Save

`Save [`*`flags`*`]` ***`waveList`*** **`[as`** ***`fileNameStr`*****`]`**

The Save operation saves the named waves to disk as text (/F, /G or /J) or as Igor binary.

Parameters

waveList is either a list of wave names or, if the /B flag is present, a string list of references to waves. For example, the following commands are equivalent, assuming that the waves in question are in the root data folder and root is the current data folder:

```
Save/J wave0,wave1 as "Test.dat"
Save/J root:wave0,root:wave1 as "Test.dat"
Save/J/B "wave0;wave1;" as "Test.dat"
Save/J/B "root:wave0;root:wave1;" as "Test.dat"
String list="root:wave0;root:wave1;"; Save/J/B list as "Test.dat"
```

The form using the /B flag and a string containing a list of references to waves saves a very large number of waves using one command. This is not possible using a list of wave names because of the 2500 byte command line length limit. When using this form, the string must contain semicolon-separated wave names or data folder paths leading to waves. Liberal names in the string may be quoted or unquoted.

The file to be written is specified by *fileNameStr* and /P=*pathName* where *pathName* is the name of an Igor symbolic path. *fileNameStr* can be a full path to the file, in which case /P is not needed, a partial path relative to the folder associated with *pathName*, or the name of a file in the folder associated with *pathName*. If it cannot determine the location of the file from *fileNameStr* and *pathName*, it displays a dialog allowing you to specify the file.

If you use a full or partial path for *fileNameStr*, see **Path Separators** on page III-451 for details on forming the path.

Flags

/A[=*a*]	Appends to the file rather than overwriting it (with /T, /G or /J). *a*=0: Does not append. *a*=1: Appends to the file with a blank line before the appended data (same as /A only). *a*=2: Appends to the file with no blank line before the appended data.
/B	The *waveList* parameter is a string containing a list of references to waves instead of a literal list of waves.
/C	Saves a copy of the wave when saving as Igor binary.
/DLIM=*delimStr*	Specifies the string to use as a column delimiter. This flag affects general text saves (/G) and delimited text saves (/J) only. *delimStr* defaults to tab. It can consist of multiple characters. If you choose a delimiter that also appears in the data you are saving, for example if you choose to save text waves containing commas using comma as the separator, the resulting file is likely to be misinterpreted by any software loading it. /DLIM was added in Igor Pro 7.00.
/DSYM=*dsStr*	Specifies a string containing the character to use as the decimal symbol for all numbers (default is a period). If *dsStr* is empty (`""`), then the decimal symbol is as defined in system preferences as of when Igor was launched.
/E=*useEscapeCodes*	Determines whether to use escape sequences for special characters. /E=1: Converts carriage-return, linefeed, tab, and backslash characters to escape sequences when writing general or delimited text files (default; same as no /E). /E=0: No escape sequences used in general or delimited text files. When saving text waves containing backslashes (such as Windows paths) in a file intended for another program, you probably should use /E=0.

/F	Writes delimited and general text files with numeric formatting as it appears in the top table. Has no effect if there is no top table or if the wave being saved does not appear in the top table.

Note: The text written to the file is exactly as displayed in the table. Set the table to display as many digits of precision as you want in the file.

Note: Fractional and out-of-range floating point wave data can not be formatted as octal or hex. See **Octal Numeric Formats** on page II-257 and **Hexadecimal Numeric Formats** on page II-257 for details.

Note: When saving a multi-column wave (1D complex wave or multidimensional wave), all columns of the wave are saved using the table format for the first table column from the wave.

/G	Saves waves in general text format.

/H	"Adopts" the waves specified by *waveList*.

"Adopt" means that any connection between the waves and external files is severed. The waves become part of the current experiment. When the experiment is next saved, the waves are saved in the experiment file (for an packed experiment) or in the experiment folder (for an unpacked experiment).

When you use the /H flag, all other flags and the *fileNameStr* parameter are ignored. The wave is not actually saved but rather is marked for saving as part of the current experiment.

You would normally do this to make an experiment more self-contained which makes it easier to send to other people. See **Sharing Versus Copying Igor Binary Wave Files** on page II-156 and the **LoadWave** /H flag.

/I	Presents a dialog from which you can specify file name and folder.

