Scope: BCDataSet extension

A current limitation in the SIDS is that GridLocation (along with PointList or PointRange) is defined under BC_t. Thus, if a user wishes to write a BC dataset under BCDataSet_t, he is forced to write it at the exact same locations. There are many circumstances when a user would wish to define the BC patch using Vertex indices (under BC_t), but then write data at FaceCenter indices (under BCDataSet_t). This proposal addresses this need.

We propose to allow the following (optional) nodes under BCDataSet_t:

```
        PointRange IndexRange_t
        PointList IndexArray_t
        GridLocation GridLocation_t
```

Since there are currently no separate functions to write the PointRange/PointList nodes (the arguments are always passed with the function that creates the parent node itself), new MLL functions must be created.

One way to do this would be to create 3 new functions for writing/reading a point set. (See also proposal for UserDefined extension.) These might look like:

```
write:
        cg_ptset_write(PointSetType_t ptset_type, int npnts, int *pnts);
read:
        cg_ptset_info(PointSetType_t *ptset_type, int *npnts);
        cg_ptset_read(int *pnts);
```

The read is made with 2 functions because of the Fortran interface, to return the information about the size of the array "pnts" prior to reading this array. This is how it was done for cg_boco_info/read and cg_conn_info/read. These calls would be used in conjunction with cg_goto.

Notes:

1. If GridLocation is NOT present under BCDataSet_t, then PointList and PointRange (if present) are ignored, and location for data specification defaults to the GridLocation given under the parent BC_t and uses ListLength passed down from there (this is the current method).

2. If GridLocation IS present, then there should also be either a PointList or PointRange. (If there is not, then GridLocation is ignored and location for data specification defaults to the GridLocation given under the parent BC_t.) The PointList or PointRange define a NEW ListLength that overrides the one passed down from the parent. In other words, the data in BCDataSet can now be specified independent of how the patches were defined under BC_t.

3. It is up to the user to insure that there is a correspondence between the list or range defined under BCDataSet_t and that under BC_t (i.e., even though one may be Vertex and one FaceCenter, they still should correspond to the same patch - there is nothing to prevent the user from giving completely dissimilar locations.)