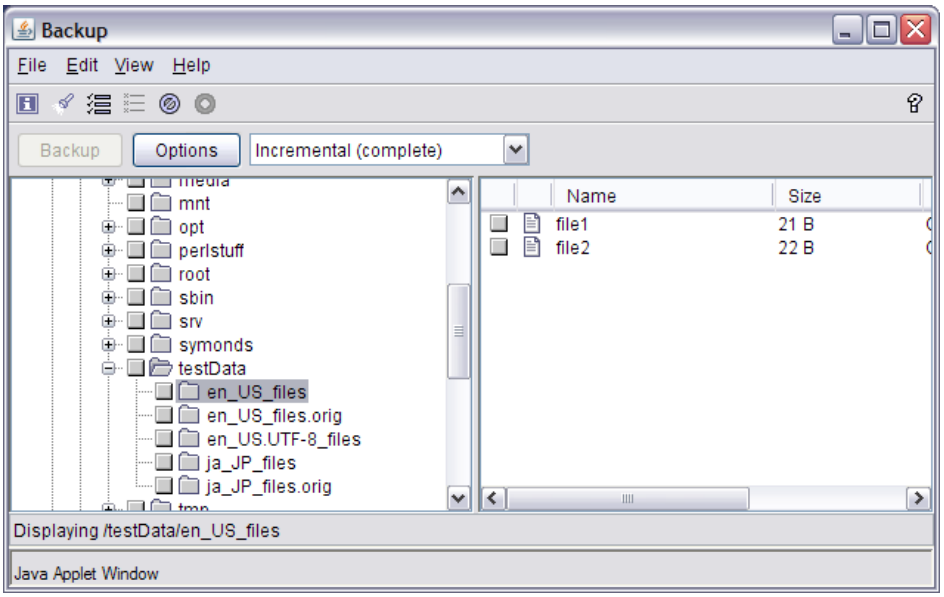
```
dsmc> rest -filelist=fileList1
Restore function invoked.

 ** Unsuccessful **
ANS1345E No objects on server match '/testData/en_US_files/file?'

 ** Unsuccessful **
ANS1345E No objects on server match '/testData/en_US_files/file?'

>dsmc rest /testData/en_US_files/file3¢
ANS1302E No objects on server match query
```

Example 6: The client GUI started in a UTF-8 locale

The Tivoli Storage Manager GUI displays all file names correctly when the dsmcad is started in the same multiple byte character set locale as the encoding of the file names if the correct fonts are used.



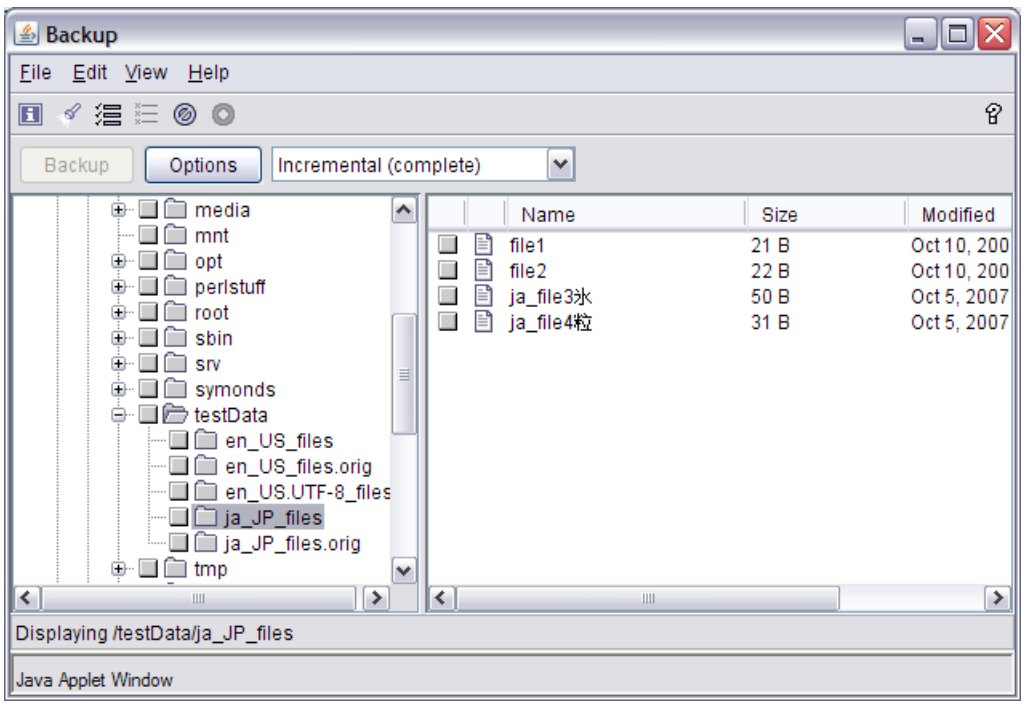
When the Tivoli Storage manager dsmcad is running in a multiple byte character set l ocale, the GUI does not display files whose names contain characters that are invalid in the multiple byte locale.



Notice that the files with invalid UTF-8 names are not displayed.

Example 7: The client GUI started in ja_JP.eucJP locale

This example shows the GUI when the dsmcad was started in the ja_JP.eucJP l ocale. The ja_JP.eucJP characters may appear different from those displayed in the terminal emulator depending upon the fonts being used. However as long as the dsmcad was started in the same locale as the files to be displayed, all of the files will be shown and none will be skipped.



