



Computational Aircraft Prototype Syntheses

AIM Programming

Utility Functions & Tessellations

For ESP Rev 1.29

Bob Haimes

bob@geocentrictech.com or haimes@mit.edu

Geocentric Technologies LLC

AIM Helper Functions

- provides useful functions for the AIM programmer
- gives access to CAPS Object data
- note that all function names begin with `aim_`
- prototypes in `$ESP_ROOT/include/aimUtil.h`
- if any of these functions are used, then the library must be included (`libaimUtil.a/aimUtil.lib`) in the AIM so/DLL build

Get CAPS revision

```
void aim_capsRev(int *major, int *minor)
```

major the returned major revision

minor the returned minor revision number

Create Dynamic Output Value Object

```
icode = aim_makeDynamicOutput(void *aimInfo, const char *dynObjName,  
                              capsValue *value)
```

aimInfo the AIM context

dynObjName the Name to give the Dynamic Output Object
must be unique for Dynamic Objects in this AIM instance

value the value structure used to create the Dynamic Value Object
the contents are copied into the capsAnalysis structure therefore any pointers in **value**
become “owned” by CAPS

icode integer return code

Notes:

- 1 In *exercises/session09* see `myAIM.c` for a simple programming example
- 2 This function can only be called from within `aimPostAnalysis`
- 3 When a Dynamic Output Object is requested (in pyCAPS) this marks the AIM as *dirty*

CAPS Directory Structure

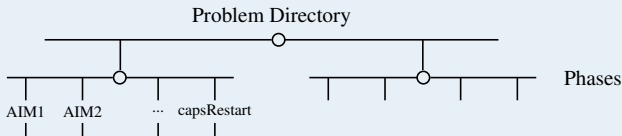
At the top specified directory level you will find *phase* subdirectories

In each *phase* subdirectory you may see:

- capsRestart.cpc – a CSM saved state file – or – capsRestart.egads – an EGADS file (for nonparametric runs)
- capsRestart – subdirectory that contains the CAPS restart data
- capsClosed – the *phase* has been closed (caps_close has been called marking completion)
- capsLock – an indication that another application is executing in this subdirectory
- AIMnames – subdirectories each related to an AIM instance in the running CAPS Problem/Phase

Notes:

- *Scratch* phase (no name specified) is not as protected as the others
- CAPS Problem directory or individual *phase* subdirectories can be copied and used elsewhere



Get Problem root

```
icode = aim_getRootPath(void *aimInfo, const char **fullPath)
```

aimInfo the AIM context

fullPath the file path to find the root of the Problem/Phase directory structure
if on Windows it will contain the drive

icode integer return code

Note: All other uses of *path* is relative to this point.

Get absolute file name in Problem/Phase/Instance directory

```
icode = aim_file(void *aimInfo, const char *rPath, char *aPath)
```

aimInfo the AIM context

rPath a path filename relative to the Problem/Phase/Instance directory structure

aPath returned absolute path filename in the Problem/Phase/Instance directory structure
(length PATH_MAX)

icode integer return code

Relative path file open

```
FILE *fp = aim_fopen(void *aimInfo, const char *rPath,  
                    const char *mode)
```

aimInfo the AIM context

rPath the relative path filename in the Problem/Phase/Instance directory structure

mode specifies the mode used for the file open

fp the returned FILE pointer

Create relative path directory

```
icode = aim_mkdir(void *aimInfo, const char *rPath)
```

aimInfo the AIM context

rPath the relative path directory in the Problem/Phase/Instance directory structure

icode integer return code

Execute a command in the AIMs path

```
icode = aim_system(void *aimInfo, const char *rpath, const char *cmd)
```

aimInfo the AIM context

rpath the relative path from the Analysis' directory or **NULL** (in the Analysis path)

cmd the command to execute in a shell

icode integer return code

Check if relative path file exists

```
icode = aim_isFile(void *aimInfo, const char *rPath)
```

aimInfo the AIM context

rPath the relative path filename in the Problem/Phase/Instance directory structure

icode CAPS_SUCCESS if the file exists, CAPS_NOTFOUND otherwise

Check if relative path directory exists

```
icode = aim_isDir(void *aimInfo, const char *rPath)
```

