|  |  |
| --- | --- |
| **gemini_logo** | **Implementation of the External Task Interface for the Recipe System** |
| **Craig Allen, Kaniela Dement** |
| Data Processing Software Group |
|  |
| V1.0 – July 25 2012 |

**Revision History**

V1.0 – July 25 2012 Kaniela Dement

**Document Purpose**

To present an implementation of an object-oriented design for a new Recipe System component called the External Task Interface (ETI). The ETI will manage many objects to set-up, execute, recover and clean external tasks called by primitives.

**Intended Audience**

The intended audience are the project managers and developers working on the Pipeline Project.

**Table of Contents**

1. Introduction 2

2. History 2

3. Implementation 2

3.1 Base Classes 2

3.2 Sub Classes 2

3.3 stackFrames Primitive source for ETI PyRAF task gemcombine: 3

4. Technical Details 3

4.1 eti.py 3

4.1.1 class ExternalTaskInterface 4

4.2 etifile.py 4

4.2.1 class ETIFile 4

4.3 etiparam.py 5

4.3.1 class ETIParam 5

4.4 pyrafeti.py 5

class PyrafETI 5

4.5 pyrafetfile.py 5

4.5.1 class PyrafETIFile 5

4.6 pyrafetiparam.py 6

4.6.1 class PyrafETIParam 6

4.6.2 class IrafStdout 6

4.7 gemcombineeti.py 6

4.7.1 class GemcombineETI 6

4.8 gemcombinefile.py 7

4.8.1 class GemcombineFile 7

4.8.2 class InAtList 7

4.8.3 class OutFile (see code) 8

4.8.4 class LogFile (see code) 8

4.9 gemcombineparam.py 8

4.9.1 class GemcombineParam 8

4.9.2 class FlVardq (see code) 8

4.9.3 class FlDqprop (see code) 8

4.9.4 class Masktype (see code) 8

4.9.5 class Combine (see code) 8

4.9.6 class Nlow (see code) 8

4.9.7 class Nhigh (see code) 8

4.9.8 class Reject (see code) 8

4.9.9 hardcoded params 8

5. Conclusion 9

# Introduction

The External Task Interface (ETI) is a way for the Recipe System to use external tools. The ETI adheres to an Object-oriented design in order to adapt to any external task. The first implementation of the ETI was done for the PyRAF task, gemcombine.The primitive that calls gemcombine is stackFrames, which is part of a mini-recipe composed by the primitive alignAndStack. The ETI gemcombine has been tested by writing a special stackFramesETI primitive that takes in the same parameters and files as stackFrames, but uses the ETI.

# History

The CLManager class was added to the Recipe System to deal with calling external PyRAF tasks. Over time it has become very bloated with unused methods, duplicate blocks of code, etc. It basically couldn’t grow with the system without getting out of control.This class has also been identified to be the cause of memory leaks. The ETI was designed to replace the CLManager with a more OO design in hopes of remedying these issues.

# Implementation

Here we present the ETI design as it has been implemented in the stackFrames primitive. The external task that the Recipe System interfaces with is the PyRAF task, gemcombine. This task is part the Gemini IRAF Package. PyRAF itself is an interface to IRAF.

## Base Classes

Three base classes house the highest level of methods and members used by the ETI. They are located in three different modules in astrodata/eti:

1. ExternalTaskInterface (eti.py)
2. ETIFile (etifile.py)
3. ETIParam (etiparam.py)

The ExternalTaskInterfaceobject delegates work by calling the methods of ETIFile and ETIParam objects to prepare, recover, and clean the external task. The execution is done through the ExternalTaskInterface sub-class execute().

## Sub Classes

Sub-classes make it possible to inherit or override methods from parent classes. This allows code to be reused and defined to certain levels. For PyRAF external tasks, a PyRAFlevel of code is created by these three sub-classes located in astrodata/eti:

1. PyrafETI (a child of ExternalTaskInterface in pyrafeti.py)
2. PyrafETIFile (a child of ETIFile in pyrafeti.py)
3. PyrafETIParam (a child of ETIParam in pyrafeti.py)

These sub-classescontain PyRAF specific methodsAn example is get\_parameter()of PyafETIFile. It returns a member named filedict, which contains the dictionary fragment necessary to prepare aparameter dictionary for execution. All PyRAF tasks require this method.

