Counting Test Benches

User Tutorial for the Counting Test Benches

May 2, 2014

Institute for Software Integrated Systems
World-class, interdisciplinary research with global impact.

VANDERBILT
UNIVERSITY

1.0 Purpose

The purpose of the Counting Test Benches are to provide the designer with information regarding how many of a certain component classification are currently in a design, if the design contains the minimum of a required component class, etc.

2.0 Procedures

2.1 Installation

Initial installations will be provided with the installation of the CyPhy tool suite. Future version may be packaged as a standalone or combined package for test benches.

2.2 Tool

The Counting Test Benches in GME consist of Python scripts that traverse the design, calculate the number of each component classification in the design, and return the metric of interest.

3.0 Requirements tested

- **Load Capacity Troops:** How many troop manikins the design can fit.
- **Troop Count:** How many troop manikins are currently in the design.
- **Transverse Center of Gravity (TCG):** Calculates the TCG of the current design.
- **Crew Capacity:** How many crew manikins the design can fit.
- **Crew Count:** How many crew manikins are currently in the design.
- **Count Portable Extinguishers:** Number of fire extinguishers design currently contains.
- **Count Bilge Subsystems:** Number of bilge subsystems (4 pumps = 1 system) the design contains.
- **Bilge Pump Capacity:** The total volumetric flow rate of all bilge pumps in design.
- **Internal BII Stowage:** Checks if all required internally mounted basic issue items (and their minimums) are accounted for in design. See Addendum 2 for required BII items.

- **External BII Stowage:** Checks if all required externally mounted basic issue items (and their minimums) are accounted for in design. See Addendum 2 for required BII items
- **Stored Rounds:** Number of ammunition boxes that are currently stored in design.
- **Count COMs:** All specified SA/ BFT and COMS C4I list is contained and mounted within the vehicle. See Addendum 2 for required COMS components.

4.0 Required Components

There must be at least one component in the system under test, however this component can be of any type.

5.0 Theory of Operation

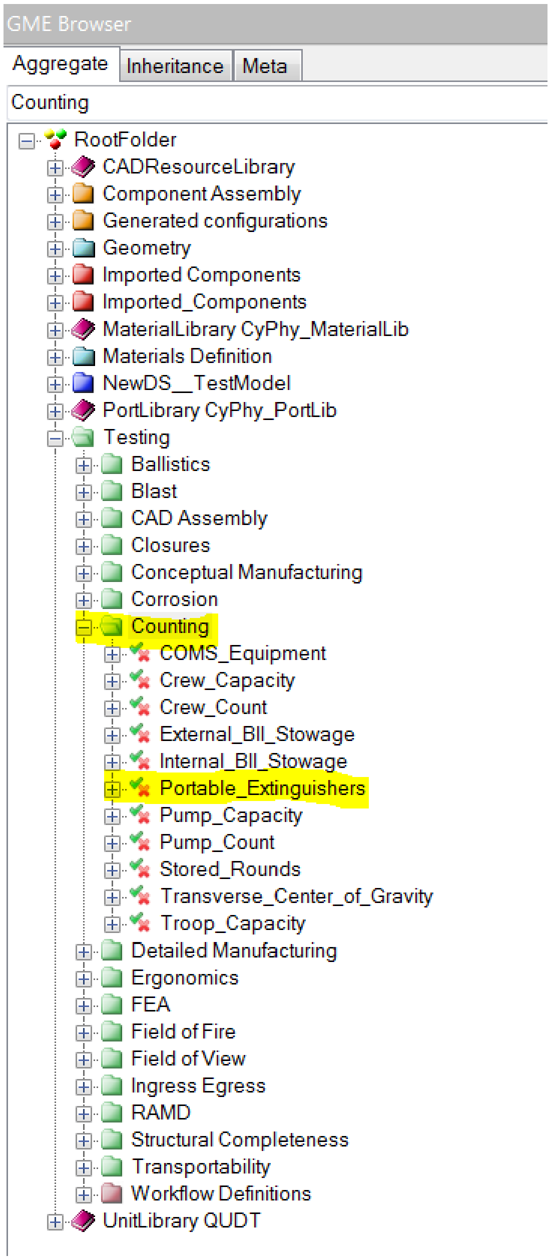
The system (design) is traversed and analyzed using Python. For the Transverse Center of Gravity Test Bench, the system (design) is assembled into a 3D CAD representation with the customization / generation of parameterized components. The data is analyzed to determine the TCG.

6.0 Test Bench Structure

The test bench contains a system under test that is assembled and analyzed for the specific metric that corresponds to the vehicle requirements as outlined in Table 1 below. As the setup is similar, the steps shown below for “Count Portable Extinguishers” can be applied to the other test benches described in this document. In the GME Browser, insert a new **Test Bench subfolder** (“Counting”) within the “Testing” **Test Bench folder**. Insert a new **Test Bench Model** (“Portable_Extinguishers”) to this subfolder.

> **NOTE:**
> As the setup is similar, the steps shown below for “Count Portable Extinguishers” can be applied to the other test benches described in this document. In this example all test benches are placed in the “Counting” subfolder.

NOTE:
Since the Transverse Center of Gravity test bench is grouped with the counting test benches but has a different setup, see the TCG Addendum 1 for its test bench structure.



