waveList is a list of waves or it can be `allinCDF` to act on all waves in the current data folder.

See Also

WaveInfo to check if a wave is locked.

SetWaveTextEncoding

`SetWaveTextEncoding [`*`flags`*`] `*`newTextEncoding`*`, `*`elements`*`, [`*`wave`*`, `*`wave`*`, ...]`

The SetWaveTextEncoding operation changes the text encoding of the specified waves and/or the text encoding of all waves in the specified data folder.

Wave text encodings are mostly an issue in dealing with pre-Igor Pro 7 experiments containing non-ASCII text. Most users will have no need to worry about or change them. You should not use this operation unless you have a thorough understanding of text encoding issues or are instructed to use it by someone who has a thorough understanding.

See **Wave Text Encodings** on page III-472 for essential background information.

SetWaveTextEncoding can work on a list of specific waves or on all of the waves in a data folder (/DF flag). When working on a data folder, it can work on just the data folder itself or recursively on sub-data folders as well.

If /CONV is present, SetTextWaveEncoding actually converts the text to a different text encoding. You would use this, for example, to convert text stored as Shift JIS (Japanese non-Unicode) into UTF-8 (Unicode).

If /CONV is omitted, SetTextWaveEncoding merely causes Igor to reinterpret text. You would do this to tell Igor what text encoding is used for a wave created by Igor Pro 6 if Igor gets it wrong.

Conversion does not change the characters that make up text - it merely changes the numeric codes used to represent those characters. Reinterpretation does not change the numeric codes but does change the characters by changing the interpretation of the numeric codes.

In some cases it may be necessary to fix text encoding issues in Igor Pro 6.3x before opening an experiment in later versions.

Parameters

newTextEncoding specifies the text encoding to set the wave element to. See **Text Encoding Names and Codes** on page III-490 for a list of codes.

newTextEncoding can be the special value 255 which marks a text wave's content as really containing binary data, not text. See **Text Waves Containing Binary Data** on page III-475 below for details.

elements is a bitwise parameter that specifies one or more elements of a wave, as follows:

Bit	Value	Meaning
0	1	Wave name
1	2	Wave units
2	4	Wave note
3	8	Wave dimension labels
4	16	Text wave content

See **Setting Bit Parameters** on page IV-12 for details about bit settings.

wave, *wave*, ... is a list of the targeted waves. The list is optional and typically should be omitted if you use the /DF flag. However, if you specify a data folder via /DF and you also list specific waves, SetWaveTextEncoding works on all waves in the data folder as well as the specifically listed waves.

Flags

/BINA={*markAsBinary*, *diagnosticsFlags*}

If *markAsBinary* is 1, SetWaveTextEncoding marks the content of any text wave as binary if the data contains control characters (codes less than 32) other than carriage return (13), linefeed (10), and tab (9). The number of waves so marked is returned via V_numMarkedAsBinary.

If *markAsBinary* is 0, SetWaveTextEncoding acts as if /BINA were omitted.

diagnosticsFlags is optional and defaults to 1. If you omit it you can also omit the braces (/BINA=1).

diagnosticsFlags is a bitwise parameter defined as follows:

Bit 0: Emit diagnostic message for each wave marked as binary.

All other bits are reserved for future use.

See **Setting Bit Parameters** on page IV-12 for details about bit settings.

The /BINA=1 flag works with with the content of text waves only. It skips non-text waves. It is also independent of the *elements* parameter. That is, it marks the text wave content and only the text wave content as binary even if *elements* is something other than 16.

If you pass 255 for *newTextEncoding* when using the /BINA flag, this tells SetWaveTextEncoding to stop processing after marking binary waves. No further conversion or reinterpretation is done.

See **Text Waves Containing Binary Data** on page III-475 for further discussion.

/CONV={*errorMode* [, *defaultTextEncoding*, *diagnosticsFlags*, *conversionFlags*]}

Causes the text data to be converted to the specified text encoding. If /CONV is present, SetTextWaveEncoding actually converts the text. If it is omitted, SetTextWaveEncoding merely causes Igor to reinterpret it.

defaultTextEncoding, *diagnosticsFlags* and *conversionFlags* are optional and are further discussed below. If you omit them you can also omit the braces (/CONV=1).

