

## Mili Python Interface

### Table of Contents

<b>Developer's Guide .....</b>	<b>3</b>
<b>Description .....</b>	<b>3</b>
<b>Mili Class .....</b>	<b>3</b>
Description .....	3
Reader Functions.....	5
Query Functions .....	7
Other Functions.....	8
<b>Other Objects .....</b>	<b>10</b>
StateMap .....	10
Directory.....	10
StateVariable .....	11
Subrecord .....	12
SubrecordContainer .....	12
MeshObjectClassData .....	12
BlockList.....	13
Item .....	13
StateAnswer .....	14
Answer .....	15
<b>Miscellaneous.....</b>	<b>16</b>
Directory Type .....	16
Superclass.....	16
ConnWords.....	16
DataType .....	16
ExtSize .....	16
Aggregate Type .....	16
Data Organization.....	17
<b>Mili Python Interface Tests .....</b>	<b>18</b>
Description .....	18
test_element_number_material .....	18
test_nodes_material .....	18
test_nodes_label .....	18
test_state_variable.....	18
test_node_attributes .....	18
test_state_variable_vector .....	18
test_modify_state_variable .....	18
test_modify_vector.....	18
test_state_variable_vector_array.....	18
test_state_variable_vector_array_component.....	18
test_modify_vector_array.....	18
test_modify_vector_array_component .....	18
<b>User's Guide .....</b>	<b>19</b>

<b>Description .....</b>	<b>19</b>
<b>Mili Class .....</b>	<b>19</b>
Description .....	19
Reader Functions.....	19
Query Functions .....	19

## Developer's Guide

### Description

This program can be imported and initialized using the following:

```
import read  
mili = read.Mili()  
mili.read(file_name)
```

From this point, the mili object contains all the information from the mili file and can be queried.

```
answer = mili.nodes_of_elem(1, "brick")  
answer2 = mili.query(['nodpos[ux]'], None, [4], 'node', [3])
```

### Mili Class

#### Description

The Mili Class has variables that store the pertinent information for a mili file. These variables are filled during the various functions that read in the Mili file and will then be used when querying. The following are variables that a user might want to access if writing a script to work alongside the reader:

state\_maps

Type: Array

Description: an array of StateMap objects

directories

Type: Array

Description: an array of Directory objects

params

Type: Dictionary

Description: maps parameter name to a list containing the value, which could be a single integer, a string, or an array

state\_variables

Type: Dictionary

Description: maps from:

name: [StateVariable, [subrecords containing this name]]

mesh\_object\_class\_datas

Type: Dictionary

Description: maps from shortname: MeshObjectClassData

labels

Type: Dictionary

Description: maps from label: element id

(superclass, class) : {label: element id}

int\_points

Type: Dictionary

Description: maps from:

stress/strain : {es\_x : [(integration points), total integration points]}

and from:

es\_x : [(integration points), total integration points]

nodes

Type: Array

Description: an array of starting node position coordinates (each entry in the list is a nodes position) with node number i-1 in the ith slot in the array, since the node array in the definition is 1-indexed, and python is 0-indexed

materials

Type: Dictionary

Description: maps from:

material number : [[id, class\_name]]

Each class can have its own element with id 2, so this organization is necessary

matname

Type: Dictionary

Description: maps from material name: material number

connectivity

Type: Dictionary

Description: maps from:

class\_name : {id : [nodes]}

The nodes make up element id in the class

dim

Type: Integer

Description: the dimension of the simulation

## Reader Functions

### *readStateMaps*

*input:*

self: the mili object

f: the file object containing the state maps

*output:*

offset: the offset in the file after reading the state maps

*description:*

Reads the state maps one at a time, creating a new `StateMap` object for each and adding to the mili's `state_map` array

### *readStateVariablesAndParams*

*input:*

self: the mili object

f: the file object containing the state maps

offset: the offset to begin reading at

*output:*

None

*description:*

Iterates over the directories matching on directories with a `DirectoryType` of state variables or parameters. Given a parameter match, add the correct data to `self.params`. Given a state variable match, create a `StateVariable` and add the state variable to `self.state_variables`. When added, an empty array is included, which will later be filled with subrecords containing the state variable.

### *readDirectories*

*input:*

self: the mili object

f: the file object containing the state maps

offset: the offset to begin reading the directories

*output:*

offset: the offset in the file after reading the directories

*description:*

Reads the directories one at a time, creating a new `Directory` for each and adding to the mili's `self.directories` array.

*readMesh*

*input:*

self: the mili object

f: the file object containing the state maps

*output:*

None

*description:*

Iterates through the directories matching on directories relating to the mesh, such as node and label definitions. Upon a match, updates the appropriate object in the mili object.