aimInfo the AIM context

rPath the relative path filename in the Problem/Phase/Instance directory structure

icode CAPS_SUCCESS if the directory exists, CAPS_NOTFOUND otherwise

Copy a file

```
icode = aim_cpFile(void *aimInfo, const char *src, const char *dst)
```

aimInfo the AIM context

src the absolute path filename to copy

dst the path/filename (may be “”) in the Problem/Phase/AIM specific directory structure where the file is copied to

icode integer return code

Make a relative symbolic Link

```
icode = aim_symlink(void *aimInfo, const char *src, const char *dst)
```

aimInfo the AIM context

src the absolute path filename to link to

dst the path/filename (may be “” or **NULL**) in the Problem/Phase/AIM specific directory structure where the relative link is made

icode integer return code

Notes:

- 1 On Windows this simply calls `aim_cpFile`
- 2 **dst** must not exist

Relative path within a Problem/Phase/Instance directory

```
icode = aim_relPath(void *aimInfo, const char *src, const char *dst,  
                   char *relPath)
```

aimInfo the AIM context

src a filename in a Problem/Phase/AIM specific directory

dst the path/filename (may be “”) in the Problem/Phase/AIM specific directory structure

relPath the relative path from **dst** to **src**

icode integer return code

Is this Analysis Directory a Link?

```
icode = aim_fileLink(void *aimInfo, char *srcPath)
```

aimInfo the AIM context

srcPath the returned full filename of the source directory (length PATH_MAX)
can be **NULL** if this information is not required

icode integer return code

CAPS_NOTFOUND indicates the analysis directory is not a CAPS link

Remove a relative path directory

```
icode = aim_rmDir(void *aimInfo, const char *rPath)
```

aimInfo the AIM context

rPath the relative path directory in the Problem/Phase/AIM directory structure

icode integer return code

Note: Wildcards * and/or ? may be used in **rPath**

Remove a relative path file

```
icode = aim_rmFile(void *aimInfo, const char *rPath)
```

aimInfo the AIM context

rPath the relative path file in the Problem/Phase/AIM directory structure

icode integer return code

Note: Returns success even if the **rPath** file does not exist

Name to Index lookup

```
icode = aim_getIndex(void *aimInfo, const char *name,  
                    enum capssType stype)
```

aimInfo the AIM context

name the pointer to the string specifying the name to look-up
NULL returns the total number of members in the subtype

stype GEOMETRYIN, GEOMETRYOUT, ANALYSISIN, ANALYSISOUT or
ANALYSISDYN0

icode index (1 bias) or negative integer return code

Index to Name lookup

```
icode = aim_getName(void *aimInfo, int index, enum capssType stype,  
                  const char **name)
```

aimInfo the AIM context

index the index to use (1 bias)

stype GEOMETRYIN, GEOMETRYOUT, ANALYSISIN, ANALYSISOUT or
ANALYSISDYN0

name the returned pointer to the string specifying the name

icode integer return code

Get GeometryIn Type

```
icode = aim_getGeomInType(void *aimInfo, int index)
```

aimInfo the AIM context
index the index of GEOMETRYIN (1 bias)
icode integer return code – 0 is Design, 1 is Configuration, 2 is Constant

Get Discretization State

```
icode = aim_getDiscrState(void *aimInfo, const char *bname)
```

aimInfo the AIM context
bname the Bound name
icode integer return code – CAPS_SUCCESS is clean

Get Value Structure

```
icode = aim_getValue(void *aimInfo, int index, enum capsType type, capsValue **value)
```

aimInfo the AIM context
index the index to use (1 bias)
type GEOMETRYIN, GEOMETRYOUT, ANALYSISIN, ANALYSISOUT or ANALYSISDYN
value the returned pointer to the capsValue structure