8. Scenario 4b: Mixed Encodings – Executing in Single-byte Locale

Summary

In environments that have mixed encodings (for example, both UTF-8 and multi-byte encodings), you can chose to execute the Tivoli Storage Manager Backup-Archive Client in either a single-byte locale or a multi-byte (including UTF-8) locale. This summary is for executing the client in a single-byte locale. By executing in a single-byte locale, you can ensure that all files and directories will be processed during scheduled backup operations and during restores of entire volumes or directories. Messages produced by the client will always be readable. The disadvantages of this environment will be when specifying explicit file names for backup and restore operations and displaying files in the GUI might not show the correct name.

How do I know I'm in this environment?

The **locale** command will return the name of a single byte character set locale such as en_US, de_DE, or one of the ISO 8859 variants. Commands that deal with file names, such as **ls,** will not always display all file names correctly.

Backup processing

Tivoli Storage Manager Client backup processing is supported with some restrictions in this environment. The error

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

will not occur in this environment, and all files meeting the selection criteria will be backed up.

File names cannot always be entered on the Tivoli Storage Manager Client command line. All single byte ISO 8859 characters can be specified; however, UTF-8 multiple byte characters and Chinese, Japanese or Korean DBCS characters can not be entered on the command line.

Entering multiples byte characters in a single byte character set locale

The Tivoli Storage Manager Client filelist option can be used to get around the above restrictions on entering multiple byte UTF-8 or DBCS characters when running in a single byte character set locale. If the **ls** command is issued in a single byte ISO 8859 locale, a DBCS locale such as js_JP.eucJP or a UTF-8 locale and the output is sent to a file, and the resulting list of file names can be used as the filelist input for either backup and restore processing. This works because the **ls** command sends the binary representation of the file names to the output file or display.

Restore processing

Tivoli Storage Manager Client restore processing is supported with some restrictions in this environment. The same file name restrictions discussed in the above section on "Backup processing" also apply to restore processing.

Include-exclude processing

Tivoli Storage Manager Client include-exclude processing is fully supported.

Messages

Only the Tivoli Storage Manager Client single byte character set messages catalogs are supported in this environment. The Tivoli Storage Manager Client DBCS messages catalogs for the Chinese, Japanese and Korean languages are not supported in this environment. The default is to use the single byte character set English en_US message catalog.

Examples

The following example shows the Tivoli Storage Manager client in an ISO 8859 locale.

Notice that file names containing multiple byte en_US.UTF-8 characters that are composed of two or more valid ISO 8859 characters, such as “file¢” and “file£,” are displayed as a sequence of two single byte characters, “file3Â¢” and “file4Â£,” when running in a single byte character set locale. When a multiple byte character in a file name contains an invalid ISO 8859 byte, the invalid byte is replaced with the character '?'. For example, the file named “ fileŒ” is displayed as “file5?”

*Example 1: The **ls** command listing of the test file names*

```
>ls -l /testData/en_US.UTF-8_files/
total 16
-rw-r--r--  1 symonds users 19 2007-10-02 10:08 file1
-rw-r--r--  1 symonds users 24 2007-10-02 10:08 file2
-rw-r--r--  1 symonds users 26 2007-10-02 10:08 file3Â¢
-rw-r--r--  1 symonds users 25 2007-10-02 10:08 file4Â£
-rw-r--r--  1 symonds users 21 2007-10-02 10:08 file5Å?
> ls -l /testData/en_US_files
total 16
-rw-r--r--  1 symonds users 21 2007-10-02 10:07 file1
-rw-r--r--  1 symonds users 22 2007-10-02 10:07 file2
-rw-r--r--  1 symonds users 16 2007-10-02 10:07 file3¢
-rw-r--r--  1 symonds users 13 2007-10-02 10:07 file4£
```

Example 2: Execution of a selective backup operation

All files can be backed up when running in a single byte locales.

```
>dsmc sel /testData/* -su=yes
Selective Backup function invoked.
Directory-->                192 /testData [Sent]
Directory-->                168 /testData/en_US.UTF-8_files [Sent]
Directory-->                144 /testData/en_US_files [Sent]
Directory-->                144 /testData/ja_JP.UTF-8_files [Sent]
Directory-->                144 /testData/ja_JP_files [Sent]
Normal File-->               19 /testData/en_US.UTF-8_files/file1 [Sent]
```

```
Normal File-->              24 /testData/en_US.UTF-8_files/file2 [Sent]
Normal File-->              26 /testData/en_US.UTF-8_files/fileÂ¢ [Sent]
Normal File-->              25 /testData/en_US.UTF-8_files/fileÂ£ [Sent]
Normal File-->              21 /testData/en_US.UTF-8_files/fileÅ [Sent]
Normal File-->              16 /testData/en_US_files/file¢ [Sent]
Normal File-->              13 /testData/en_US_files/file£ [Sent]
Normal File-->              21 /testData/en_US_files/file1 [Sent]
Normal File-->              22 /testData/en_US_files/file2 [Sent]
Normal File-->              21 /testData/ja_JP.UTF-8_files/file1 [Sent]
Normal File-->              22 /testData/ja_JP.UTF-8_files/file2 [Sent]
Normal File-->              29 /testData/ja_JP.UTF-8_files/file3ç [Sent]
¡ [Sent]
Normal File-->              29 /testData/ja_JP.UTF-8_files/file4å³ [Sent]
Normal File-->              21 /testData/ja_JP_files/file1 [Se nt]
Normal File-->              22 /testData/ja_JP_files/file2 [Sent]
Normal File-->              12 /testData/ja_JP_files/file4±¦ [Sent]
Normal File-->              15 /testData/ja_JP_files/file3Ìµ [Sent]
Selective Backup processing of '/testData/*' finished without failure.
```

