

## Silo - Bug # 533: Memory leak: calling DBGetUcdmesh() for a quad mesh

<b>Status:</b>	Resolved	<b>Priority:</b>	Normal
<b>Author:</b>	Jeffrey Grandy	<b>Category:</b>	
<b>Created:</b>	01/04/2011	<b>Assigned to:</b>	Mark Miller
<b>Updated:</b>	09/12/2012	<b>Due date:</b>	
<b>Likelihood:</b>	2 - Rare		
<b>Severity:</b>	3 - Major Irritation		
<b>Silo Found in Version:</b>	trunk		
<b>OS:</b>	All		
<b>Support Group:</b>	Any		
<b>Subject:</b>	Memory leak: calling DBGetUcdmesh() for a quad mesh		
<b>Description:</b>	<p>DBGetUcdmesh() is allocating memory and returns a null pointer when MeshName is a quad mesh. The application cannot free this memory because a null pointer was returned. The leak appears to be quite large, and may be proportional to the number of nodes in the mesh. See the snippet below.</p> <pre>DBShowErrors(suspend, NULL) ;  DBucdmesh* dbucdm = DBGetUcdmesh(silofile, MeshName ) ;  if ( dbucdm != 0 ) { ... }  DBShowErrors(resume, NULL) ;</pre>		

### History

#### 09/11/2012 10:35 pm - Mark Miller

- Status changed from New to Pending
- Assigned to set to Mark Miller
- Target version set to 4.9
- Estimated time set to 6.00
- Likelihood changed from 3 - Occasional to 2 - Rare

I have resolved this issue. Testing it revealed a comedy of issues with PDB driver in the way logic was written to test object types.

#### 09/12/2012 03:00 pm - Mark Miller

- Status changed from Pending to Resolved

The PJ\_GetObject method in the PDB driver had an argument for caller to pass in to return a string representation of the silo object type. After calling PJ\_GetObject, that string was compared to the correct Silo object type the function is querying and would then fail the read if the object type strings didn't agree. This is a hold over from the days when the PJ routines for Silo were written as though that code was going to live apart from Silo as its own independent library. In any event, I moved the code to check object types into PJ\_GetObject where I can abort allocations if the object types don't agree. That fixes the leaks in these circumstances. I also adjusted multi\_test.c test client to test for a variety of bad reads of this ilk.