GME Browser

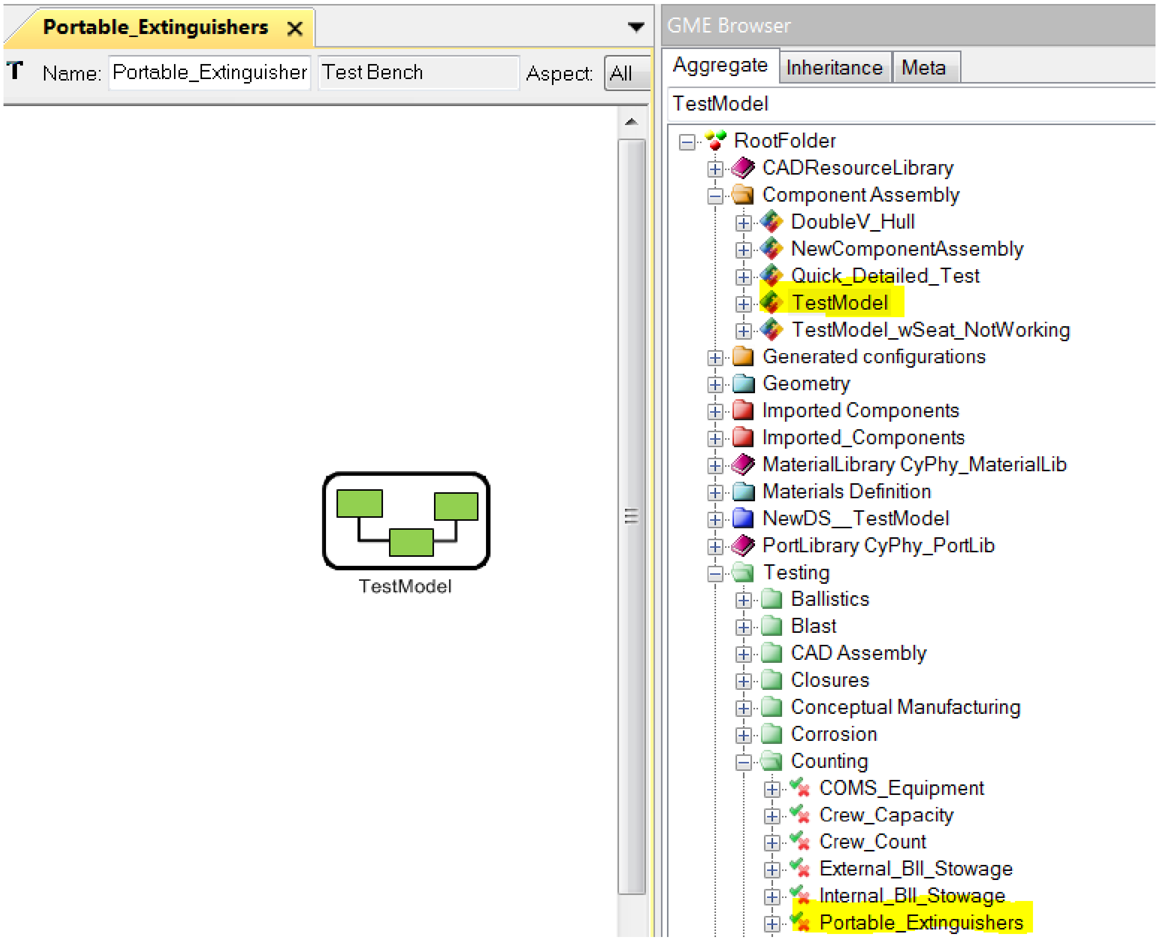
Aggregate | Inheritance | Meta

Counting

- RootFolder
  - CADResourceLibrary
  - Component Assembly
  - Generated configurations
  - Geometry
  - Imported Components
  - Imported_Components
  - MaterialLibrary CyPhy_MaterialLib
  - Materials Definition
  - NewDS__TestModel
  - PortLibrary CyPhy_PortLib
  - Testing
    - Ballistics
    - Blast
    - CAD Assembly
    - Closures
    - Conceptual Manufacturing
    - Corrosion
    - Counting
      - COMS_Equipment
      - Crew_Capacity
      - Crew_Count
      - External_BII_Stowage
      - Internal_BII_Stowage
      - Portable_Extinguishers
      - Pump_Capacity
      - Pump_Count
      - Stored_Rounds
      - Transverse_Center_of_Gravity
      - Troop_Capacity
    - Detailed Manufacturing
    - Ergonomics
    - FEA
    - Field of Fire
    - Field of View
    - Ingress Egress
    - RAMD
    - Structural Completeness
    - Transportability
    - Workflow Definitions
  - UnitLibrary QUDT

Figure 1: Setup test bench folder structure

An assembly now needs to be added to the test bench. In the "Portable_Extinguishers" test bench Copy/Paste…As Reference the assembly to be tested. When prompted, select **TopLevelSystemUnderTest**.



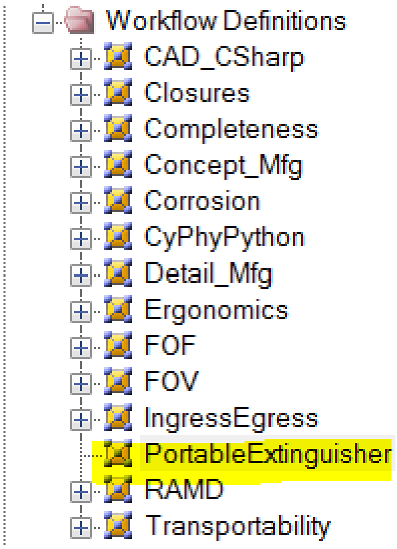
Name: Portable_Extinguisher Test Bench Aspect: All

Figure 2: Copy/Paste…As Reference

Now a Workflow Definition needs to be created. There are two options, depending on if you would like the script to automatically run each time the test bench is invoked, or if you would like to specify which Python script to run when you invoke the test bench.

6.1 Option 1 (Automatic):

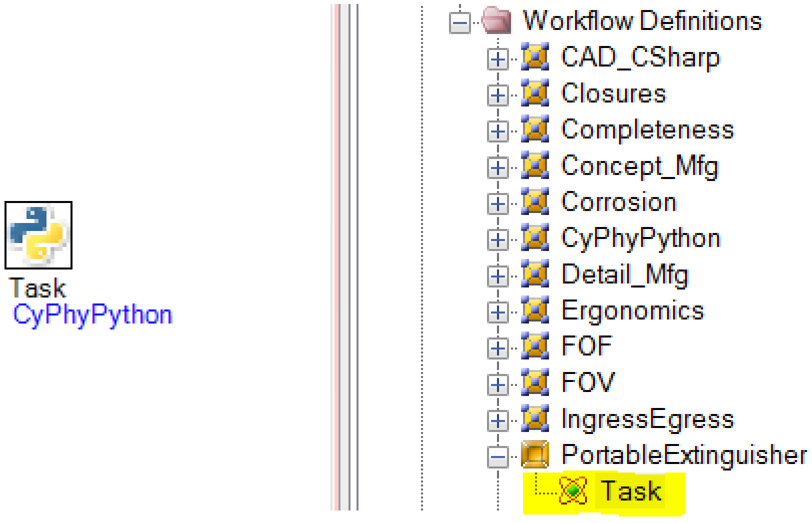
In GME, within the Workflow Definition subfolder create a new **workflow model** named “PortableExtinguisher”.