Example 3: Query file backup in single-byte character locale

A query backup when running in the same single byte character set locale displays all of the files that were backed up. Of course, since we are running in a single byte locale, the names of files with multiple byte character set encoded characters are not displayed co rrectly.

```
dsmc> q ba /testData/* -su=yes
   Size  A/I  File
   ----  ---  ----
   168    A  /testData/en_US.UTF-8_files
   144    A  /testData/en_US_files
   144    A  /testData/ja_JP.UTF-8_files
   144    A  /testData/ja_JP_files
    19    A  /testData/en_US.UTF-8_files/file1
    24    A  /testData/en_US.UTF-8_files/file2
    26    A  /testData/en_US.UTF-8_files/fileÂ¢
    25    A  /testData/en_US.UTF-8_files/fileÂ£
    21    A  /testData/en_US.UTF-8_files/fileÅ
    21    A  /testData/en_US_files/file1
    22    A  /testData/en_US_files/file2
    16    A  /testData/en_US_files/file¢
    13    A  /testData/en_US_files/file£
    21    A  /testData/ja_JP.UTF-8_files/file1
    22    A  /testData/ja_JP.UTF-8_files/file2
    29    A  /testData/ja_JP.UTF-8_files/file3ç
    29    A  /testData/ja_JP.UTF-8_files/file4å³
    21    A  /testData/ja_JP_files/file1
    22    A  /testData/ja_JP_files/file2
    15    A  /testData/ja_JP_files/file3Ìµ
```

```
 12    A  /testData/ja_JP_files/file4±¦
```

Example 4: Query file backup in a UTF-8 locale

The same query for all of the backed up files is different when run in a UTF -8 locale. Notice that the file names containing characters that are not valid in the UTF -8 locale (/testData/en_US_files/file¢, /testData/en_US_files/file£ and /testData/ja_JP_files/file4 右) are not shown in the list of files. These files can neither be queried nor restored when running in a locale where the file names contain invalid characters. They can only be restored (with a single file restore) in the locale corresponding to their encoding, or in a single byte locale where all byte encodings are valid.

```
dsmc> q ba /testData/* -su=yes
    Size   A/I  File
    ----   --- ----
    168    A  /testData/en_US.UTF-8_files
    144    A  /testData/en_US_files
    144    A  /testData/ja_JP.UTF-8_files
    144    A  /testData/ja_JP_files
     19    A  /testData/en_US.UTF-8_files/file1
     24    A  /testData/en_US.UTF-8_files/file2
     26    A  /testData/en_US.UTF-8_files/file¢
     25    A  /testData/en_US.UTF-8_files/file£
     21    A  /testData/en_US.UTF-8_files/fileŒ
     21    A  /testData/en_US_files/file1
     22    A  /testData/en_US_files/file2
     21    A  /testData/ja_JP.UTF-8_files/file1
     22    A  /testData/ja_JP.UTF-8_files/file2
     29    A  /testData/ja_JP.UTF-8_files/file3無
     29    A  /testData/ja_JP.UTF-8_files/file4右
     21    A  /testData/ja_JP_files/file1
     22    A  /testData/ja_JP_files/file2
     15    A  /testData/ja_JP_files/file3
```

Example 5: Executing a restore operation

The Tivoli Storage Manager Client should normally be run in the same locale as that of the file to be restored. However when multiple files with names that are encoded with different incompatible locales are to be restored in a single operation, it is possible (in a single byte character set locale such as en_US) to use the filelist option explicitly to restore such files.

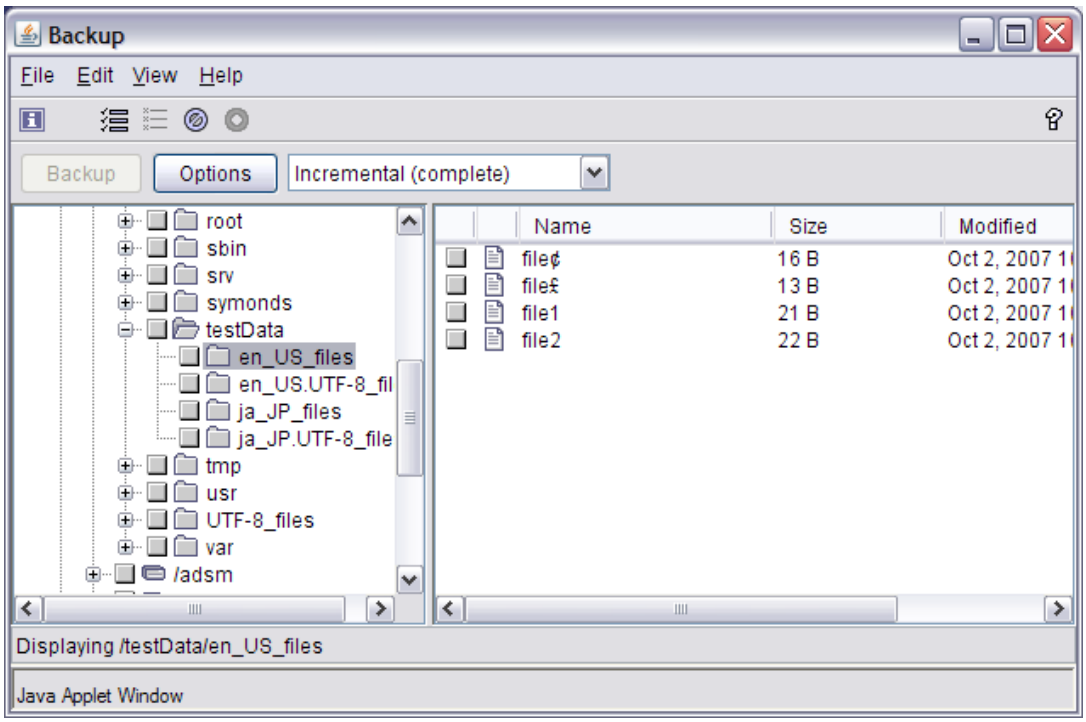
```
ls /testData/en_US.UTF-8_files/* > fileList1
```

Now edit fileList1 to remove all lines except the lin e for the file to be restored, “/testData/en_US.UTF-8_files/file5Œ”, and then use fileList1 as input to the restore command.

