duces the correct text encoding but it is not guaranteed as it is possible in Igor6 to make waves using any text encoding. For example, when running IgorJ you can switch the command line to an English font and create a wave whose name contains non-ASCII English characters. This name will therefore use MacRoman on Macintosh and Windows-1252 on Windows but IgorJ63x will record the wave name's text encoding as Shift JIS because that is the user interface text encoding. It is similarly possible to create a Japanese wave name when running an English Igor, causing Igor to record the wave name's text encoding as MacRoman or Windows-1252 even though the wave name is really Japanese.

Igor6.3x sets a wave's text encoding information when the wave is created. If a wave was created prior to Igor6.3x and is opened in Igor6.3x, its text encoding is unknown. Saving the experiment in Igor6.3x does not change this - the wave's text encoding is still unknown. So an Igor6.3x experiment can contain a mix of Igor6.3x waves with known text encodings and pre-Igor6.3x waves with unknown text encodings.

The three main causes of text encoding misidentification are:

- The file is a plain text file which contains no text encoding information
- The file is a pre-Igor6.3x experiment file which contains no text encoding information
- The Igor6.3x assumption that the text encoding can be deduced from the text encoding used to enter text into the user interface was incorrect

In order to convert text to UTF-8, Igor has to know the text encoding used when the text was created. If it gets this wrong then errors or garbled text will result. Some strategies for fixing such problems are described below.

Invalid Text Problems

A file that contains text in a particular text encoding can be damaged, creating sequences of bytes that are illegal in that text encoding.

In Shift JIS, for example, there are rules about how double-byte characters are formed. The first byte must be in a certain range of values and a second byte must be in a different range of values. If, by error, a line break is inserted between the first and second bytes, the resulting text is not valid as Shift JIS.

Damage like this can happen in Igor6 with long Japanese textboxes. When Igor generates a recreation macro for a graph, it generates TextBox commands for textboxes. If the text is long, Igor breaks it up into an initial TextBox command and subsequent AppendText commands. Igor6 is not smart enough to know that it must keep the two bytes of a double-byte Japanese characters together. This problem is asymptomatic in Igor6 because the two bytes of the double-byte character are reassembled when the macro runs. But it causes an error if you load the Igor6 file into Igor7 or later because Igor must convert the entire file to UTF-8 for internal storage before any reassembly occurs.

There are many other ways to damage a Shift JIS file. Similar damage can occur with other multi-byte text encodings such as UTF-8.

When a file contains invalid text but you suspect that it is mostly OK, you can use the Choose Text Encoding dialog to change the text encoding conversion error handling mode. This allows you to open the file despite the invalid text.

Mixed Text Encoding Problems

In Igor6, you can enter text in different text encodings by merely using different fonts. For example:

1. Run Igor6.3x with an English font controlling the built-in procedure window.
2. Create a graph.
3. Create an annotation using a Japanese font and Japanese characters in the graph.
4. Save the graph as a recreation macro.
5. Save the experiment.