If there are different tasks for the same external tool then another layer of sub-classes may be created. This is the case for the PyRAF task gemcombine. This task requires some special methods, files, and parameters that are not used for other PyRAF ETI tasks. As other tasks are converted to use the ETI then certain items can be identified to move back up to the PyRAF level. These are the sub-classes for the gemcombine level in gempy/eti:

1. GemcombineETI (a child of PyrafETI in gemcombineeti.py)
2. GemcombineFile (a child of PyrafETIFile in gemcombinefile.py)
3. GemcombineParam (a child of PyrafETIParam in gemcombineparam.py)

Gemcombine adds even more sub-class levels necessary for specific types of files and parameters. Please note that not all parameters have their own special class. The ones here are subject to change by the user through the primitive parameter system.

1. InAtList (a child of GemcombineFile in gemcombinefile.py)
2. OutFile (a child of GemcombineFile in gemcombinefile.py)
3. LogFile (a child of GemcombineFile in gemcombinefile.py)
4. FlVardq (a child of GemcombineParam in gemcombineparam.py)
5. FlDqprop (a child of GemcombineParam in gemcombineparam.py)
6. Masktype (a child of GemcombineParam in gemcombineparam.py)
7. Combine (a child of GemcombineParam in gemcombineparam.py)
8. Nlow (a child of GemcombineParam in gemcombineparam.py)
9. Nhight (a child of GemcombineParam in gemcombineparam.py)
10. Reject (a child of GemcombineParam in gemcombineparam.py)

## stackFrames Primitive source for ETI PyRAF task gemcombine:

from gempy.eti.gemcombineeti import GemcombineETI

def stackFramesETI(self, rc):

…

gemcombine\_task = GemcombineETI(rc)

adout = gemcombine\_task.run()

…

yield rc

Using the ETI for this primitive has cut down the number of linescode from 197 to 110. Also the CLManger has been completely bypassed. The run() function runs the prepare, execute, recover and clean methods.

# Technical Details

Here is a breakdown (parameters, members, methods, and classes)of modules required for the Gemcombine External Task Interface for the primitive stackFrames . The eti and pyraf eti modules are located in astrodata/eti, however because gemcombine is specific for the gemini iraf package it has been decided to keep these eti modules in gempy/eti. Therefore modules in gempy will inherit modules from astrodata and gemini specific modules will be separated.

## eti.py

Module that contains the ETI base class located in astrodata/eti directory.

### class ExternalTaskInterface

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| param\_objs | list | Container for ETIParam object(s) |
| file\_objs | List | Container for ETIFile object(s) |
| rc | ReductionContext | Reduction metadata object |

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_(rc) | assign members |
| add\_file(file) | |  |  |  | | --- | --- | --- | | Parameter | Type | Description | | File | ETIFile | File Object |   Add file object to file\_objs list |
| add\_param(param) | |  |  |  | | --- | --- | --- | | Parameter | Type | Description | | param | ETIParam | Parameter Object |   Add parameter object to param\_objs list |
| clean() | Calls file and param object clean |
| execute() | Placeholder function(override exists in sub-class) |
| prepare() | Calls file and param object prepare |
| recover() | Calls file and param object recover |
| run() | Calls prepare execute recover clean respectively |

## etifile.py

Module that contains the ETI base class for input/output files. Located in astrodata/eti.

### class ETIFile

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| name | String | Name of file |
| rc | ReductionContext | Reduction metadata object |

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_(name, rc) | assign parameters as members |
| prepare() | pass |
| recover() | pass |
| clean() | pass |

## etiparam.py

Module that contains the ETI base class for parameters. Located in astrodata/eti.

### class ETIParam

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| name | String | Name of parameter |
| rc | ReductionContext | Reduction metadata object |

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_(name, rc) | assign parameters as members |
| prepare() | pass |
| recover() | pass |
| clean() | pass |

## pyrafeti.py

Module that contains the sub-class for pyraf external tasks. Located in astrodata/eti.

### class PyrafETI

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| rc | Reduction Context | Reduction metadata object |

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_(rc) | call the base class init (ExternalTaskInterface.\_\_init\_\_()) |
| execute() | Pass |
| recover() | Pass |

## pyrafetfile.py

Module that contains the sub-class for pyraf external task files. Located in astrodata/eti.