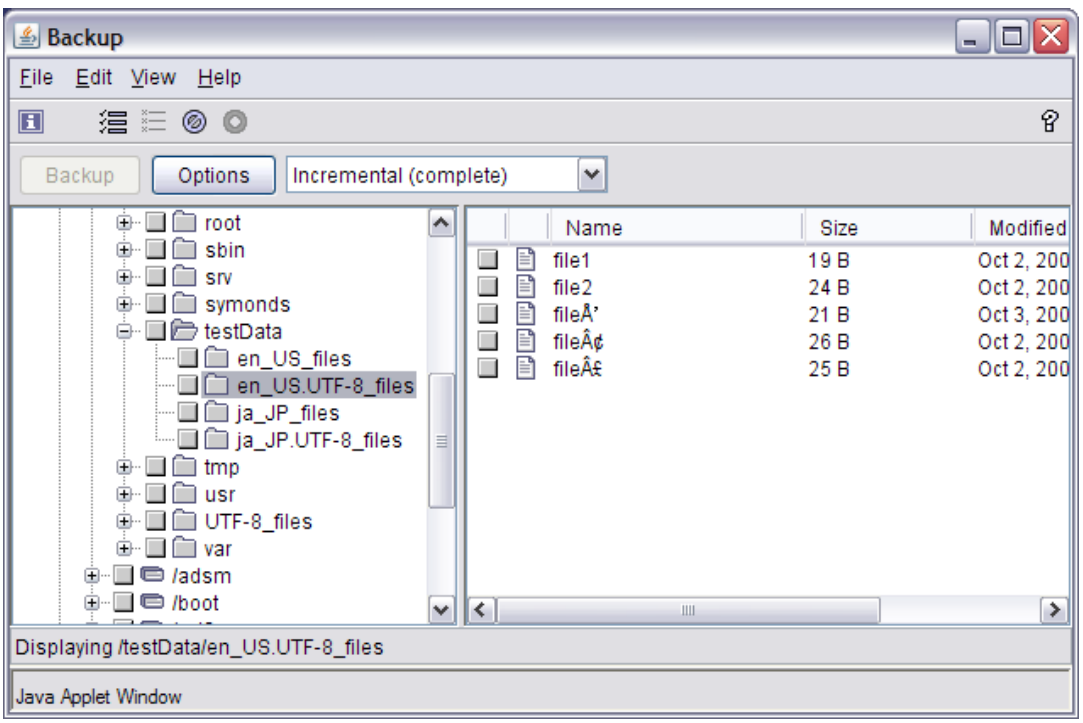
```
>dsmc rest -filelist=fileList1
Restoring              21 /testData/en_US.UTF-8_files/file5Å
```

Example 6: The client GUI

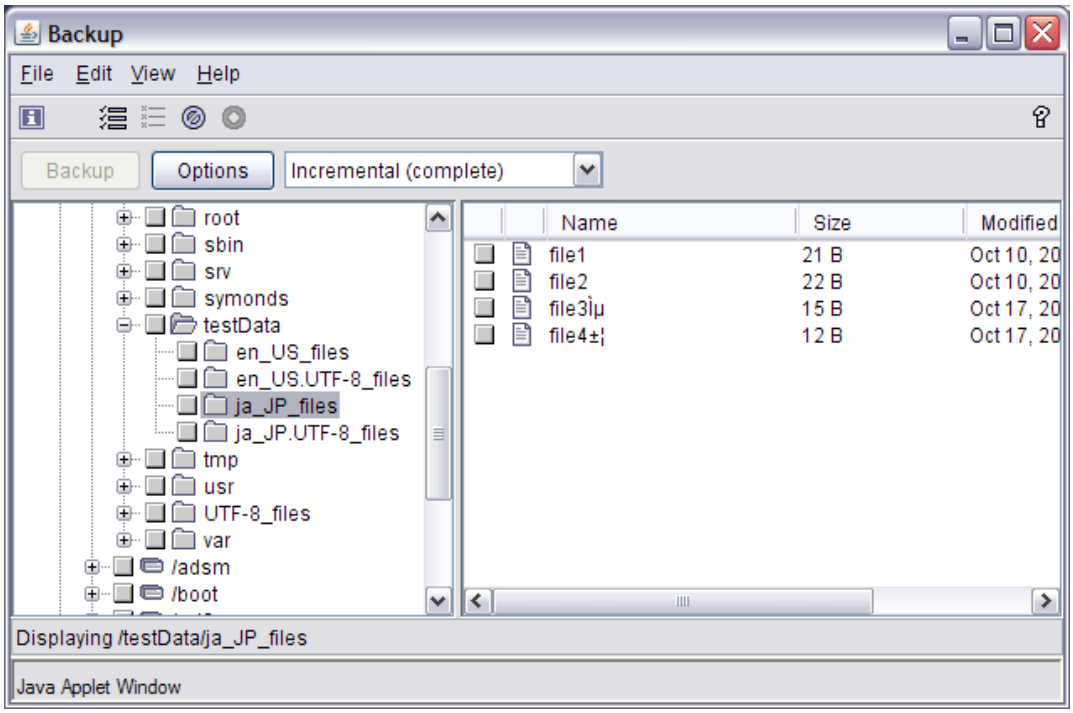
The Tivoli Storage Manager GUI shows all file names when the dsmcad is started in a single byte locale. File names containing single byte character set characters will be di splayed correctly.