If /CONV is specified and the original text encoding is binary (255) then SetWaveTextEncoding does nothing, because all conversions involving binary are NOPs.

If /CONV is specified and *newTextEncoding* is binary (255) then no conversion is done, because all conversions involving binary are NOPs, but the wave is marked as binary. Thus this amounts to the same thing as reinterpreting the wave's text data as binary.

If /CONV is specified, *defaultTextEncoding* is omitted or is -1, and the original text encoding is unknown (0) then SetWaveTextEncoding does nothing. That is, it does no reinterpretation or conversion.

errorMode determines how SetWaveTextEncoding behaves if the conversion can not be done because the text can not be mapped to the specified text encoding. This will occur if the text contains characters that can not be represented in the specified text encoding or if Igor's notion of the original text encoding is wrong. In the latter case, call SetWaveTextEncoding without /CONV to correct Igor's interpretation of the text and then call SetWaveTextEncoding with /CONV to do the conversion.

errorMode takes one of these values:

1: Generate error. SetWaveTextEncoding returns an error to Igor.

2: Use a substitute character for any unmappable characters. The substitute character for Unicode is the Unicode replacement character, U+FFFD. For most non-Unicode text encodings it is either control-Z or a question mark.

3: Skip unmappable input characters. Any unmappable characters will be missing in the output.

4: Use escape sequences representing any unmappable characters or invalid source text.

If the source text is valid in the source text encoding but can not be represented in the destination text encoding, unmappable characters are replaced with \uXXXX where XXXX specifies the UTF-16 code point of the unmappable character in hexadecimal.

If the conversion can not be done because the source text is not valid in the source text encoding, invalid bytes are replaced with \xXX where XX specifies the value of the invalid byte in hexadecimal.

defaultTextEncoding is optional. If it is present, not -1, and if the wave text element's original encoding is unknown (0), then the wave text element is treated as if it were the specified *defaultTextEncoding*. This allows you to convert the text of Igor Pro 6 waves that are set to unknown when you know that they are really some other text encoding. For example, if you know a wave's text data text encoding is Shift JIS, you can convert it to UTF-8 in one step, like this:

```
SetWaveTextEncoding /CONV={1,4} 1, 16, textWave0
```

Without the *defaultTextEncoding*, you would have to do two steps - the first to tell Igor what the real text encoding is and the second to do the conversion:

```
// Text encoding is Shift JIS
SetWaveTextEncoding 4, 16, textWave0

// Convert to UTF-8
SetWaveTextEncoding /CONV=1 1, 16, textWave0
```

Passing -1 for *defaultTextEncoding* acts the same as omitting it.

diagnosticsFlags is an optional bitwise parameter defined as follows:

Bit 0: Emit diagnostic message if text conversion succeeds.

Bit 1: Emit diagnostic message if text conversion fails.

Bit 2: Emit diagnostic message if text conversion is skipped.

Bit 3: Emit summary diagnostic message.

All other bits are reserved for future use and must be 0.

See **Setting Bit Parameters** on page IV-12 for details about bit settings.

diagnosticsFlags defaults to 2 (bits 1 set) if the /DF flag is not present and to 10 (bits 1 and 3 set) if the /DF flag is present.

Text conversion may be skipped because the wave element is marked as unknown text encoding, because it is marked as binary, because of the /ONLY or /SKIP flags, or because *newTextEncoding* is the same as the wave element's original text encoding.

conversionFlags is an optional bitwise parameter defined as follows:

Bit 0: Do a dry run only.

All other bits are reserved for future use and must be 0.

See **Setting Bit Parameters** on page IV-12 for details about bit settings.

The *conversionFlags* parameter was added in Igor Pro 9.00.

conversionFlags defaults to 0. If *conversionFlags* is 1, SetWaveTextEncoding goes through the entire conversion process but does not actually do any conversions. This feature was added to provide a way to determine if conversions are needed to create a pure UTF-8 experiment. For most purposes you can omit the *conversionFlags* parameter.

