objNameStr can optionally include a module name or independent module name prefix such as "ProcGlobal#" to check for the existence of functions. This works for macros as well.

The return values are:

0: Name not in use, or does not conflict with a wave, numeric variable or string variable in the specified data folder.

1: Name of a wave in the specified data folder.

2: Name of a numeric or string variable in the specified data folder.

3: Function name.

4: Operation name.

5: Macro name.

6: User-defined function name.

exists is not aware of local variables or parameters in user-defined functions, however it is aware of local variables and parameters in macros.

objNameStr is a string or string expression, *not* a name.

Examples

```
// Prints 2 if V_flag exists as a global variable in the current data folder:
Print exists("V_Flag")

// Prints 5 if a macro named Graph0 exists.
Print exists("ProcGlobal#Graph0")
```

See Also

The **DataFolderExists** and **WaveExists** functions and the **WinType** operation.

exp

exp(*num*)

The exp function returns $e^{num.}$ In complex expressions, *num* is complex, and exp(*num*) returns a complex value.

ExperimentInfo

ExperimentInfo [/Q[=*quiet*]] [keyword=*value* [, keyword=*value* ...]]

The ExperimentInfo operation returns information about the current Igor experiment. The information is returned in the output variable V_Flag and the output string variable S_Value, as described below.

ExperimentInfo was added in Igor Pro 8.00.

To get the name of the current experiment, use **IgorInfo**(1).

Flags

/Q[=*quiet*] If you omit /Q or pass 0 for quiet , ExperimentInfo prints information to the history area of the command window.

If you specify /Q or /Q=1, ExperimentInfo prints nothing to the history area. Information is still returned via V_Flag and S_Value.

Keywords

getRequiredIgorVersion Returns the version of Igor required to open the current experiment via V_Flag. Returns a message explaining the requirement via S_Value.

See **Getting the Required Igor Version** on page V-210 for details.

getLongNameUsage[={*dfRef* , *mask*, *options*, *length*}

Returns information about long object names used in the current experiment via V_Flag and S_Value. This may be of interest because experiments that use long object names (longer than 31 bytes) require Igor Pro 8.00 or later.

V_Flag is set to the number of long names found.

S_Value is set to a description of the long names found. If you omit /Q or specify /Q=0, the description is also printed to the history area of the command window.

All of the parameters are optional. However, if you specify any of them, you must specify all of them.

dfRef specifies the data folder to start from and defaults to root:. It is a full or partial path to the starting data folder. You can specify root: or * for *dfRef* to get the default. You can specify : to get the current data folder.

getLongNameUsage reports on the contents of the data folder specified by *dfRef* . Consequently, the specified data folder itself is not reported even if its name is long.

mask specifies what types of long object names you want to know about. It is a bitwise parameter defined as follows:

Bit 0:	Wave names
Bit 1:	Variable names
Bit 2:	Data folder names
Bit 3:	Target window names
Bit 4:	Symbolic path names

mask defaults to 31 which specifies all of the above.

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

options controls various aspects of the returned information. It is a bitwise parameter defined as follows:

Bit 0:	Produce text formatted for reading by humans
Bit 1:	Include headers if bit 0 is set
Bit 2:	Use full paths for waves, variables and data folders
Bit 3:	Use full paths for waves, variables and data folders

options defaults to 15.

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

length specifies the length in bytes greater than which a name must be to be included in the output. It defaults to 31. If you specify 0 for *length* then waves are listed regardless of the length of their name.

See **Getting Long Name Usage Info** on page V-210 for details.

getLongDimensionLabelUsage[={*dfRef* , *mask*, *options*, *length*}

Returns information about waves with long dimension labels in the current experiment via V_Flag and S_Value. This may be of interest because such experiments require Igor Pro 8.00 or later.

V_Flag is set to the number of waves with long dimension labels found.

S_Value is set to a description of the waves found. If you omit /Q or specify /Q=0, the description is also printed to the history area of the command window.

All of the parameters are optional. However, if you specify any of them, you must specify all of them.

dfRef specifies the data folder to start from and defaults to root:. It is a full or partial path to the starting data folder. You can specify root: or * for *dfRef* to get the default. You can specify : to get the current data folder.

mask determines if data folders are listed along with waves with long dimension labels to provide context. It is a bitwise parameter defined as follows:

Bit 2:	Include data folder names

All other bits are ignored.

mask defaults to 4 which means that bit 2 is set and data folders are listed, but they are listed only if bit 0 of the options parameter is also set.

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

options controls various aspects of the returned information. It is a bitwise parameter defined as follows:

Bit 0:	Produce text formatted for reading by humans
Bit 1:	Include headers if bit 0 is set
Bit 2:	Use full paths for waves, variables and data folders
Bit 3:	Use full paths for waves, variables and data folders

options defaults to 15.

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

length specifies the length in bytes greater than which a dimension label must be for the wave be included in the output. It defaults to 31. If you specify 0 for *length* then waves are listed regardless of the length of their dimension labels.

Getting the Required Igor Version

The getRequiredIgorVersion keyword returns the version of Igor required to open the current experiment via V_Flag. If no particular version is required, it returns 0. If V_Flag is non-zero, S_Value contains an explanation of why a particular version of Igor is required.

