# CTAM Internal Testing Feature

**Motivation:**

1. Encourage users to adopt CTAM for not just compliance testing but also for internal validation.
2. Permits IP sensitive test content to be accidentally pushed to Github.
3. Permits test content to be well defined and validated before exposure to external users.

**Goals:**

1. Seamless move of internal test cases & corresponding modules to external test case & interfaces.
   1. Internal test case or linked test group will not have any import of internal interface class objects directly – everything goes through external interface classes.
2. Internal interface classes should be allowed to use modules of external interface class.
3. The footprint of supporting internal test cases should be very minimal on the externally visible classes/files.

A diagram of a test

Description automatically generated

**Solution:** - Use meta class

**Refer the next pages on how to enable this.**

1. **Internal tests/interfaces folder structure:**

A screenshot of a computer screen

Description automatically generated



1. **Internal interface Meta Class creation:**

* In the **internal\_interfaces** folder, keep all the interface files which will contain functions related to internal test cases. Naming convention:

1. File name (for eg.): **telemetry\_ifc\_int.py**
2. Class name (for eg.): **TelemetryIfcInt(type)**

* Classes defined under internal interface files are Meta classes inheriting from 'type'.
* It will contain a function "\_\_new\_\_" with positional args: (cls, name, bases, data\_dict) as shown below.
* The interface modules for the internal test cases will be defined under \_\_new\_\_ as shown in the below example.

A screen shot of a computer program

Description automatically generated

* Note (important) - All built-in modules used in the inner function need to be imported within the respective function scope also as shown in the below example

A screen shot of a computer program

Description automatically generated

* After defining all inner functions, add the piece of code to the end of \_\_new\_\_ method of the (say, TelemetryIfcInt) class.

**def get\_function\_object(f\_def):**

**# compile the code containing the function definition**

**code\_obj = compile(ast.Module(body=[f\_def], type\_ignores=[]), filename='<ast>', mode='exec')**

**# evaluate the code object and obtain the function object**

**namespace = {}**

**exec(code\_obj, namespace)**

**return namespace[f\_def.name]**

# **iterate and fill data\_dict**

**src = inspect.getsource(TelemetryIfcInt)**

**node = ast.parse(src)**

**for func\_def in ast.walk(node):**

**if  type(func\_def).\_\_name\_\_ == 'FunctionDef' and func\_def.name !="\_\_new\_\_":**

**func\_object = get\_function\_object(func\_def)**

**data\_dict[func\_def.name] = func\_object**

**# Create the class using the super() method**

**new\_class = super().\_\_new\_\_(cls, name, bases, data\_dict)**

**return new\_class**

* This code will dynamically allocate all function objects against their function name in data\_dict dictionary by compiling the whole class using Abstract Syntax Tree(ast) module.
* Get\_function\_object() method converts the obtained FunctionDef object after compiling, to a callable Function object.

1. **External Interface class modifications:**

* In External interface Class definition add a parameter named "metaclass"

class TelemetryIfc(FunctionalIfc, metaclass=Meta):

* Add the import statement for internal ifc file with alias "Meta"

try:

   from internal\_interfaces.telemetry\_ifc\_int import TelemetryIfcInt as Meta

except:

   from utils.ctam\_utils import MetaNull as Meta

* MetaNull is an empty meta class which is imported when internal testing if turned off.

A screen shot of a computer code

Description automatically generated

1. **Internal Test case execution:**

* In test\_runner.json, set internal\_testing to true.
* Keep internal test cases in the internal test folder.
* Keep functions related to the internal test cases in the internal interface python file.
* Make sure interface files and classes follow the name convention mentioned at the start of the document.
* Set internal\_testing to false if only external test cases need to be executed.

1. **Merging internal test case to external:**

* When moving an internal test case to external, move the test case file to **tests** folder.
* Remove the function from internal interface file and keep it in the external interface file and remove the internal import statements, if any, within that function.