Character Encoding

4

In order to support a wide range of languages, The current XView release uses Extended UNIX Code (EUC) as its primary encoding method. EUC encoding is suited for internationalized applications because it is compatible with ASCII[1] and, at the same time, supports multiple character sets.

The character sets you use depend on the locale(s) associated with your application:

- *Non-Asian locales* usually have a single character set. For example, ASCII or ISO Latin-1 is suited for English or western European languages.
- *Asian locales* usually have multiple character sets.

EUC characters and text strings use either *multibyte* or *wide character* representation. In multibyte representation, characters are represented by a varying number of bytes. In wide character representation, characters are represented by a fixed number of bytes.

In the current XView release, attributes and functions have been modified to handle multibyte strings, and additional attributes and functions accommodate wide characters.

XView also uses Compound Text encoding for transferring data between X clients.

1. The multibyte API is compatible with earlier versions of XView, such as domestic U.S. XView 3.1, which used ASCII or ISO Latin-1 characters.

For detailed discussions on encoding, refer to these documents:

- EUC, multibyte, and wide character: *Developer's Guide to Internationalization.*
- Compound Text: *Compound Text Encoding, Version 1.1, MIT X Consortium Standard, X Version 11, Release 5* by Robert W. Scheifler.

Encodings Used in Asian Locales

As you write your program, you will need to choose a suitable character encoding and API. Figure 4-1 shows how you can use different encodings (EUC wide character and multibyte, and Compound Text) within the same application.