File names containing UTF-8 multiple byte characters will be shown, but multiple byte characters in the file names displayed will be garbled.



File names containing DBCS characters will be shown, but multiple byte characters in the file names displayed will be garbled.



Examples using ja_JP.eucJP encoded files

The following example shows the Tivoli Storage Manager client in an ISO 8859 locale.

Notice that file names containing multiple byte ja_JP.eucJP characters that are composed of two or more valid ISO 8859 characters are displayed as a sequence of two single byte characters “file3É¹” and “file4Î³ ” when running in a single byte character set locale. When a multiple byte character in a file name contains an invalid ISO 8859 byte, the invalid byte is replaced with the character '?'.

*Example 1: The **ls** command listing of the test file names*

```
>ls -l /testData/ja_JP_files/
total 16
-rw-r--r--  1 symonds users 21 2007-10-02 10:08 file1
-rw-r--r--  1 symonds users 22 2007-10-02 10:08 file2
-rw-r--r--  1 symonds users 50 2007-10-02 10:08 file3É¹
-rw-r--r--  1 symonds users 31 2007-10-02 10:08 file4Î³
```

Example 2: Include-Exclude statements

The Tivoli Storage Manager include/exclude processing works correctly for these ja_JP.eucJP files even though the client is running in a single byte ISO 8859 locale. In this example the file named file3É¹ is excluded even though we are running in a single byte character set locale.

The dsm.sys configuration file can contain a mixture of single byte and multiple byte characters. Notice the include statement for the DBCS file “ /testData/ja_JP_files/file4右”is commented out.

```
exclude /testData/ja_JP_files/*
include /testData/ja_JP_files/ja_file1
include /testData/ja_JP_files/ja_file2
include /testData/ja_JP_files/ja_file4Î³
* include /testData/ja_JP_files/ja_file3É'
```

Example 3: Executing a selective backup operation

The Tivoli Storage Manager Client include/exclude processing works correctly for all files including those with multiple byte characters in their names. In the example below, all of the included files, including those with multiple byte characters in their names, are backed up.

```
>dsmc sel /testData/ja_JP_files/*
Selective Backup function invoked.
ANS1115W File '/testData/ja_JP_files/ja_file3É'' excluded by Include/Exclude list
Directory-->                160 /testData/ja_JP_files [Sent]
Normal File-->               21 /testData/ja_JP_files/file1 [Sent]
Normal File-->               22 /testData/ja_JP_files/file2 [Sent]
Normal File-->               31 /testData/ja_JP_files/ja_file4Î³ [Sent]
Selective Backup processing of '/testData/ja_JP_files/*' finished without failure.
```

If the "*" is removed from the include statement for the file file3É¹ , that file is also backed up.

```
tsm> sel /testData/ja_JP_files/*
Selective Backup function invoked.
Directory-->                160 /testData/ja_JP_files [Sent]
Normal File-->               21 /testData/ja_JP_files/file1 [Sent]
Normal File-->               22 /testData/ja_JP_files/file2 [Sent]
Normal File-->               50 /testData/ja_JP_files/ja_file3É' [Sent]
Normal File-->               31 /testData/ja_JP_files/ja_file4Î³ [Sent]
Selective Backup processing of '/testData/ja_JP_files/*' finished without failure.
```

Example 4: Query backup files

The files backed up in the directory "/testData/ja_JP_files" can be queried with the Tivoli Storage manager client command

```
>dsmc q ba /testData/ja_JP_files/*
Size  A/I File
----  --- ----
21    A  /testData/ja_JP_files/ja_file1
22    A  /testData/ja_JP_files/ja_file2
50    A  /testData/ja_JP_files/ja_file3¦Ã
31    A  /testData/ja_JP_files/ja_file4¦Ã
```

Notice that the names of the files returned does not match either the correct ja_JP.eucJP name or the name returned by the **ls** command.

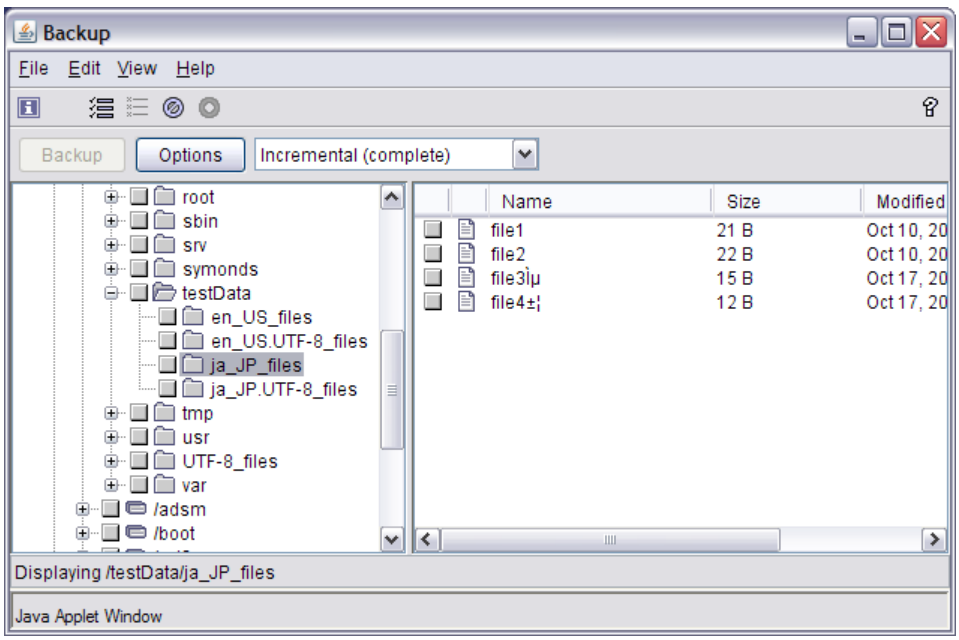
Example 5: Executing a restore operation