Initialize Value Structure

```
icode = aim_initValue(capsValue *value)
```

value a pointer to the capsValue structure
sets the initial state of the structure as if aimOutput has been invoked

icode integer return code

Free Value Structure

```
icode = aim_freeValue(capsValue *value)
```

value a pointer to the capsValue structure
frees pointers in value and calls aim_initValue to rest

icode integer return code

Get AnalysisIn State WRT the Analysis

```
icode = aim_newAnalysisIn(void *aimInfo, int index)
```

aimInfo the AIM context

index the index to use (1 bias)

icode integer return code – CAPS_SUCCESS is new (i.e., dirty), CAPS_CLEAN if not

Get Geometry State WRT the Analysis

```
icode = aim_newGeometry(void *aimInfo)
```

aimInfo the AIM context

icode CAPS_SUCCESS for new, CAPS_CLEAN if not regenerated since last here

Get the number of instances in the Analysis

```
icode = aim_numInstance(void *aimInfo)
```

aimInfo the AIM context

icode Error code (negative) or the number of instances

Get the instance index for the Analysis

```
icode = aim_getInstance(void *aimInfo)
```

aimInfo the AIM context

icode Error code (negative) or the Instance index (Bias 0)

Get Bound Names

```
icode = aim_getBounds(void *aimInfo, int *nBname, char ***bnames)
```

aimInfo the AIM context

nBname returned number of Bound names

bnames returned pointer to list of Bound names (freeable)

icode integer return code

Clear AIM's directory

```
icode = aim_clear(void *aimInfo)
```

aimInfo the AIM context

icode integer return code

Get Value Attributes

```
icode = aim_valueAttrs(void *aimInfo, int index, enum capsType type,  
                      int *nValue, char ***names, capsValue **values)
```

aimInfo the AIM context

index the index to use (1 bias)

stype GEOMETRYIN, GEOMETRYOUT, ANALYSISIN, ANALYSISOUT or
ANALYSISDYNO

nValue returned number of attributes

names the returned names – nValue in length (freeable)

values the returned pointer to the capsValue structures – nValue in length (freeable)

icode integer return code

Note: use `aim_freeAttrs` to free up the memory.

Get Analysis (our) Attributes

```
icode = aim_analysisAttrs(void *aimInfo, int *nValue, char ***names,  
                           capsValue **values)
```

aimInfo the AIM context

nValue returned number of attributes

names the returned names – nValue in length (freeable)

values the returned pointer to the capsValue structures – nValue in length (freeable)

icode integer return code

Note: use `aim_freeAttrs` to free up the memory.

Free Attribute storage

```
void aim_freeAttrs(int nValue, char **names, capsValue *values)
```

aimInfo the AIM context

nValue the number of attributes

names the names to be freed – nValue in length

values the pointer to the capsValue structures – nValue in length

Get Bodies

```
icode = aim_getBodies(void *aimInfo, const char **intent, int *nBody,  
                     ego **bodies)
```

aimInfo the AIM context

intent the returned pointer to the capsIntent string used to filter the Bodies

nBody the returned number of EGADS Body Objects that match the intent

bodies the returned pointer to a list of EGADS Body/Node Objects

icode integer return code

Is Node Body

```
icode = aim_isNodeBody(ego body, double *xyz)
```

body the EGADS Body Objects to query

xyz the returned XYZ of the Node (if a Node Body)

icode integer return code

Register a New Tessellation

```
icode = aim_newTess(void *aimInfo, ego tess)
```

aimInfo the AIM context

tess the EGADS Tessellation Object to register

icode integer return code

Notes:

- 1 If the Body associated with **tess** already has a registered Tessellation Object, the previous tessellation **ego** will be deleted
- 2 Any Tessellation Object registered will be deleted by CAPS before a Geometry regeneration
- 3 If the Body associated with **tess** is not on the `OpenCSM` stack, the Body Object will be deleted when the Tessellation Object is cleaned up (i.e., CAPS takes ownership of the Body from the AIM)

- In *examples* modify `session09.py` to:
 - Filter (*capsIntent*) myAIM on “CFD”
 - A second instance of myAIM that filters on “STRUCTURES”
- Analysis Dynamic Outputs:
 - Move around the query of “Everything” – ANALYSISDYNODoes the execution sequence make sense (set the *define* DEBUG)
 - Examine `dynoutOnly.py` – note that requesting Dynamic Outputs alone triggers execution
- Write a file in “CFD” that can we read in “STRUCTURES”
 - Does “CFD” get executed first?
 - How do you get the proper path AIM instance to AIM instance?
- Do the same with a *linkage*
 - Clue → you need to hook an output from “CFD” to an input into “STRUCTURES”