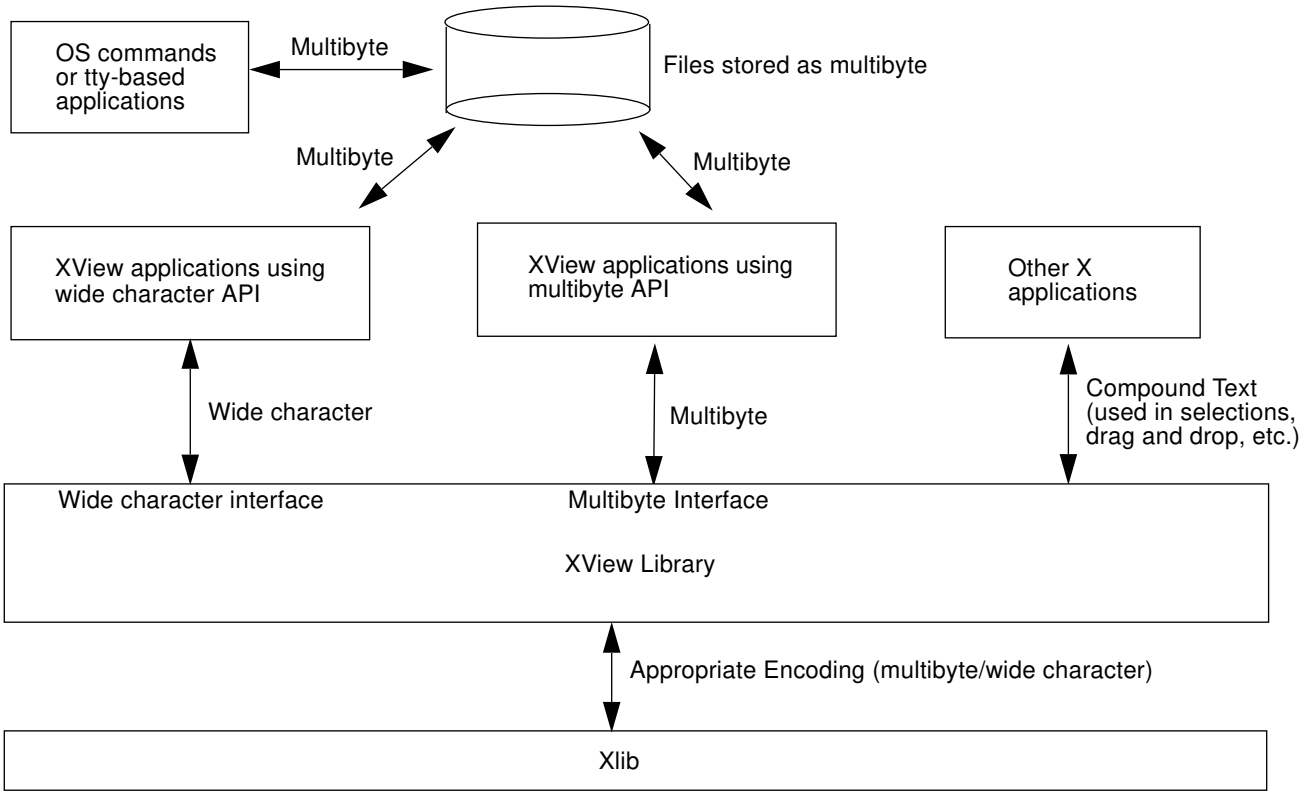

Figure 4-1 Encodings Used for Asian Locales

When to Use Multibyte and Wide Character

The wide character API (type `wchar_t`) consists of wide character string handling attributes and functions. The multibyte API (type `char`) is the same as in earlier, single-byte versions of XView.

You can mix wide character and multibyte characters within the same application. You can also mix wide character and multibyte APIs.

The XView library dynamically adjusts its internal data representation depending upon both the locale the application is running in and the nature of the data. As a result of this, programming convenience is the primary consideration regarding the choice of API.

When to Use Compound Text

The character encodings or character sets used in multibyte and wide character implementations may differ among vendors. For an application to communicate with other applications, a common encoding scheme is needed. XView relies on Compound Text, which is specified by the X Consortium.

Use Compound Text encoding in these situations:

- When an application needs to send a string composed of characters other than ISO Latin-1 across the X server to another application. XView uses Compound Text for data transfers to and from other X clients; for example, selection services (including drag and drop operations) and sending properties to the window manager
- When an application implements its own interclient communication; for example, a canvas-based application that uses selection service.

Note than an application is free to use a private encoding scheme for its own use, as long as the ICCCM is followed.

EUC Programming Issues

The following sections discuss special programming issues related to screen column definitions and passing multibyte strings.

Screen Columns

A screen column is defined as the pixel space required by a single ASCII character.[1] Asian characters may use a wider screen space than ASCII characters and are generally represented by more than one byte. Thus, in Asian locales:

screen columns != character count != byte count

Asian characters may also be interspersed with ASCII characters. In Asian locales, a fixed unit in pixels is needed to specify the space required by a screen column. Then wide Asian characters can occupy two or more columns, as in Figure 4-2.

Figure 4-2 ASCII and Japanese Characters

A number of functions and attributes use screen columns as arguments or returned values.

For example, `PANEL_VALUE_STORED_LENGTH` limits the number of characters that can be entered into a panel item. In Asian locales, `PANEL_VALUE_STORED_LENGTH` is measured in bytes. However, this attribute is screen-column based. If `PANEL_VALUE_STORED_LENGTH` and `PANEL_VALUE_DISPLAY_LENGTH` are specified to be 80, the user has allocated 80 screen columns, but not necessarily 80 characters, for display. In traditional Chinese, a Han character can be composed of 4 bytes and occupy 2 columns. Therefore, the `PANEL_VALUE_STORED_LENGTH` limit can be reached at 20 characters, yet only 40 screen columns are occupied.[2]

1. In previous releases (such as domestic U.S. XView 3.1), a screen column was defined as the space occupied by one character, and one character was represented by one byte. Therefore, the following used to apply:

```
screen columns==character count==byte count
```

Wide Character Attributes and Functions

Most XView multibyte attributes and functions that take a string or character as an argument have wide character analogs. These wide character attributes and functions have similar names composed of the original names suffixed with `_WCS`, `_WC`, `_wc`, or `_wcs`:

- `_WC` for wide character attributes
- `_wc` for wide character functions
- `_WCS` for wide character string attributes
- `_wcs` for wide character string functions

2. The screen column concept is only applicable in the case of fixed-width fonts. By default, the C locale uses fixed-width fonts for `textsw` and `ttysw`, and variable-width fonts for frame and panel. Currently, Asian locales use only fixed-width fonts.

4