These files can be restored while running in a single byte character set locals such as en_US by using a wild card specification; however, it is recommended to run the Tivoli Storage Manager Client in the same locale as that of the file to be restored.

```
>dsmc rest /testData/ja_JP_files/*
Restore function invoked.
ANS1247I Waiting for files from the server...
Restoring             160 /testData/ja_JP_files [Done]
Restoring              21 /testData/ja_JP_files/file1 [Done]
Restoring              22 /testData/ja_JP_files/file2 [Done]
Restoring              50 /testData/ja_JP_files/ja_file3É¹ [Done]
Restoring              31 /testData/ja_JP_files/ja_file4Î³ [Done]
Restore processing finished.
```

Example 6: The client GUI

The ja_JP.eucJP names displayed by the Tivoli Stora ge Manager GUI managed by a dsmcad started in a single byte character set locale are not displayed correctly, but they can be selected and will be processed correctly for a backup or restore operation .



9. Appendix A – System Configuration Used for This Paper

This paper is divided into four basic character encoding environments:

1. A pure ISO 8859 environment where all of the file names have the same single byte ISO 8859 encoding.
2. A pure DBCS multiple byte environment all of the file names have a singl e DBCS encoding, such as ja_JP.eucJP.
3. A pure UTF-8 environment where all of the file names are encoded according to the Unicode UTF-8 standard.
4. A mixed environment where the file names are encoded in two of more incompatible character sets (such as ISO 8859 and UTF-8, or different DBCS locales such as ja_JP.eucJP and zh_CN, or a DBCS locale such as ja_JP and UTF -8).

The following file system configuration was established to demonstrate working in these four different environments.

en_US_files

ja_JP_files

en_US.UTF-8_files

ja_JP.UTF-8_files

The directory /testData/en_US_files con tains files with the following ISO 8859-1 (en_US) encoded names.

File Name	Locale	Hexadecimal encoding
file1	en_US	66 69 6C 65 31
file2	en_US	66 69 6C 65 32
file3¢	en_US	66 69 6C 65 33 A2
filee4£	en_US	66 69 6C 65 34 A3

The directory /testData/ja_JP_files contains files with the following DBCS ja_JP.eucJP encoded names.

File Name	Locale	Hexadecimal encoding
file1	ja_JP.eucJP	66 69 6C 65 31
file2	ja_JP. eucJP	66 69 6C 65 32

File Name	Locale	Hexadecimal encoding
file3無	ja_JP. eucJP	66 69 6C 65 33 CC B5
file4右	ja_JP. eucJP	66 69 6C 65 34 B1 A6

The directory /testData/en_US.UTF-8_files contains files with the following English UTF-8 encoded names.

File Name	Locale	Hexadecimal encoding
file1	en_US.UTF-8	66 69 6C 65 31
file2	en_US.UTF-8	66 69 6C 65 32
file3¢	en_US.UTF-8	66 69 6C 65 33 C3 A2
file4£	en_US.UTF-8	66 69 6C 65 34 C3 A3
file5Œ	en_US.UTF-8	66 69 6C 65 35 C5 92

The directory /testData/ja_JP.UTF-8_files contains files with the following Japanese UTF-8 encoded names.

File Name	Locale	Hexadecimal encoding
file1	ja_JP.UTF-8	66 69 6C 65 31
file2	ja_JP.UTF-8	66 69 6C 65 32
file3無	ja_JP.UTF-8	66 69 6C 65 33 E7 84 A1
file4右	ja_JP.UTF-8	66 69 6C 65 34 E5 8F B3

These names of the files were chosen to help the reader understand the difference b etween ISO 8859 encoding, UTF-8 encoding, and a DBCS encoding such as ja_JP.

Notice that the files named “file3¢” and “file4£” look the same in both the en_US_files directory and the en_US.UTF-8_files directory, and the files file3無 and file4右 appear the same in the ja_JP_files and the ja_JP.UTF-8 directory; however, the encoding of these multiple byte characters differs in the different locales, and the file names are actually quite different.

The character 'Œ' does not correspond to a valid single byte ISO 8859 character, so a file with the name “file5Œ” could not be created when running in a single byte character set locale.

The file names in these examples all have a unique numeric identifier in their name. This is very helpful when running in a mixed locale environment because it provides a way to identify the file when the file name contains characters that cannot be displayed correctly in the current locale.

10. Appendix B – Locales

The locale determines the internationalization characteristics of the computer. The default system locale is defined in a system configuration file such as /etc/environment. Individual user locales that override the system default are set in the user login configuration files such as .profile.

The current locale can be discovered by issuing the command **locale**.

```
>locale
LANG=en_US
LC_COLLATE=en_US
LC_CTYPE=en_US
LC_MONETARY=en_US
LC_NUMERIC=en_US
LC_TIME=en_US
LC_MESSAGES=en_US
LC_ALL=
```

Locale variables are defined as follows:

LANG - This variable corresponds to the “locale” as used in this document. It determines the character set encoding used for displays and for file names created during the session running under that locale.