/DF={*dfr*, *recurse*, *excludedDFR*}

dfr is a reference to a data folder. SetWaveTextEncoding operate on all waves in the specified data folder. If *dfr* is null ($"") SetWaveTextEncoding acts as if /DF was omitted.

If *recurse* is 1, SetWaveTextEncoding works recursively on all sub-data folders. Otherwise it affects only the data folder referenced by *dfr*.

excludedDFR is an optional reference to a data folder to be skipped by SetWaveTextEncoding. For example, this command sets the text encoding of text wave data for all waves in all data folders except for root:Packages and its sub-data folders:

```
SetWaveTextEncoding /DF={root:,1,root:Packages} 1, 16
```

If *excludedDFR* is null ($"") SetWaveTextEncoding acts as if *excludedDFR* was omitted and no data folders are excluded.

/KIND=*kind* SetWaveTextEncoding changes only waves of the specified kind. *kind* is:

1: Home waves only

2: Shared waves only

3: Home and shared waves (default)

A "home wave" is a wave that is stored in the experiment file for packed experiment or in the disk folder corresponding to the wave's data folder for an unpacked experiment.

A "shared wave" is a wave that is stored in a standalone file for packed experiment or in a standalone file outside the disk folder corresponding to the wave's data folder for an unpacked experiment.

See **References to Files and Folders** on page II-24 for an explanation of shared versus home waves.

/ONLY=*targetedTextEncoding*

SetWaveTextEncoding changes only wave elements that are currently set to *targetedTextEncoding*. For example, this command converts the content of all waves that are currently set to unknown text encoding (0) to UTF-8 (1), treating the waves originally marked as unknown as Shift JIS (4) waves:

```
SetWaveTextEncoding /DF={root:,1} /ONLY=0 /CONV={1,4} 1, 16
```

Passing -1 for *targetedTextEncoding* acts as if you omitted /ONLY altogether.

/SKIP=*skipTextEncoding*

SetWaveTextEncoding skips all wave elements that are currently set to *skipTextEncoding*. For example, this command converts the content of all waves to UTF-8 (1) except those that are set to Japanese (4):

```
SetWaveTextEncoding /DF={root:,1} /SKIP=4 /CONV=1 1, 16
```

As explained under /CONV, binary (255) wave elements are always skipped and unknown (0) wave elements are skipped if the /CONV defaultTextEncoding parameter is omitted.

Passing -1 for *skipTextEncoding* acts as if you omitted /SKIP altogether.

/TYPE=*type* SetWaveTextEncoding changes only waves of the specified type.

type is:

1:	Text waves only
2:	Non-text waves only
3:	All waves (default)

/Z[=*z*] Prevents procedure execution from aborting if SetWaveTextEncoding generates an error. Use /Z or the equivalent, /Z=1, if you want to handle errors in your procedures rather than having execution abort.

/Z does not suppress invalid parameter errors. It suppresses only errors in doing text encoding reinterpretation or conversion.

Details

Because SetWaveTextEncoding is intended for use by experts or by users instructed by experts, it does not respect the lock state of waves. That is, it will change waves even if they are locked using SetWaveLock.

For background information on wave text encodings, see **Wave Text Encodings** on page III-472.

A wave's text wave content text encoding can also be set to the special value 255. This marks a text wave as really containing binary data, not text. See **Text Waves Containing Binary Data** on page III-475 for details.

Using SetWaveTextEncoding

One of the main uses for SetWaveTextEncoding is to set the encoding settings to some value other than 0 (unknown) so that Igor's idea of the text encoding used for the items accurately reflects the actual text encoding. We call this "reinterpreting" the item. It does not change the numeric codes representing the text but rather just changes the setting that controls Igor's idea of how the text is encoded. This does change the meaning of the underlying numeric codes. In other words, it changes the characters represented by the text.

You would do reinterpretation if you load an Igor6 experiment and Igor interprets the waves using the wrong text encoding. For example, if you load an experiment that you know uses Shift JIS encoding but Igor interprets it as Windows-1252, you get garbage for Japanese text. Reinterpreting it as Shift JIS fixes this. An example is provided below.