/J	Saves waves in delimited text format.

The delimiter defaults to tab unless you specify another delimiter using /DLIM.

/M=*termStr*	Specifies the terminator character or characters to use at the end of each line of text. The default is `/M="\r"`, which uses a carriage return character. This is the Macintosh convention. To use the Windows convention, carriage return plus linefeed, specify `/M="\r\n"`. To use the Unix convention, just a linefeed, specify `/M="\n"`.

/O	Overwrites file if it exists already.

/P=*pathName*	Specifies the folder to store the file in. *pathName* is the name of an existing symbolic path.

/T	Saves waves in Igor Text format.

/U={*writeRowLabels*, *rowPositionAction*, *writeColLabels*, *colPositionAction*}

These parameters affect the saving of a matrix (2D wave) to a delimited text (/J) or general text (/G) file. They are accepted no matter what the save type is but are ignored when they don't apply.

If *writeRowLabels* is nonzero, Save writes the row labels of the matrix as the first column of data in the file.

rowPositionAction has one of the following values:

0:	Don't write a row position column.
1:	Writes a row position column based on the row scaling of the matrix wave.
2:	Writes a row position column based on the contents of the row position wave for the matrix. The row position wave is an optional 1D wave whose name is the same as the matrix wave but with the prefix "RP_".

	writeColLabels and *colPositionAction* have analogous meanings. The prefix used for the column position wave is "CP_". See Chapter II-9, **Importing and Exporting Data**, for further details.
/W	Saves wave names (with /G or /J).

Details

The Save operation saves only the named waves; it does not save the entire experiment.

Waves saved in Igor binary format are saved one wave per file. If you are saving more than one wave, you must not specify a *fileNameStr*. Save will give each file a name which consists of the wave name concatenated with ".ibw".

When you save a wave as Igor binary, unless you use the /C flag to save a copy, the current experiment subsequently references the file to which the wave was saved. See **References to Files and Folders** on page II-24 for details.

In a general text file (/G), waves with different numbers of points are saved in different groups. Waves with different precisions and number types are saved in same group if they have the same number of points.

In a delimited text file (/J), all waves are saved in one group whether or not they have the same number of points.

If you save multiple 2D waves, the blocks of data are written one after the other.

If you save 3D waves, the data for each wave is written as a contiguous block having as many columns as there are columns in the wave, and R*L rows, where R is the number of rows in the multidimensional wave and L is the number of layers. All rows for layer 0 are saved followed by all rows for layer 1, and so on.

If you save 4D waves, the data for each wave is written as a contiguous block having R*L*C rows, where R is the number of rows, L is the number of layers and C is the number of chunks. Igor writes all data for chunk 0 followed by all data for chunk 1, and so on.

The Save operation will always present a save dialog if you try to save to an existing file without using the overwrite flag.

Here are some details about saving an Igor binary wave file.

If you omit the path or the file name, the Save operation will normally present a save dialog. However, if the wave has already been saved to a stand-alone file and if you use the overwrite flag, it will save the wave to the same file without a dialog. Also, if the wave has never been saved and the current experiment is an unpacked experiment, it will save to the home folder without a dialog.

If you use /P=*pathName*, note that it is the name of an Igor symbolic path, created via **NewPath**. It is not a file system path like "`hd:Folder1:`" or "`C:\\Folder1\\`". See **Symbolic Paths** on page II-22 for details.

Examples

This function uses the string list of references to waves to save some or all of the waves in the current data folder:

```
Function SaveWavesFromCurrentDF(matchStr)
   String matchStr                  // As for the WaveList function.

   String list
   list = WaveList(matchStr, ";", "")
   Save/O/J/W/I/B list
End
```

For example, to save all of the waves in the current data folder, execute:

```
SaveWavesFromCurrentDF("*")
```

To save those waves in the current data folder whose name starts with "wave", execute:

```
SaveWavesFromCurrentDF("wave*")
```

This function saves all of the waves used in a particular graph:

```
Function SaveWavesFromGraph(graphName)      // Saves all waves in graph.
   String graphName                         // "" for top graph.

   String list, traceList, traceName
   Variable index = 0
   list = ""
   traceList = TraceNameList(graphName, ";", 1)
```