LC_MESSAGES - This variable determines the format for affirmative and negative system responses.

LC_COLLATE - This variable defines character or string collation information.

LC_CTYPE - This variable defines character classification, case conversion, and other character attributes.

LC_MONETARY -This variable defines rules and symbols for formatting monetary numeric information.

LC_NUMERIC - This variable defines rules and symbols for formatting non-monetary numeric information.

LC_TIME -This variable defines rules for formatting date and time items.

LC_ALL - This variable is used to set all of the above environment variables except LANG.

Other pertinent notes about locale:

1. The locales supported by the system can be discovered by issuing the command **locale -a**
2. The locale for a system or login session is set by exporting the above environment variables.
3. The locale can be changed at any time by exporting a different value for LC_ALL or LANG.

11. Appendix C – ISO 8859-1 and 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 the 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 no rmally be replaced with the '?' character.

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

Because a single standard for the 8 bit ASCII characters can not 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 '¢' in ISO 8859-1; however, it is the character ' Ђ' in ISO 8859-5 and 'ą' is ISO 8859-16.

12. Appendix D – 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 16,000 different characters. This original standard has been extended to support more than 16,000 characters and to support systems such a 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 of 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".

The encoding for seven bit ASCII characters (in the range hexadecimal 00 to hexadecimal 7F) is the same in both ISO 8859 and UTF-8. That is why names such as "file1" have the same encoding in both ISO 8859 and UTF-8.

All other characters required at least two bytes when encoded in UTF-8.

Each of the individual bytes in a multiple byte UTF-8 character always has a value greater than hexadecimal 7F. This restriction is important because the UTF-8 encoding of the seven bit ASCII characters is the same as the ISO 8859 encoding. If one of the bytes in a multiple byte UTF-8 character had the same value as a seven bit ASCII character, there would be no fool-proof way to determine if the byte was part of the multiple byte character or if it was a stand-alone character.

A second implication of this restriction on the values of individual bytes in a UTF-8 multiple byte character is that in a single byte character set locale, the display of a multiple byte UTF-8 character frequently shows as two or more ISO 8859 special characters. However, not all UTF-8 multiple byte characters can be displayed as two or more single byte special characters. This is because byte values in the range hexadecimal 80 to hexadecimal 9F are not defined in ISO 8859.

In the previous examples, the character '¢' has an ISO 8859-1 value of hexadecimal 'A2' and a UTF-8 multiple byte character value of hexadecimal 'C3 A2'. The UTF-8 multiple byte encoding is the same as the ISO 8859-1 encoding with 'Â' in front of it.

This correspondence between the display of ISO 8859 characters and UTF-8 characters is not generally true. All individual UTF-8 multiple byte character byte values are in the range of

hexadecimal '80' to 'FF'; however, bytes in the range of hexadecimal '80' through '9F' do not correspond to displayable characters in ISO 8859. Any UTF-8 character containing a byte within this range cannot be displayed as two or more ISO 8859 single byte characters. In the previous example, the character 'Œ' has the hexadecimal value 'C5 92'. The byte hexadecimal 'C5' corresponds to the ISO 8859 character 'Å', but the byte hexadecimal '92' does not correspond to an ISO 8859 character and cannot be displayed. Frequently the invalid character is displayed as '?'.

UTF-8 encodes each character in one to four octets (8-bit bytes):

1. One byte is needed to encode the 128 US-ASCII characters (Unicode range U+0000 to U+007F).
2. Two bytes are needed for Latin letters with diacritics and for characters from Greek, Cyrillic, Armenian, Hebrew, Arabic, Syriac and Thaana alphabets (Unicode range U+0080 to U+07FF).
3. Three bytes are needed for the rest of the Basic Multilingual Plane (which contains virtually all characters in common use).
4. Four bytes are needed for characters in the other planes of Unicode, which are rarely used in practice.

13. Appendix E – Japanese 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. The Tivoli Storage Manager client supports ja_JP.eucJP, which is incompatible with UTF-8.

1. Conversion between ISO 8859 and UTF-8

Cut and paste operations between terminal emulator windows attempt to convert between ISO 8859 and UTF-8. If a character such as '¢' that is valid in both locales is copied from an ISO 8859 locale to a UTF-8 locale, the character ' will be converted to its UTF-8 representation. However, if the character such as 'Œ' that is valid in the UTF-8 but not the ISO 8859 locale is copied from the UTF-8 locale to an ISO 8859 locale, the paste operation will not succeed, and no character will be pasted into the ISO 8859 window.

2. Conversion to wide characters

In order to do internal string comparisons, the TSM client converts the character strings to wide character strings for processing include/exclude statements and comparing with objects previously backed up to TSM server.

Conversion of the character string to a wide character string is dependent on the locale that TSM client is running under. If a character cannot be converted to a valid wide character under the locale TSM client is running, TSM client will skip that object for backup processing, or if the problem character is within an include/exclude statement, TSM client will ignore that include/exclude statement.

The recommendation is to run TSM client using a SBCS Locale like en_US because single byte locales will never fail the wide character string conversion.

14. Appendix F – Locale and the UNIX Shell

It is important to realize that programs take on the loc ale that they are launched in, and once a program has been launched, modifying the locale environment variables does not change the locale of already-launched programs.