*readSubrecords*

*input:*

self: the mili object

f: the file object containing the state maps

*output:*

None

*description:*

Iterates through the directories matching on directories with state record data. Given a match, creates a `Subrecord` and adds the subrecord to `self.srec_container`, which contains the subrecords.

*read*

*input:*

self: the mili object

file\_name: the name of the Mili file (the .pltA file)

state\_map\_file\_name: the name of the state map file (the .pltA00 file)

*output:*

None

*description:*

Calls the other reader functions to build up a Mili object.

## Query Functions

### *query*

#### *input:*

self: the mili object

names: the names of the state variables

material: the material in the result

if this value is nonzero – looks for all labels matching this material

and class\_name

labels: the labels in the result

class\_name: the class\_name of the result

state\_numbers: the state\_numbers in the result

modify: whether or not there were modifications

int\_points: the integration points

raw\_data: whether or not to simply output raw data, not Answer

#### *output:*

answer: an Answer containing the information from the query

#### *description:*

Searches for the given state variables at specified states, labels, etc.

### *elements\_of\_material*

#### *input:*

self: the mili object

material: the name of the material

raw\_data: whether or not to simply output raw data, not Answer

#### *output:*

answer: an Answer containing the elements of the specified material

#### *description:*

Given a material name, finds all the elements that are of this material.

Note: If you know the material number, you can use self.matname, a dictionary from material name to number, to find the name.

### *nodes\_of\_material*

#### *input:*

self: the mili object

material: the name of the material

class\_name: the class name

raw\_data: whether or not to simply output raw data, not Answer

#### *output:*

answer: an Answer containing the nodes of the specified material and class

#### *description:*

Given a material name and class name, finds all the nodes.

Note: If you know the material number, you can use `self.matname`, a dictionary from material name to number, to find the name.

### *nodes\_of\_elem*

#### *input:*

`self`: the mili object  
`label`: the label of the element  
`class_name`: the class name  
`raw_data`: whether or not to simply output raw data, not `Answer`

#### *output:*

`answer`: an `Answer` containing the nodes of the specified material and class

#### *description:*

Given a label and class name, finds all the nodes.

## Other Functions

### *set\_string*

#### *input:*

`self`: the mili object  
`subrecord`: the subrecord object

#### *output:*

`ret_final`: a string representation of the state variables on the subrecord

#### *description:*

Iterates through the state variables on the subrecord, appending to a return string along the way, given the type of the state variable. At the end of the method, `ret_final` contains a single string for every variable in the subrecord that can be used in a call using `struct`.

### *parse\_name*

#### *input:*

`self`: the mili object  
`name`: the name of the Mili file (the `.pltA` file)

#### *output:*

returns `[vector, component]`  
`vector`: the name of the vector, if it is a vector  
`component`: the name of the component, if there is a vector, or the name of the original state variable if there is no vector.

#### *description:*

Checks to see if the name contains an open bracket, indicating this is a vector with a component given. If so, breaks the name down into these two parts. Otherwise, returns the original name in the component slot.



*create\_answer*

*input:*

self: the mili object  
res: the result  
names: the names of the state variables  
materials: the materials in the result  
labels: the labels in the result  
class\_name: the class\_name of the result  
state\_numbers: the state\_numbers in the result  
modify: whether or not there were modifications  
raw\_data: whether or not to simply output raw data, not Answer

*output:*

if raw\_data:  
    res: contains the results in the following structure:  
        res[state\_number][name][label]  
else:  
    answer: an Answer containing the information from res

*description:*

For every state, creates a StateAnswer, which contains all the information in res[state\_number]. Then loops over the names and labels, gathering the information out of res[state\_number][name][label] and adding the created Item from each of these the state answer.

*is\_vec\_array*

*input:*

self: the mili object  
name: the names of the state variable  
class\_name: the class\_name of the state variable

*output:*

element set name or None

*description:*

Checks to see if name is an element set and returns the correct element set name given the class\_name.

*variable\_at\_state*

*input:*

self: the mili object  
subrecord: the subrecord to search  
labels: the labels of the state variable  
name: the name of the state variable  
vars: the array of state variable values on the subrecord  
sup\_class: the superclass of the state variable

clas: the class name of the state variable  
sub: the number of the subrecord  
res: the result  
modify: whether or not this is part of a modification  
int\_points: the integration points

*output:*

if `modify` is `True`, returns `[res, indices]` where `res` is a dictionary structure containing the result and `indices` are the indices in the subrecord's array of variables to modify.  
if `modify` is `False`, returns the values obtained by searching the subrecord for the specified state variables, labels, etc.

*description:*

Searches the subrecord for the given input.

## Other Objects

### StateMap

*Description*

Contains information at given time instance for *every* subrecord. The state records themselves are stored in a separate binary file with a `.plt00` ending.