Workflow Definitions
CAD_CSharp
Closures
Completeness
Concept_Mfg
Corrosion
CyPhyPython
Detail_Mfg
Ergonomics
FOF
FOV
IngressEgress
PortableExtinguisher
RAMD
Transportability

Figure 3: Workflow definition

Open the “PortableExtinguisher” Workflow Model and drag a “Task” element into the workspace. Select “CyPhyPython Interpreter” as the interpreter from the window that pops up.

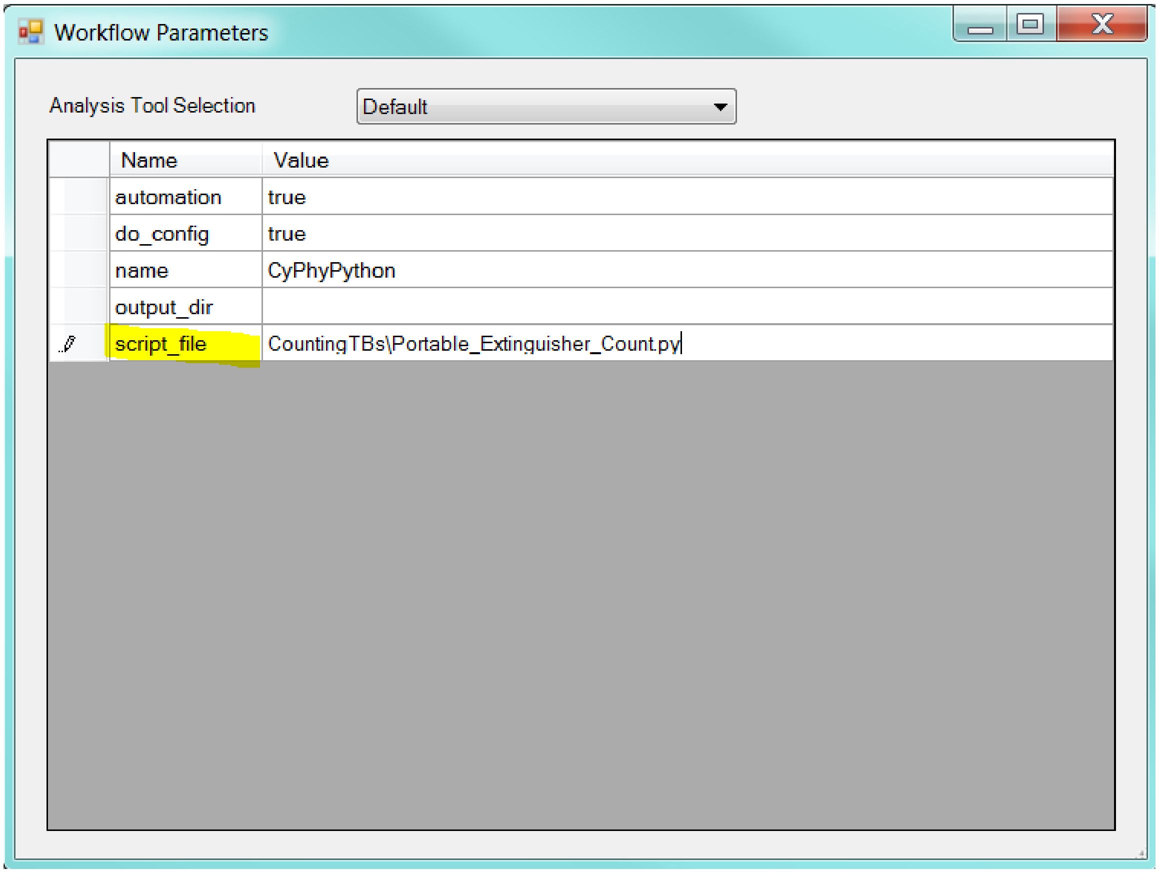


Task
CyPhyPython

Workflow Definitions
CAD_CSharp
Closures
Completeness
Concept_Mfg
Corrosion
CyPhyPython
Detail_Mfg
Ergonomics
FOF
FOV
IngressEgress
PortableExtinguisher
Task

Figure 4: Add a task to the PortableExtinguisher Workflow

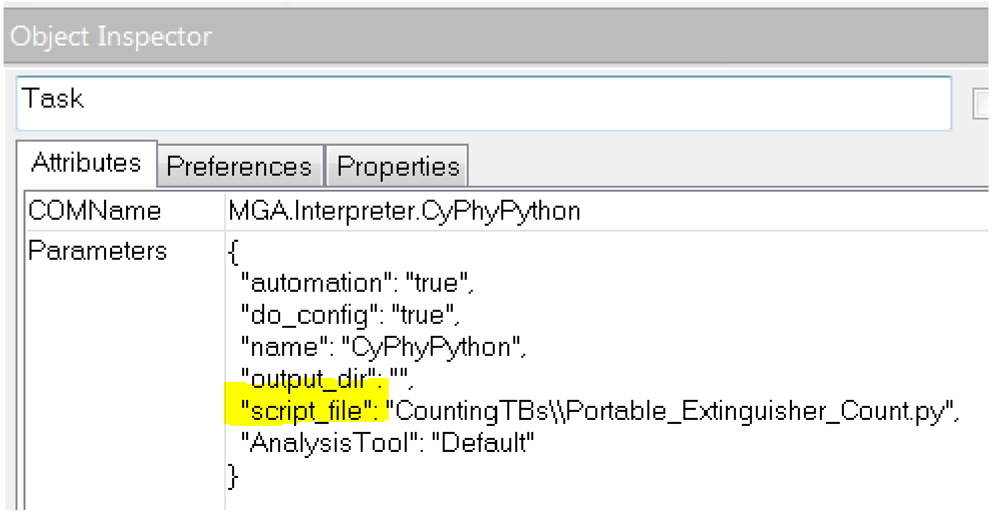
Double click the newly created task. For “script_file” enter the relative path (from your project directory) to the portable extinguisher Python script. Press Enter and exit. Highlighting the task your Object Inspector should specify the script you pointed to (Figures 5 and 6).