### class PyrafETIFile

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| rc | Reduction Context | Reduction metadata object |
| filedict | dictionary | container for the file name and value |

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_(rc) | call the base class init (ETIFile.\_\_init\_\_()) |
| get\_parameter() | return the filedict |
| prepare() | Pass |
| recover() | Pass |
| clean() | Pass |

## pyrafetiparam.py

Module that contains the sub-class for pyraf external task parameters. It also contains a class needed to record the IRAF standard output. Located in astrodata/eti.

### class PyrafETIParam

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| rc | Reduction Context | Reduction metadata object |
| paramdict | dictionary | container for the parameter name and value |

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_(rc) | call the base class init (Param.\_\_init\_\_()) |
| get\_parameter() | return the paramdict |
| prepare() | Pass |
| recover() | Pass |

### class IrafStdout

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_(rc) | Assign the logger as a member |
| flush() | Pass |
| write(out) | Write PYRAF task stdout to logger |

## gemcombineeti.py

Module that contains the sub-class for the pyraf gemcombine task. Located in gempy/eti.

### class GemcombineETI

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| rc | Reduction Context | Reduction metadata object |
| clparam\_dict | dictionary | container for PYRAF parameter name and values |

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_(rc) | call the base class init (PyrafETI.\_\_init\_\_())  call add\_file and add\_param methods (inherited from base class) for every file and parameter object imported including hardcoded params |
| get\_parameter() | return the paramdict |
| execute() | Copy the clparam\_dict to prevent object collision then update the copied dictionary (xcldict) with PYRAF parameters by running throught the file and param object lists and calling their respective get\_parameter methods.  Unlearn the gemcombine task to start with a clean slate, run through the parameters and use the setParam PYRAF method to log the parameters BEFORE making the actual gemcombine call.  Execute gemcombine by passing it the now populated parameter dictionary (xcldict) and then check the status, raise Errors.OutputError if encounter trouble otherwise log a passing message |
| run() | Convenience function to run all the needed operations, prepare, execute, recover, clean, then return the result (in this case it is one AstroData instance) |
| recover() | Call all parameter and file object recover() methods, be sure to load the output file into memory and return the AstroData object |

## gemcombinefile.py

Module that contains the sub-class for pyraf files and deeper sub-classes for specific gemcombine files.

### class GemcombineFile

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| rc | Reduction Context | Reduction metadata object |
| diskinlist | list | input file names |
| taskname | string | gemcombine |
| pid\_str | string | os.getpid() |
| pid\_task | string | the pid plus the taskname |
| adinput | list | list of ad objects from rc |

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_(rc) | call the base class init (PyrafETIFile.\_\_init\_\_())  assign members |
| get\_prefix() | return prefix string for temporary filenames |

### class InAtList

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| rc | Reduction Context | Reduction metadata object |
| atlist | None | input file names |

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_(rc) | call the base class init (GemcombineFile.\_\_init\_\_())  assign members |
| prepare() | Run through inputs and for each one create a temporary name then write a file that contains each of those names, this is the @list |
| clean() | Delete the temporary files using the @list, then delete the @list as well |

### class OutFile (see code)

### class LogFile (see code)

## gemcombineparam.py

Module that contains the sub-class for pyraf parameters and deeper sub-classes for specific gemcombine parameters.

### class GemcombineParam

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| rc | Reduction Context | Reduction metadata object |
| key | string | Parameter name |
| value | string | Parameter value |
| pid\_task | string | The process id and the task name |
| adinput | list | List of ad objects from rc |
|  |  |  |

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_(rc) | call the base class init (PyrafETIParam.\_\_init\_\_()), assign members |
| nonecheck() | Checks for python None and changes it to pyraf "none" |
| prepare() | Update paramdict (inherited from PyrafETIParam class)with key and value |

### class FlVardq (see code)

### class FlDqprop (see code)

### class Masktype (see code)

### class Combine (see code)

### class Nlow (see code)

### class Nhigh (see code)

### class Reject (see code)

### hardcoded params

A dictionary that contains the hardcoded params for gemcombine.

# Conclusion

The ability for the Recipe System to interface to other tools is very important because it allows the flexibility needed to complete complicated data reductions. The External Task Interface gives this ability to the Recipe System with a simple, clear, object oriented design.

This design allows the workload to be evenly distributed, therefore the objects contain a minimal amount of lines of code rather than just functions that tend to grow exponentially. Another advantage is when it comes to making changes. Because the code is encapsulated in objects, developers go to the design to find where to make the necessary changes rather than getting lost in the code. Whether adding a new feature, modifying an existing one, or completely removing it, this design allows the developers to do it in one place. This saves time and effort.