AbortOnValue. A user abort is generated when the user clicks the Abort button or presses the user abort key combination.

When code executes in the try-catch area, a programmatic abort immediately jumps to the code in the catch-endtry area rather than jumping to the end of the user function. A user abort jumps to the catch-endtry area when a flow control keyword such as for or while executes or at the end of the try code. Normal flow (no aborts) skips all code within the catch-endtry area.

Details

During execution of code in the catch-endtry area, user aborts are suppressed. This means that, if the user attempts to abort procedure execution by pressing the **User Abort Key Combinations** or by clicking the Abort button, this will not abort the catch code itself.

When an abort occurs, information about the cause of the abort is returned via the `V_AbortCode` variable as follows:

-4:	Abort triggered by AbortOnRTE.
-3:	Abort caused by Abort operation.
-2:	Stack overflow abort.
-1:	User abort.
>=1:	Abort triggered by AbortOnValue.

See Also

Flow Control for Aborts on page IV-48 and **try-catch-endtry Flow Control** on page IV-49 for further details.

The **AbortOnRTE** and **AbortOnValue** keywords, and the **Abort** operation.

UInt64

`uint64` ***`localName`***

Declares a local unsigned 64-bit integer in a user-defined function or structure.

UInt64 is available in Igor Pro 7 and later. See **Integer Expressions** on page IV-38 for details.

See Also

Int, **Int64**

UniqueName

`UniqueName(`*`baseName`*`, `*`objectType`*`, `*`startSuffix`*`[, `*`windowNameStr`*`])`

The UniqueName function returns the concatenation of *baseName* and a number such that the result is not in conflict with any other object name.

windowNameStr is optional. If missing, it is taken to be the top graph, panel, layout, or notebook according to the value of *objectType*.

In Igor Pro 9.00 or later, you can use the **CreateDataObjectName** function as a replacement for some combination of CheckName, CleanupName, and UniqueName to create names of waves, global variables, and data folders.

Details

baseName should be an unquoted name, such as you might receive from the user via a dialog or control panel.

objectType is one of the following:

1	Wave
2	Reserved
3	Numeric variable
4	String variable
5	XOP target window
6	Graph window
7	Table window
8	Layout window
9	Control panel window
10	Notebook window
11	Data folder
12	Symbolic path
13	Picture
14	Annotation in the named or topmost graph or layout
15	Control in the named or topmost graph or panel
16	Notebook action character in the named or topmost notebook
17	Gizmo window (added in Igor Pro 9.00)

startSuffix is the number used as a starting point when generating the numeric suffix that makes the name unique. Normally you should pass zero for startSuffix. If you know that names of the form base0 through baseN are in use, you can make UniqueName run a bit faster by passing `N+1` as the *startSuffix*.

The *windowNameStr* argument is used only with object types 14, 15, and 16. The returned name is unique only to the window (other windows might have objects with the same name). If a named window is given but does not exist, UniqueName returns *baseName startSuffix*. *windowNameStr* is ignored for other object types.

The Main Namespace

Waves, numeric variables, string variables, built-in and external functions, built-in and external operations, user-defined functions, macros and reserved keywords exist in the main namespace. The names of each of those objects must be unique in that namespace.

Window Namespace

Values of *objectType* from 5 through 10 refer to the window namespace.

The expression

```
CheckName("<name>", x)     // where x is 5, 6, 7, 8, 9 or 10
```

returns 0 if the specified name is syntactically legal and unique in the window namespace.

That does not guarantee that the name is allowed as a window name because the DoWindow/C operation, which changes a window's name, requires that the name also not conflict with objects in the main namespace. Consequently, to determine if a name is allowed as a window name, use this:

```
// Window names must be unique in both the window and main namespaces
Variable nameOK = CheckName("<name>",6)==0 && CheckName("<name>",1)==0
```

Window Macro Names

Window macro names are treated different from other main namespace names. DoWindow/C allows you to change the name of a window to the name of an existing window macro but not to any other name in the main namespace. Also this expression:

```
CheckName("<name>", x)     // where x is 1, 3, 4 (main namespace)
```

returns 0 even if the specified name is used as a window macro name.

UniqueName Thread Safety

As of Igor Pro 8.00, you can call UniqueName from an Igor preemptive thread but only if *objectType* is 1 (wave), 3 (global numeric variable), 4, (global string variable), 11 (data folder), or 12 (symbolic path). For any other value of *objectType*, UniqueName returns a runtime error.