Analysis Tool Selection: Default

Name	Value
automation	true
do_config	true
name	CyPhyPython
output_dir	
script_file	CountingTBs\Portable_Extinguisher_Count.py

Figure 5: Setting Workflow parameters for the new Task



COMName: MGA.Interpreter.CyPhyPython

Parameters:

```
{
 "automation": "true",
 "do_config": "true",
 "name": "CyPhyPython",
 "output_dir": "",
 "script_file": "CountingTBs\\Portable_Extinguisher_Count.py",
 "AnalysisTool": "Default"
}
```

Figure 6: Object Inspector for the new Task

6.2 Option 2: Manually Select

Create a Workflow Definition as above, but name it “CyPhyPython”.
Do not specify a script file. Leave parameters as their defaults.

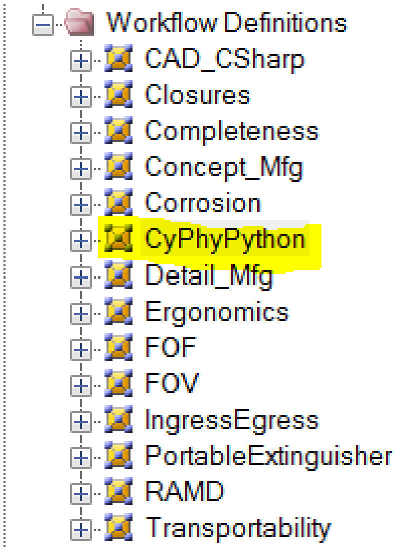


Figure 7: Add CyPhyPython Task

Impact of option 1 versus 2:

- For option 1, a new workflow will need to be created for each counting test bench so that each task points to the correct Python file.
- For option 2, only one workflow is needed. When the Test bench is invoked, the user will be asked to manually point to the desired Python script.

Open the “Portable_Extinguisher” test bench and drag and drop either the “PortableExtinguisher” or “CyPhyPython” workflow definition, followed by 1 metric. Metrics can be dragged and dropped from the Part Browser.

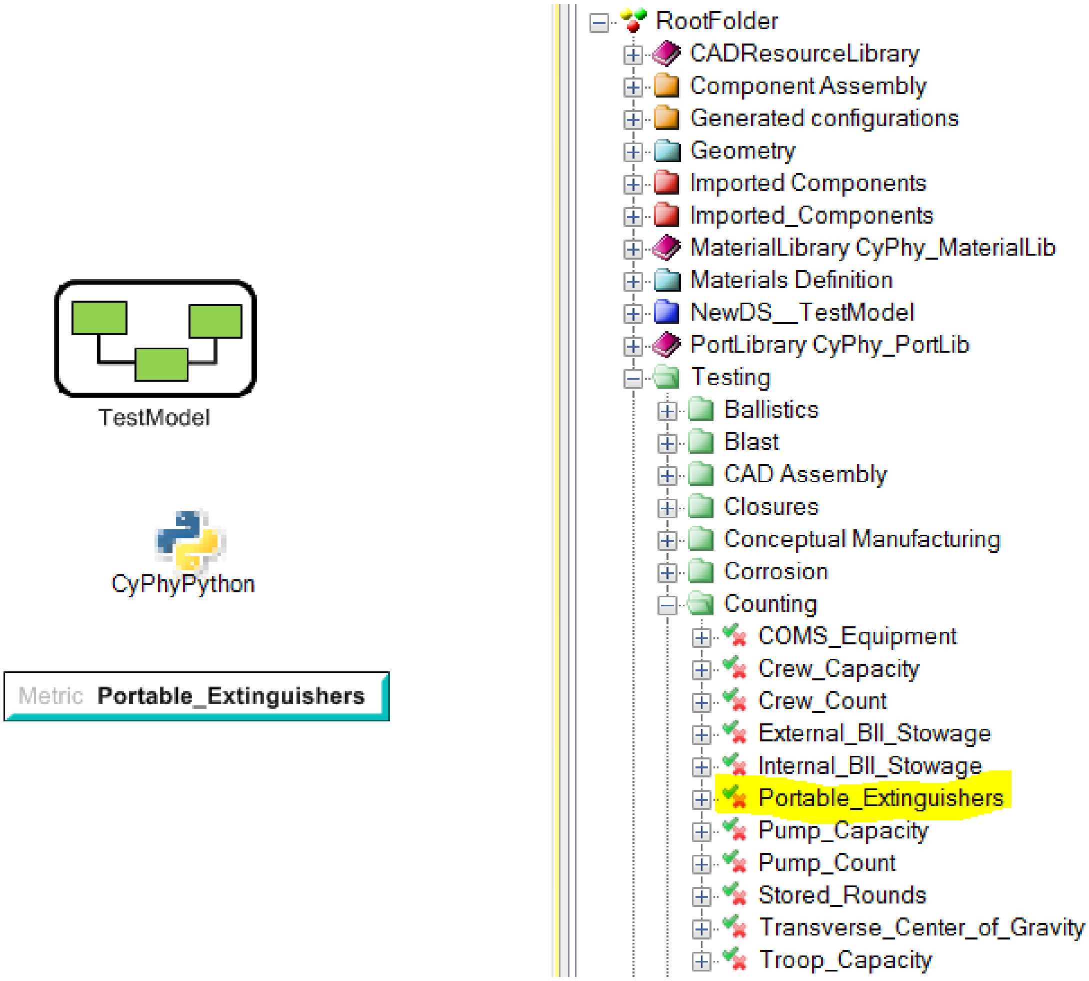
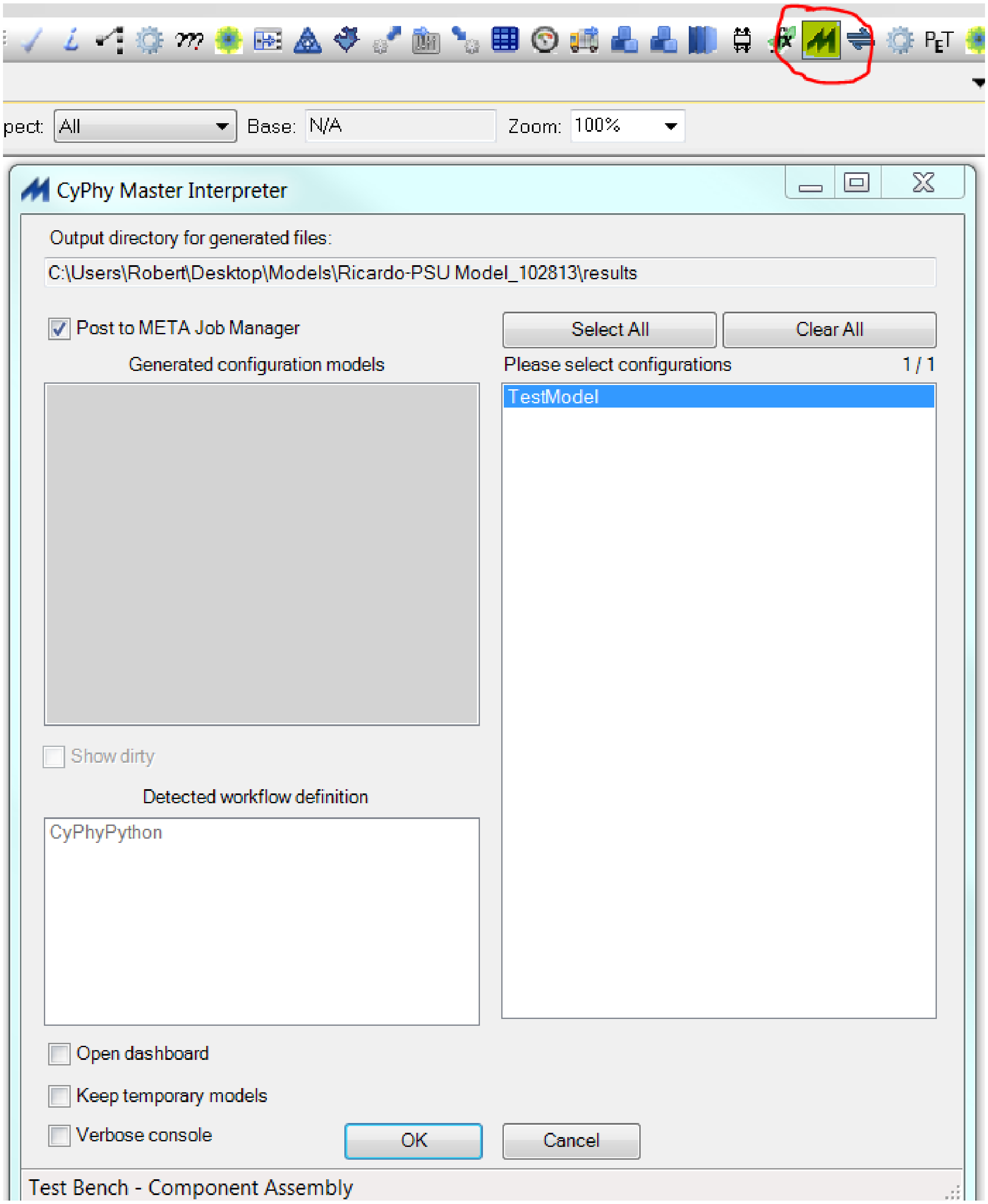


