Lightweight Packages

A lightweight package is one that consists of at most a few procedure files and does not create clutter in the current experiment unless it is actually used.

If your package is lightweight you might prefer to dispense with loading and unload it and just keep it loaded all the time. To do this you would organize your files like this:

```
Igor Pro User Files
   Your Package
      Your Package Part 1.ipf
      Your Package Part 2.ipf
   Igor Procedures
      Alias or shortcut pointing to the "Your Package" folder
```

Here both of your package procedure files are global, meaning that Igor loads them at launch time and never unloads them. You do not need procedures for loading and unloading your package.

If you have an ultra-light package, consisting of just a single procedure file, you can dispense with the "Your Package" folder and put the procedure file directly in the Igor Procedures folder.

Managing Package Data

When you create a package of procedures, you need some place to store private data used by the package to keep track of its state. It's important to keep this data separate from the user's data to avoid clutter and to protect your data from inadvertent changes.

Private data should be stored in a data folder named after the package inside a generic data folder named Packages. For example, if your package is named My Package you would store your private data in `root:Packages:My Package`.

There are two general types of private data that you might need to store: overall package data and per-instance data. For example, for a data acquisition package, you may need to store data describing the state of the acquisition as a whole and other data on a per-channel basis.

Creating and Accessing the Package Data Folder

This section demonstrates the recommended way to create and access a package data folder. We use a bottleneck function that returns a **DFREF** for the package data folder. If the package data folder does not yet exist, the bottleneck function creates and initializes it. This way calling functions don't need to worry about whether the package data folder has been created.

First we write a function to create and initialize the package data folder:

```
Function/DF CreatePackageData()      // Called only from GetPackageDFREF
   // Create the package data folder
   NewDataFolder/O root:Packages
   NewDataFolder/O root:Packages:'My Package'

   // Create a data folder reference variable
   DFREF dfr = root:Packages:'My Package'

   // Create and initialize package data
   Variable/G dfr:gVar1 = 1.0
   String/G dfr:gStr1 = “hello”
   Make/O dfr:wave1
   WAVE wave1 = dfr:wave1
   wave1= x^2

   return dfr
End
```