As used here, the concept of the version of Igor required to open the current experiment is very narrow. If getRequiredIgorVersion returns a non-zero value via V_Flag, this means that the current experiment can not be opened at all in versions of Igor earlier than specified by V_Flag. Currently the only situation that getRequiredIgorVersion flags is the use of long object names, specifically long wave, variable, data folder, target window, and symbolic path names, or long wave dimension labels. If the experiment uses a long name or long dimension label, the experiment can not be opened in versions of Igor prior to 8.00, and getRequiredIgorVersion sets V_Flag to 8.00. See **Long Object Names** on page III-502 and **Long Dimension Labels** on page II-94 for background information.

If getRequiredIgorVersion returns zero, it means that the current experiment can be opened in earlier versions of Igor. However, it does not guarantee that the experiment will open without errors. For example, if you use a function or operation added in Igor Pro 8.00 in a procedure, you can open the experiment in earlier versions, but you will get a compile error when you do so. getRequiredIgorVersion does not flag the use of all Igor features that require a particular version. It flags only the specific situation explained in the previous paragraph.

Getting Long Name Usage Info

For background information on long object names, see **Long Object Names** on page III-502.

Experiments that use long object names (longer than 31 bytes), require Igor Pro 8.00 or later. You can use the ExperimentInfo operation with the getLongNameUsage keyword to determine what long object names, if any, the current experiment uses. This will help if you want to modify the experiment so it can be opened in an earlier version of Igor.

The getLongNameUsage keyword reports information about long wave, variable, data folder, target window, and symbolic path names. It does not report information about long axis, annotation, control, special character, or XOP names or about long dimension labels.

getLongNameUsage returns information via the V_Flag and S_Value output variables and, if you omit /Q or specify /Q=0, it prints information to the history area of the command window.

In human-readable mode (bit 0 of options parameter set), if you include data folders in the output (bit 2 of mask parameter set), ExperimentInfo generates output for each data folder, whether its name is long or not, and uses indentation. This allows you to see any long names in the context of the entire data folder hierarchy.

If human-readable mode is off, ExperimentInfo generates output for a given data folder only if its name is long. This mode is intended for use by code rather than reading by humans. In this mode, the hierarchy (i.e., which data object exists in which data folder) is discernible only if you call for full paths.

If you specify a starting data folder using the *dfRef* parameter, and if human-readable mode is off, ExperimentInfo does not generate output for the starting data folder name, even if it is long. This is because ExperimentInfo generates output for the contents of the starting data folder.

Liberal names are quoted if you specify full paths (bit 1 of options parameter set) but unquoted if you request simple names only. The reason for not quoting simple names is explained under **Accessing Global Variables and Waves Using Liberal Names** on page IV-68.

GetLongNameUsage Examples

```
// Display long names for all data objects (waves, variables, data folders)
// and for target windows an symbolic paths.
ExperimentInfo getLongNameUsage

// Print number of long names uses and nothing else
ExperimentInfo/Q getLongNameUsage; Print V_Flag

// More specialized tests - paste the following into the procedure window
// of an experiment that uses long names and execute DemoGetLongNamesUsage()

Constant kLongNamesLength = 31
Constant kHumanMask = 1             // Human mode
Constant kHeadersMask = 2           // Include headers
Constant kFullPathsMask = 4         // Use full paths
Constant kRecursiveMask = 8         // Recursive

Function DemoGetLongNamesUsage()
    Print "=== Data folders, human, names only ==="
    ExperimentInfo getLongNameUsage={*, 4, kHumanMask|kRecursiveMask, kLongNamesLength}
    Printf "\r"

    Print "=== Data folders, human, full paths ==="
    ExperimentInfo getLongNameUsage={*, 4, kHumanMask|kFullPathsMask | kRecursiveMask,
    kLongNamesLength}
    Printf "\r"

    Print "=== Data folders, non-human, names only ==="
    ExperimentInfo getLongNameUsage={*, 4, 0|kRecursiveMask, kLongNamesLength}
    Printf "\r"

    Print "=== Data folders, non-human, full paths ==="
    ExperimentInfo getLongNameUsage={*, 4, kFullPathsMask|kRecursiveMask,
    kLongNamesLength}
    Printf "\r"

    Print "=== All data objects, human, names only ==="
    ExperimentInfo getLongNameUsage={*, 7, kHumanMask|kRecursiveMask, kLongNamesLength}
    Printf "\r"

    Print "=== All data objects, human, full paths ==="
    ExperimentInfo getLongNameUsage={*, 7, kHumanMask|kFullPathsMask | kRecursiveMask,
    kLongNamesLength}
    Printf "\r"

    Print "=== All data objects, non-human, names only ==="
    ExperimentInfo getLongNameUsage={*, 7, 0|kRecursiveMask, kLongNamesLength}
    Printf "\r"

    Print "=== All data objects, non-human, full paths ==="
    ExperimentInfo getLongNameUsage={*, 7, kFullPathsMask|kRecursiveMask,
    kLongNamesLength}
    Printf "\r"
```