Figure 8: Portable Extinguisher Test Bench

To exercise the test bench, run the Master Interpreter (highlighted in task bar). Check the “Post to META Job Manager” as shown in Figure 9.



pect: All Base: N/A Zoom: 100%

CyPhy Master Interpreter

Output directory for generated files:

C:\Users\Robert\Desktop\Models\Ricardo-PSU Model_102813\results

☑ Post to META Job Manager

Generated configuration models

Please select configurations 1 / 1

TestModel

□ Show dirty

Detected workflow definition

CyPhyPython

□ Open dashboard

□ Keep temporary models

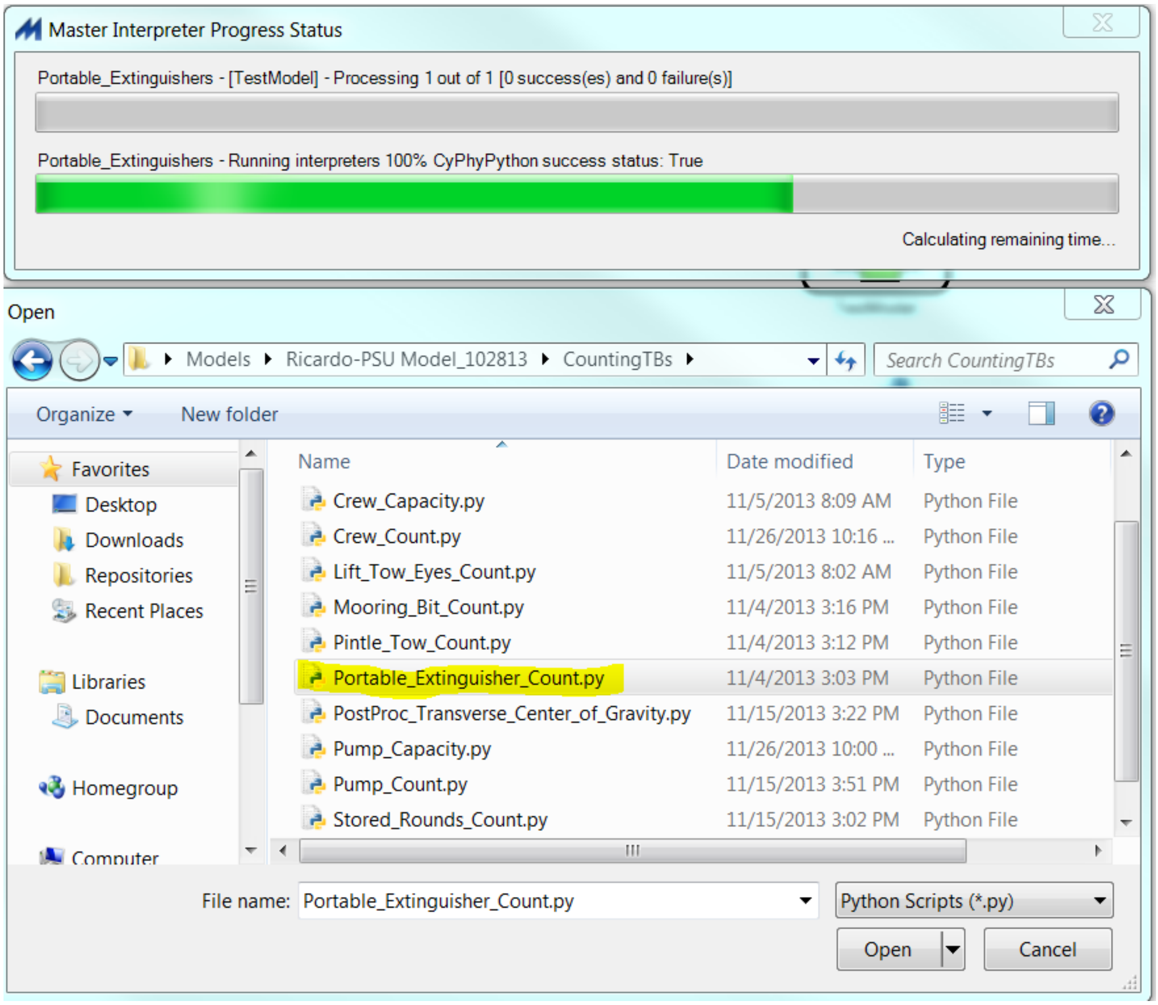
□ Verbose console

Test Bench - Component Assembly

Figure 9: Running the test bench

If you chose the CyPhyPython workflow, a dialog box will appear asking to point to the script. Locate the script and select “Open”. The job manager will then appear and execute if the test bench was correctly set up.

NOTE:
If you chose the PortableExtinguisher workflow, no dialog box will appear.



Portable_Extinguishers - [TestModel] - Processing 1 out of 1 [0 success(es) and 0 failure(s)]

Portable_Extinguishers - Running interpreters 100% CyPhyPython success status: True

Calculating remaining time...

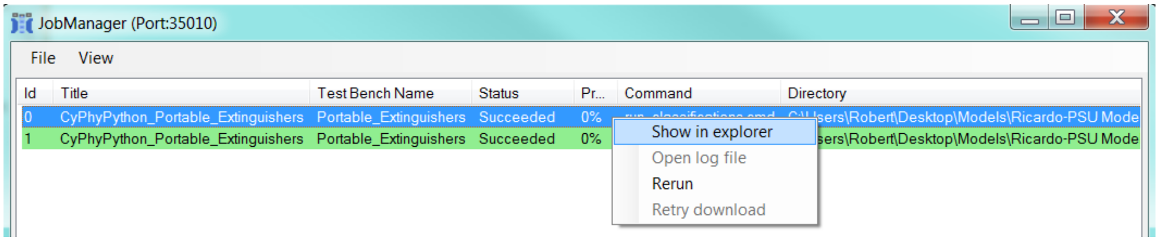
Name	Date modified	Type
Crew_Capacity.py	11/5/2013 8:09 AM	Python File
Crew_Count.py	11/26/2013 10:16 ...	Python File
Lift_Tow_Eyes_Count.py	11/5/2013 8:02 AM	Python File
Mooring_Bit_Count.py	11/4/2013 3:16 PM	Python File
Pintle_Tow_Count.py	11/4/2013 3:12 PM	Python File
Portable_Extinguisher_Count.py	11/4/2013 3:03 PM	Python File
PostProc_Transverse_Center_of_Gravity.py	11/15/2013 3:22 PM	Python File
Pump_Capacity.py	11/26/2013 10:00 ...	Python File
Pump_Count.py	11/15/2013 3:51 PM	Python File
Stored_Rounds_Count.py	11/15/2013 3:02 PM	Python File

File name: Portable_Extinguisher_Count.py

Python Scripts (*.py)

Figure 10: Selecting the Python script

The test bench will create a results folder and then run. To access the results folder right click the job in Job Manager and choose “Show in explorer”.



Id	Title	Test Bench Name	Status	Pr...	Command	Directory
0	CyPhyPython_Portable_Extinguishers	Portable_Extinguishers	Succeeded	0%	run_classifications.cmd	C:\Users\Robert\Desktop\Models\Ricardo-PSU Mode
1	CyPhyPython_Portable_Extinguishers	Portable_Extinguishers	Succeeded	0%		sers\Robert\Desktop\Models\Ricardo-PSU Mode

Figure 11: Accessing the test bench result folder

7.0 Description

The Counting Test Benches in GME consist of Python scripts that traverse the design, calculate the number of each component classification in the design, and return the metric of interest.

For all tests other than TCG, The System Under Test is traversed and analyzed in Python. For Transverse Center of Gravity, the System Under Test is assembled in CREO. A Python post processing script then calculates the TCG metric. Results are returned in "testbench_manifest.json", which is a summary of the test bench status and results.

8.0 Metrics

Type	TB #	Metrics	Units	Description
Load Capacity Troops	5	Troop_Capacity	Marines	Number of troop manikins design can support.
Troop Count		Troop_Count	Marines	How many troop manikins are currently in design.
Crew Capacity	7	Crew_Capacity	T/F	Number of crew manikins design can support. Returns True if number is greater than 3.
Crew Count		Crew_Count	Marines	How many crew manikins are currently in design.
Transverse Center of Gravity (TCG)	94	Transverse_Center_of_Gravity	mm	The TCG of the design.
Portable Extinguishers	146	Portable_Extinguishers	#	The number of fire extinguishers in the design.
Bilge Subsystems Count	147	Bilge_Pump_System	T/F	Counts number of bilge pump systems in design (4 pumps = 1 system). Returns True if count is greater than 4.
Bilge Pump Capacity	148	Bilge_Pump_Capacity	lpm	Returns volumetric flow rate of all bilge pumps.
Internal BII Stowage	165	Internal_BII_Stowage	T/F	Checks if all required internally mounted basic issue items (and their minimums) are accounted for in design.
External BII Stowage	166	External_BII_Stowage	T/F	Checks if all required externally mounted basic issue items (and their minimums) are accounted for in design.
Stored Rounds	169	Stored_Rounds	#	Number of ammunition boxes that are currently stored in design.
COMS Equipment	184	COMS_Equipment	T/F	All specified SA/ BFT and COMS C4I list is contained and mounted within the vehicle. Returns True if all communications components are contained in design.

