

Roman - Pandeia Engine Output API

Overview



API Warning

The Pandeia Engine API may change at any time. The following information and examples are for Pandeia Engine 2.0.x only, and are not guaranteed to be accurate or functional for previous or future releases. Updates (and information about planned pending API changes) can be found on the [Pandeia Engine News](#) page.

Pandeia Engine outputs are, like the inputs, a hierarchical dictionary. They contain:

- "scalar": 0-dimensional extracted values from the extraction aperture, at the wavelength of interest (if a spectroscopic calculation)
- "1d": 1-dimensional spectral outputs (single-valued arrays, if the calculation is an imaging-type calculation)
- "2d": 2-dimensional detector images
- "3d": 3-dimensional flux cubes of the input scene, created from the scene definition. For IFU modes only, also includes reconstructed 3-dimensional data cubes of the extracted data.
- "input": A complete copy of the [input dictionary](#)
- "information": The configured exposure properties of the calculation
- "warnings": all warnings produced by the Engine
- "web_report": a hierarchical dictionary of formatted results, that forms the report pane of the JWST web ETC.
- "transform": WCS information about the 2D output products.
- "debugarrays": (optional)

In addition, if `pandeia.engine.perform_calculation()` is run with the "as_fits" keyword set to True, the 1d, 2d, and 3d output dictionaries will contain data formatted as FITS HDULists.

Full API

Complete API documentation can be found in [this file](#).

Examples

The scalar outputs are a superset of the values [the Webapp](#) produces in the Results tab.

The following code block will (given an result dictionary named 'result') print out results.

Example python code to pretty-print the scalar results and warnings

```
# print the scalar products with the correct units.
print("-----\n      RESULTS      \n-----")
for x in sorted(result["scalar"]):
    if isinstance(result["scalar"][x],float):
        basename = "{:20}: {:>10.3f}"
    else:
        basename = "{:20}: {}"

    if "time" in x:
        basename += " sec"
    elif "size" in x or "offset" in x:
        basename += " arcsec"
    elif "area" in x: # checking this before background means the
                     # background_area will be given the correct units.
        basename += " pixel^2"
    elif "wavelength" in x:
        basename += " microns"
    elif (((("extracted" in x) or ("sky" in x)) or ("total" in x) or ("brightest" in x)) and ("integrations")
not in x):
        basename += " e-/sec"
    elif "background" in x:
        basename += " MJy/sr"
    elif "cr_ramp_rate" in x:
        basename += " events/integration/pixel"
    else:
        pass
    print(basename.format(x,result["scalar"][x]))

if len(result['warnings']) > 0:
    print("-----\n      WARNINGS      \n-----")
    for x in result['warnings']:
        print("{:20}: {}".format(x,result['warnings'][x]))
print("-----")
```

For formatted output identical to the JWST Webapp, use this code:

Example Python code (new)

```
from textwrap import wrap

report = result['web_report']
print("-"*30 + " RESULTS " + "-"*30)
for category in report:

    print("\033[1m" + category["category"] + "\033[0m")
    print('-'*len(category["category"]))
    for item in category["items"]:
        if "value" in item:
            namelist = wrap(item['name'], width=36, subsequent_indent="    ")
            if len(namelist) > 1:
                for i in range(len(namelist)-1):
                    print(f"{namelist[i]:<36}")
            if "indent" in item and item["indent"]:
                namelist[-1] = "    " + namelist[-1]
            print(f"{namelist[-1]:<36} {item['value']:>16} {item['unit']:<10}")
        else:
            print(f"{item['name']}")
    print()
print("-"*70)

if len(result['warnings']) > 0:
    print("-----\n WARNINGS \n-----")
    for x in result['warnings']:
        print("{:20}: {}".format(x,result['warnings'][x]))
    print("-----")
```