*\_\_init\_\_*

*input:*

`self`: the mili object  
`file_number`: the file number of this `StateMap`  
`file_offset`: the offset for the state of the `StateMap`  
`time`: the time in the simulation for this `StateMap`  
`state_map_id`: the number of the `StateMap`

*output:*

`None`

*description:*

Initializes the variables of the `StateMap`

### Directory

*Description*

An organization tool for Mili files. These files dictate the layout of the `.pltA` file.

*\_\_init\_\_*

*input:*

self: the Directory object  
type\_idx: the type of the Directory  
modifier\_idx1: type specific information  
modifier\_idx2: type specific information  
string\_qty\_idx: number of strings associated with this Directory  
offset\_idx: offset to being reading Directory  
length\_idx: the length of the Directory

*output:*

None

*description:*

Initializes the variables of the Directory

StateVariable

*Description*

Objects that are updated at different time instances.

list\_size: the number of variables if this is a vector

order: the number of dims

dims: the rank of the state variables in svars

svars: the list of state variables for vectors and vector arrays

*\_\_init\_\_*

*input:*

self: the StateVariable object  
name: the name of the StateVariable  
title: the title of the StateVariable  
agg\_type: the aggregate type of the StateVariable  
data\_type: the data type of the StateVariable

*output:*

None

*description:*

Initializes the variables of the StateVariable

*atom\_qty*

*input:*

self: the StateVariable object  
state\_variables: the mili.state\_variable object that can be used to map from name to the StateVariable

*output:*

qty: the quantity of variables in the StateVariable

*description:*

Loops over the svars and uses the order and dims to find the total number of state variables (“atoms”) in the StateVariable

## Subrecord

*Description*

A group of StateVariables.

qty\_blocks: the number of sections of elements (e.g. 3)

mo\_blocks: the sections of elements (e.g. 1-5, 7-9, 11-12)

mo\_qty: the total number of elements (e.g. 10)

offset: the offset of the Subrecord

size: the size of the Subrecord

\_\_init\_\_

*input:*

self: the Subrecord object

name: the name of the Subrecord

class\_name: the class name

organization: either result or object ordered

qty\_svars: number of state variables in the Subrecord

svar\_names: names of the state variables Subrecord

*output:*

None

*description:*

Initializes the variables of the Subrecord

## SubrecordContainer

*Description*

A group of Subrecords.

subrecs: the array of subrecords

size: the combined size of the subrecords

## MeshObjectClassData

*Description*

The mesh data needed for the mili file.

*\_\_init\_\_*

*input:*

self: the MeshObjectClassData object  
short\_name: the shorter name of the MeshObjectClassData  
long\_name: the longer name of the MeshObjectClassData  
superclass: the superclass of the class  
blocklist: the BlockList for this MeshObjectClassData

*output:*

None

*description:*

Initializes the variables of the MeshObjectClassData

*add\_block*

*input:*

self: the MeshObjectClassData object  
start: the start element  
stop: the stop element

*output:*

None

*description:*

Adds the start, stop section to the BlockList

BlockList

*Description*

Contains the sections of elements for a MeshObjectClassData.

*\_\_init\_\_*

*input:*

self: the BlockList object  
obj\_qty: the number of elements in blocks  
block\_qty: the length of blocks  
blocks: a list of tuples, where each tuple is a start, stop

*output:*

None

*description:*

Initializes the variables of the BlockList

Item

*Description*

A single piece of an Answer. Any number of its variables may end up uninitialized. Often the Item represents a StateVariable.

`always_print`: whether or not this Item is always printed in an Answer

`__init__`

*input:*

`self`: the Item object  
`name`: the name of the Item  
`material`: the material of the Item  
`mo_id`: the element id of the Item  
`label`: the label of the Item  
`class_name`: the class name  
`modify`: whether or not the Item was modified  
`value`: the value of the Item

*output:*

None

*description:*

Initializes the variables of the Item

`set`

*input:*

`self`: the Item object  
`value`: the value of the Item

*output:*

None

*description:*

Sets the value and also sets `always_print`

`__str__`

*input:*

`self`: the Item object

*output:*

`ret`: the string representation of the Item

*description:*

Displays the Item in a readable format for outputting the answer to the screen or to a file.

StateAnswer

*Description*

The representation of all the data for a state, used in a state based query.

`items`: a list of Item objects

`state_number`: the state number of the StateAnswer

*\_\_init\_\_*

*input:*

self: the StateAnswer object

*output:*

None

*description:*

Initializes the variables of the StateAnswer

*\_\_str\_\_*

*input:*

self: the StateAnswer object

*output:*

ret: the string representation of the StateAnswer

*description:*

Displays the StateAnswer in a readable format for outputting the answer to the screen or to a file.