This subtlety is important because the UNIX shell is a program, and once it has been l aunched, changing the locale environment variables does not change the locale of the shell. For example, if the shell is launched in the *en_US.UTF-8* locale and the locale environment variables are changed so that the Tivoli Storage Manager Backup -Archive Client is launched in the *en_US* locale, then the locale of the shell and the locale of the Tivoli Storage Manager client will be out of sync. The locale of the shell would continue to be *en_US.UTF-8;* only the locale of the Tivoli Storage Manager Client would be changed. All of the characters in commands issued from the shell, such as **dsmc sel f£** are in *en_US.UTF-8*, even though the Tivoli Storage manager client is running in the *en_US* locale. In this case, the shell command **dsmc sel f£** would backup the file whose name is encoded with the UTF -8 binary encoding *0x66C2A3* even though the Tivoli Storage Manager Client is running in a *en_US* locale.

Conversely, if the shell is running in the *en_US* locale, and the locale environment variables are modified so that the Tivoli Storage Manager Client is running in the *en_US.UTF-8* locale, the shell command **dsmc sel f£** would attempt to backup the file whose name is encoded with the ISO 8857-1 binary encoding *0x66A3*. In this case, the command would fail with the following error message because the ISO 8857-1 file name is not a valid UTF -8 name:

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

The following examples illustrate the results of invoking the Tivoli Storage Manager Client in different shell and client locales. Xterm1 is started i n a UTF-8 locale (for example, *en_US.UTF-8*), and xterm2 is started in the single byte character set *en_US* locale. The UTF -8 xterm1 creates a 512-byte file whose name is encoded in UTF -8, "/mixed/f£" (the last two characters of the name are encoded as *0x66C2A3)*, and the single byte character set *en_US* xterm2 creates a 256 -byte file whose name is encoded in *en_US*, "/mixed/f£" (the last two characters of the name are encoded as *0x66A3*).

This example shows the output when both the shell and the Tivoli Stor age Manager Client are in the same en_US.UTF -8 locale. Notice that the 512 -byte *en_US.UTF-8* file name is displayed correctly while the 256 -byte *en_US* file name is not displayed correctly. The 512 -byte UTF -8 file is backed up in response to the command **dsmc sel /mixed/f£**. A wild carded selective command to back up both files backs up the 512 -byte *en_US.UTF-8* file, but not the 256 -byte *en_US* file:

```
ls -l /mixed/*

 -rw-rw-r-- 1 symonds symonds 512 Jan 7 19:48 /mixed/f£
 -rw-rw-r-- 1 symonds symonds 256 Jan 7 19:50 /mixed/f?

dsmc sel /mixed/f£
Normal File-->                              512  /mixed/f£  [Sent ]

dsmc sel /mixed/*
ANS1228E Sending of object /mixed/f? failed
ANS4042E Object name /mixed/file? contains one or more unrecognized
characters and is not valid
Normal File-->                              512  /mixed/f£  [Sent ]
```

Now change the locale environment variables to en_US. The Tivoli Storage Manager Client is now running in a single byte locale. Notice that the **ls** command still has the same output, and the command dsmc sel /mixed/f£ continues to backup the 512-byte UTF-8 file. However the Tivoli Storage Manager Client's wild carded selective command successfully backs up both files and does not encounter any failures.

```
export LANG=en_US
export LC_ALL=en_US

ls -l /mixed/*

 -rw-rw-r-- 1 symonds symonds 512 Jan 7 19:48 /mixed/f£
 -rw-rw-r-- 1 symonds symonds 256 Jan 7 19:50 /mixed/f?

dsmc sel /mixed/f£
Normal File-->                          512  /mixed/f£  [Sent ]

dsmc sel /mixed/*
Normal File-->                          256  /mixed/f?  [Sent ]
Normal File-->                          512  /mixed/f£  [Sent ]
```

This example shows the output when both the shell and the Tivoli Storage Manager Client are in the same en_US single byte locale. Notice that the 256-byte *en_US* file name is displayed correctly and the 512-byte *en_US.UTF-8* file name is displayed as a combination of three characters, the character 'f', the first byte in the multiple byte character 'Â' followed by the second byte in that character, '£'.

Also notice that in this case, the 256-byte *en_US* file is backed up in response to the command **dsmc sel /mixed/f£,** and the Tivoli Storage Manager Client wild carded selective command does not encounter any failures.

ls -l /mixed/*

```
 -rw-rw-r-- 1 symonds symonds 256  Jan 7 19:48 /mixed/f£
 -rw-rw-r-- 1 symonds symonds 512  Jan 7 19:50 /mixed/fÂ£
```

dsmc sel /mixed/f£

```
Normal File-->          256  /mixed/f£  [Sent ]
```

dsmc sel /mixed/*

```
Normal File-->          256  /mixed/f£  [Sent ]
Normal File-->          512  /mixed/fÂ£  [Sent ]
```

Now change the locale environment variables to *en_US.UTF-8*. Notice that the **ls** command still has the same output (the file f£ is still displayed as a combination of three characters, the character 'f', the first byte in the multiple byte character 'Â' followed by the second byte in that character '£').

However, now the 512-byte *en_US.UTF-8* file is backed up in response to the command **dsmc sel /mixed/f£** and the wild carded selective command to back up both files now only backs up the 512-byte *en_US.UTF-8* file, but not the 256-byte *en_US* file.

```
export LANG=en_US.UTF-8
export LC_ALL=en_US.UTF-8

ls -l /mixed/*

 -rw-rw-r-- 1 symonds symonds 512 Jan 7 19:48 /mixed/fÂ£
 -rw-rw-r-- 1 symonds symonds 256 Jan 7 19:50 /mixed/f?
```

```
dsmc sel /mixed/f£
Normal File-->                          512   /mixed/f£  [Sent ]

dsmc sel /mixed/*
ANS1228E Sending of object /mixed/f£ failed
ANS4042E Object name /mixed/file£ contains one or more unrecognized
characters and is not valid
Normal File-->                          512   /mixed/fÂ£  [Sent ]
```