Table 1: Counting Test Bench Metrics by vehicle role

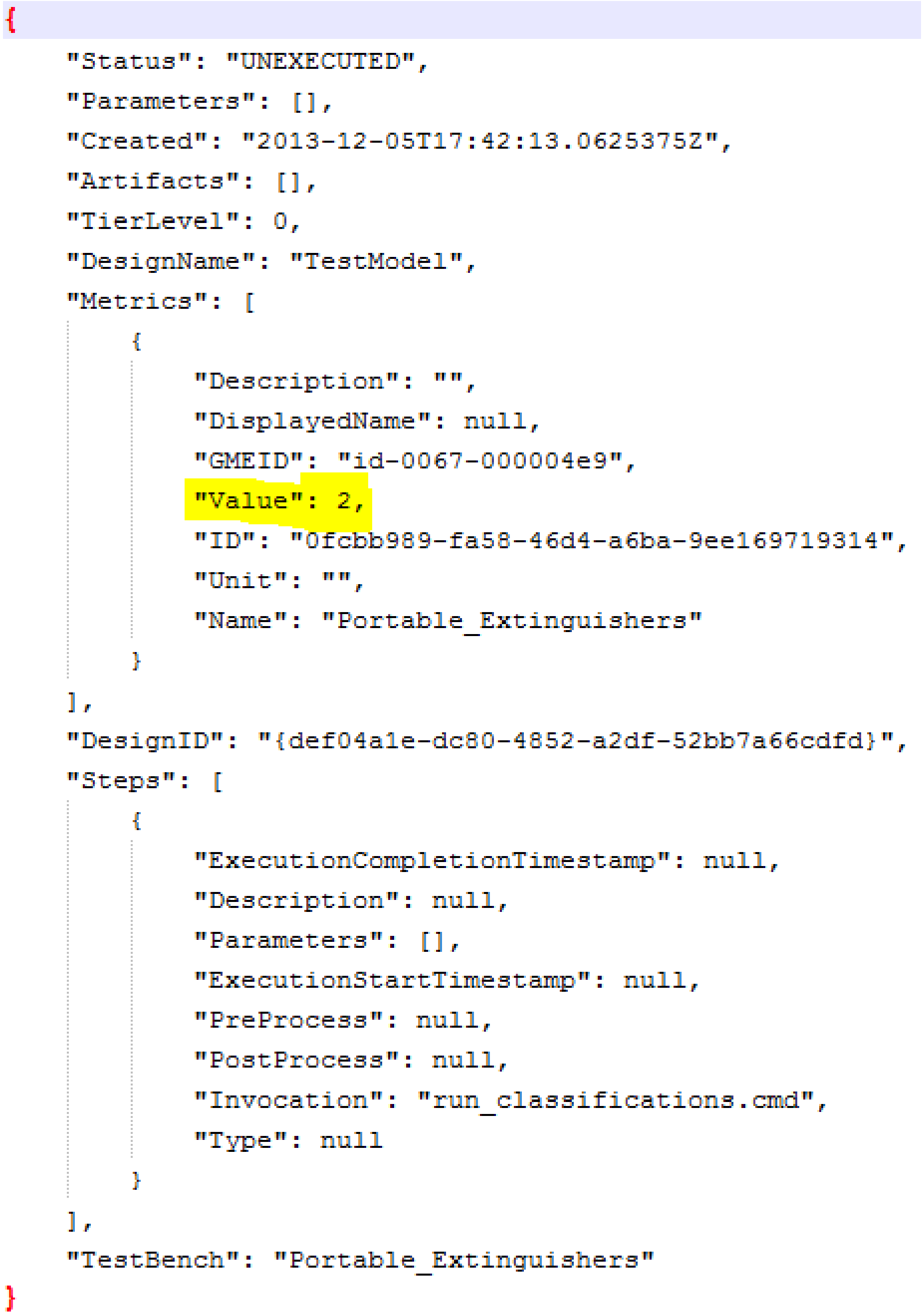
9.0 Required Connection to System Under Test

NONE

10.0 Outputs

Text & 2D

The output of the test benches are in “testbench_manifest.json”.



```
{
    "Status": "UNEXECUTED",
    "Parameters": [],
    "Created": "2013-12-05T17:42:13.0625375Z",
    "Artifacts": [],
    "TierLevel": 0,
    "DesignName": "TestModel",
    "Metrics": [
        {
            "Description": "",
            "DisplayedName": null,
            "GMEID": "id-0067-000004e9",
            "Value": 2,
            "ID": "0fcbb989-fa58-46d4-a6ba-9ee169719314",
            "Unit": "",
            "Name": "Portable_Extinguishers"
        }
    ],
    "DesignID": "{def04a1e-dc80-4852-a2df-52bb7a66cdfd}",
    "Steps": [
        {
            "ExecutionCompletionTimestamp": null,
            "Description": null,
            "Parameters": [],
            "ExecutionStartTimestamp": null,
            "PreProcess": null,
            "PostProcess": null,
            "Invocation": "run_classifications.cmd",
            "Type": null
        }
    ],
    "TestBench": "Portable_Extinguishers"
}
```

Figure 12: Summary Results sample

Addendum 1: Transverse Center of Gravity (TCG)

Test Bench Structure

Set up a **test bench folder** (“Transverse_Center_of_Gravity”) as described above.
Create a **workflow** as shown above (“CAD_CSharp”).
Insert a **task** and select the CyPhy2CAD_CSharp interpreter.