Answer

*Description*

The return value of a query that contains all information requested.

state\_answers: list of StateAnswer objects

*\_\_init\_\_*

*input:*

self: the Answer object

*output:*

None

*description:*

Initializes the variables of the Answer

*set*

*input:*

self: the Answer object

names: the names of the Answer

materials: the material sof the Answer

mo\_ids: the element ids of the Answer

labels: the labels of the Answer

class\_name: the class name

modify: whether or not the Answer was modified

*output:*

None

*description:*

If this `Answer` is not state based, this function creates a list of `Item` objects and sets the `self.items`

*\_\_str\_\_*

*input:*

self: the `Answer` object

*output:*

ret: the string representation of the `Answer`

*description:*

Displays the `Answer` in a readable format for outputting the `Answer` to the screen or to a file.

## Miscellaneous

### Directory Type

*Description*

A mapping of the string directory types to the integer representation

### Superclass

*Description*

A mapping of the string superclass types to the integer representation

### ConnWords

*Description*

A mapping of the string connection types to the integer representation

### DataType

*Description*

A mapping of the string data types to the integer representation

### ExtSize

*Description*

A mapping of the string data types to the size in bytes

### Aggregate Type

*Description*

A mapping of the string aggregate types to the integer representation



## Data Organization

### *Description*

A mapping of the string data organization types to the integer representation

## Mili Python Interface Tests

### Description

These tests all use the d3samp6.dyn file as the basis for the tests. They cover the basic functionality.

#### test\_element\_number\_material

Testing what element numbers associated with a material

#### test\_nodes\_material

Testing Testing what nodes are associated with a material

#### test\_nodes\_label

Testing what nodes are associated with a label

#### test\_state\_variable

Testing accessing a variable at a given state

#### test\_node\_attributes

Testing accessing accessing node attributes -> this is a vector component  
Tests both ways of accessing vector components (using brackets vs not)  
e.g. nodpos[ux] and ux

#### test\_state\_variable\_vector

Testing the accessing of a vector, in this case node position

#### test\_modify\_state\_variable

Testing the modification of a scalar state variable

#### test\_modify\_vector

Testing the modification of a vector state variable

#### test\_state\_variable\_vector\_array

Testing accessing a vector array

#### test\_state\_variable\_vector\_array\_component

Testing accessing a vector array component

#### test\_modify\_vector\_array

Testing modifying a vector array

#### test\_modify\_vector\_array\_component

Test modifying a vector array component

## User's Guide

### Description

This program can be imported and initialized using the following:

```
import read  
  
mili = read.Mili()  
  
mili.read(file_name)
```

From this point, the mili object contains all the information from the mili file and can be queried.

```
answer = mili.nodes_of_elem(1, "brick")  
  
answer2 = mili.query(['nodpos[ux]'], None, [4], 'node', [3])
```

### Mili Class

#### Description

The Mili Class has variables that store the pertinent information for a mili file. These variables are filled during the various functions that read in the Mili file and will then be used when querying.

#### Reader Functions

##### *read*

##### *input:*

self: the mili object

file\_name: the name of the Mili file (the .pltA file)

state\_map\_file\_name: the name of the state map file (the .pltA00 file)

##### *output:*

None

##### *description:*

Calls the other reader functions to build up a Mili object.

#### Query Functions

##### *query*

##### *input:*

self: the mili object

names: the names of the state variables  
class\_name: the class\_name of the result  
material: the material in the result  
    if this value is nonzero – looks for all labels matching this material  
    and class\_name  
labels: the labels in the result  
    if this value is zero – gets all labels pertaining to the class  
state\_numbers: the state\_numbers in the result  
modify: whether or not there were modifications  
int\_points: the integration points  
raw\_data: whether or not to simply output raw data, not Answer

*output:*

answer: an Answer containing the information from the query

*description:*

Searches for the given state variables at specified states, labels, etc.

*elements\_of\_material*

*input:*

self: the mili object  
material: the name of the material  
raw\_data: whether or not to simply output raw data, not Answer

*output:*

answer: an Answer containing the elements of the specified material

*description:*

Given a material name, finds all the elements that are of this material.  
Note: If you know the material number, you can use self.matname, a dictionary from material name to number, to find the name.

*nodes\_of\_material*

*input:*

self: the mili object  
material: the name of the material  
class\_name: the class name  
raw\_data: whether or not to simply output raw data, not Answer

*output:*

answer: an Answer containing the nodes of the specified material and class

*description:*

Given a material name and class name, finds all the nodes.  
Note: If you know the material number, you can use self.matname, a dictionary from material name to number, to find the name.

*nodes\_of\_elem*

*input:*

self: the mili object

label: the label of the element

class\_name: the class name

raw\_data: whether or not to simply output raw data, not Answer

*output:*

answer: an Answer containing the nodes of the specified material and class

*description:*

Given a label and class name, finds all the nodes.