NAMED PARAMETERS:

I neglected to describe the main reason why I liked to use a stored procedure to call the RPG logic. Parameters can be named with functions and stored procedures. Here are a few brief nores.

Named Parameters allow for greater flexibility...

- When calling an RPG program with *NOPASS, you *cannot skip parameters*. For instance, you cannot skip parameter 3 and still supply parameter 4.
- "=>" does not mean "*equal to or greater*", it is a means to equate the named parameter to a value.
    - It should be read as ***parameter "String_Separator" has the value '"'***.

Rules...

- Just like in commands, if you do not name a parameter, it must be referred to in the sequence that it is defined, and comma separated.
- Just like in commands, if you name the parameters, you are free to put them in any order.
- Unlike RPG, naming the parameters allows skipping a parameter.

The following two calls are effectively the same...

```
call derrja.spSQLtoIFS(
    SQLStatement          => 'SELECT * FROM DERRJA.EMPLOYEE',
    Include_Header_Yn     => 'N',
    FileLocation          => '/home/Testdescriptor/testEmployee.csv',
    String_Separator      => '"',
    Column_Separator      => ',');

call derrja.spSQLtoIFS(
    '/home/Testdescriptor/testEmployee.csv',
    'SELECT * FROM DERRJA.EMPLOYEE', 'N', '"', ',');
```

The same file location and SQL statement, but allowing the defaults to occur on the remaining parameters.

```
call derrja.spSQLtoIFS(
    SQLStatement          => 'SELECT * FROM DERRJA.EMPLOYEE',
    FileLocation          => '/home/Testdescriptor/testEmployee.csv');
```

The same file location and SQL statement, except that the third parameter is omitted and defaulted, and the fourth is supplied.

```
call derrja.spSQLtoIFS(
    SQLStatement          => 'SELECT * FROM DERRJA.EMPLOYEE',
    FileLocation          => '/home/Testdescriptor/testEmployee.csv')
    String_Separator      => '$';     (I know, why would you ever do this?)
```