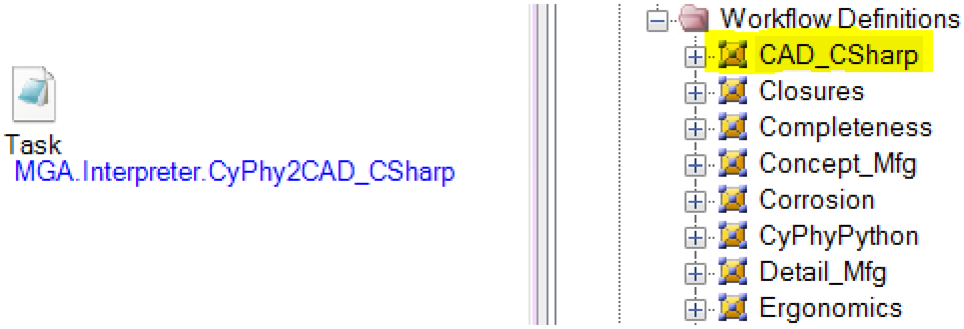
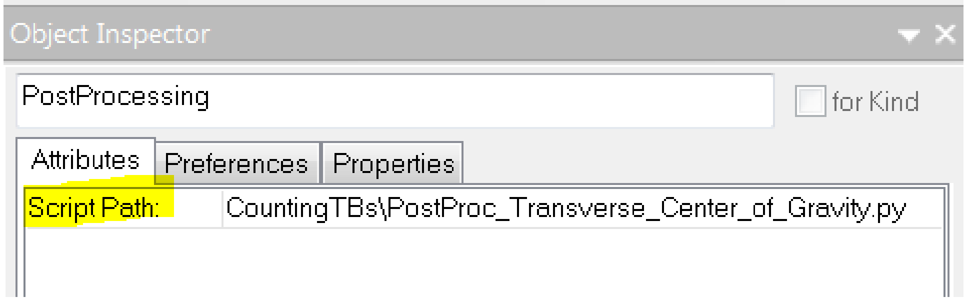


Figure 13: CAD Workflow Definition

The test bench requires 4 components:

- TopLevelSystemUnderTest
- CAD_CSharp workflow
- PostProcessing block
- Metric (“Transverse_Center_of_Gravity”)

Drag/Drop a PostProcessing block and point the script path to the TCG Python script.



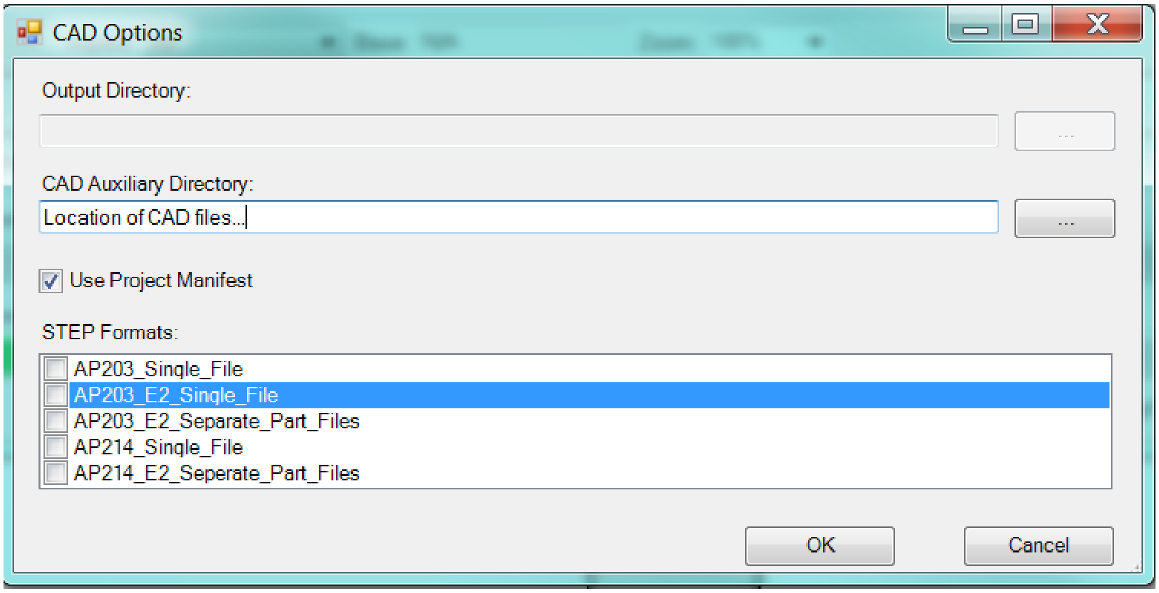
Object Inspector

PostProcessing

Script Path: CountingTBs\PostProc_Transverse_Center_of_Gravity.py

Figure 14: Post Processing script path

Run the Master Interpreter. Check "Use Project Manifest" and point to the location of the Creo files (if applicable). The results are in testbench_manifest.json, similar to the counting analyses.



CAD Options

Output Directory:

CAD Auxiliary Directory: Location of CAD files...

☑ Use Project Manifest

STEP Formats:

□ AP203_Single_File
□ AP203_E2_Single_File
□ AP203_E2_Separate_Part_Files
□ AP214_Single_File
□ AP214_E2_Seperate_Part_Files

Figure 15: CAD Options

Addendum 2: Required Component Classes

Internal/External Basic Issue Items

Class	Amount	Internal/External
axe	1	Internal
bag_pamphlet	1	Internal
bag_stowage	1	Internal
bag_tool_canvas	1	Internal
bar_tow	1	outside
box_tool	1	Internal
breaker_bar	1	External
cable_tow	1	External
can_jerry_coolant	2	Internal
can_jerry_diesel	2	Internal
can_oil	1	Internal
chock	4	Internal
crowbar	1	Internal
extinguisher_fire	2	Internal
flashlight	1	Internal
grease_gun	1	Internal
hammer_sledge	1	Internal
handle_mattock	1	External
holder_flag	1	Internal
hook_boat	1	External
jack_track	1	Internal
kit_cleaning_gun	1	Internal
kit_first_aid	1	Internal
lantern	1	Internal
litter_collapsed	1	Internal
litter_open	1	Internal
mattock	1	External
nato_slave	1	External
oiler	1	Internal
operators_manual	1	Internal

Class	Amount	Internal/External
padlock	5	Internal
pin_drift	2	Internal
rope_tow	1	Internal
searchlight	1	Internal
shackle	2	Internal
shoe_track	3	Internal
shovel	1	Internal
can_jerry_water	per crew/troop	Internal
pack_approach	per crew/troop	Internal
pack_assault	per crew/troop	Internal
weapon_personal	per crew/troop	Internal

COMS Equipment

Class	Amount
Crew_Unit_Vehicle_Intercom	1
Loud_Speaker_Vehicle_Intercom	1
Radio_Interface_Vehicle_Intercom	1
Radio_Vehicle	1