The other use for SetWaveTextEncoding is to actually change the numeric codes representing the text - i.e., to convert the content to a different text encoding. For example, if you have text waves from Igor Pro 6 that are encoded in Japanese (Shift JIS), you may want to convert the text to UTF-8 (a form of Unicode) so that you can combine Japanese and non-Japanese characters or use other features that require Unicode. This also applies to western text containing non-ASCII characters encoded as MacRoman or Windows-1252. Converting changes the underlying numeric codes but does not change the characters represented by the text.

The main use for converting text is to convert Igor Pro 6 waves from whatever encoding they use, which typically will be MacRoman, Windows-1252, or Shift JIS, to UTF-8 (a form of Unicode) which is a more modern representation but is not backward compatible with Igor Pro 6.

If the /CONV flag is omitted SetWaveTextEncoding does reinterpretation. If the /CONV flag is present then SetWaveTextEncoding does text conversion except if the original text encoding is unknown (0) or binary (255) in which case it does nothing.

If a wave has mistakenly been marked as containing binary, use SetTextWaveEncoding without /CONV to set it to the correct text encoding.

If a wave's text encoding is set to unknown (0) but you know that it really contains text in some specific encoding, you can use SetTextWaveEncoding without /CONV to set it to the correct text encoding and then use SetTextWaveEncoding with /CONV to convert it to the desired final text encoding. Alternately you can combine these two steps by using /CONV and providing a value for the optional *defaultTextEncoding* parameter.

Output Variables

The SetWaveTextEncoding operation returns information in the following variables:

`V_numConversionsSucceeded`	Set only when the /CONV flag is used. Zero otherwise. V_numConversionsSucceeded is set to the number of successful text element conversions.
`V_numConversionsFailed`	Set only when the /CONV flag is used. Zero otherwise. V_numConversionsFailed is set to the number of unsuccessful text conversions. Because SetWaveTextEncoding quits when any error occurs, V_numConversionsFailed will be either 0 or 1.
`V_numConversionsSkipped`	Set only when the /CONV flag is used. Zero otherwise. V_numConversionsSkipped is set to the number of skipped text element conversions. Text conversion may be skipped because the wave element is marked as unknown text encoding, because it is marked as binary, because of the /ONLY or /SKIP flags or because newTextEncoding is the same as the wave element's original text encoding. V_numConversionsSkipped does not count waves skipped because of the /TYPE flag.
`V_numMarkedAsBinary`	Set only when the /BINA flag is used. Zero otherwise. V_numMarkedAsBinary is set to the number of text waves whose text wave content was marked as binary. V_numMarkedAsBinary was added in Igor Pro 9.00.
`V_numWavesAffected`	V_numWavesAffected is set to the number of waves modified in any way, whether through reinterpretation, conversion of one or more elements, or marking as binary. V_numWavesAffected was added in Igor Pro 9.00.

Examples

```
// Keep in mind that, if /CONV is present, SetWaveTextEncoding does nothing
// for wave text elements currently set to binary (255).
// Also, if /CONV is present, SetWaveTextEncoding does nothing
// for wave text elements currently set to unknown (0) if the /CONV
// optional defaultTextEncoding parameter is omitted.
// In the following examples 1 means UTF-8, 4 means Shift JIS, and 16
// means text wave content.

// Reinterpret specific text waves as Shift JIS if they are currently set to unknown
SetWaveTextEncoding /ONLY=0 4, 16, textWave0, textWave1

// Convert specific waves' content to UTF-8
SetWaveTextEncoding /CONV=1 1, 16, textWave0, textWave1

// Reinterpret all text waves as Shift JIS if they are currently set to unknown
SetWaveTextEncoding /DF={root:,1} /ONLY=0 4, 16

// Convert all waves' content to UTF-8
SetWaveTextEncoding /DF={root:,1} /CONV=1 1, 16

// Convert all text waves' content from Shift JIS to UTF-8
// if it is currently set to unknown
SetWaveTextEncoding /DF={root:,1} /ONLY=0 /TYPE=1 /CONV={1,4} 1, 16

// Same as before but exclude the root:Packages data folder
```