OpenCPI - Open Component Portability Infrastructure

An Open Source Infrastructure for Heterogeneous/Embedded Component-Based Systems and Applications

OpenCPI Unit Test Framework: Introduction

Application Execution

- Two ways to execute applications:
  1) `ocpirun`
     - Simple command-line program
     - Only supports static control
     - Used by the OpenCPI Unit Test Framework
  2) OpenCPI Application Control Interface (ACI)
     - Written in C++ or Python3
     - Allows for dynamic control

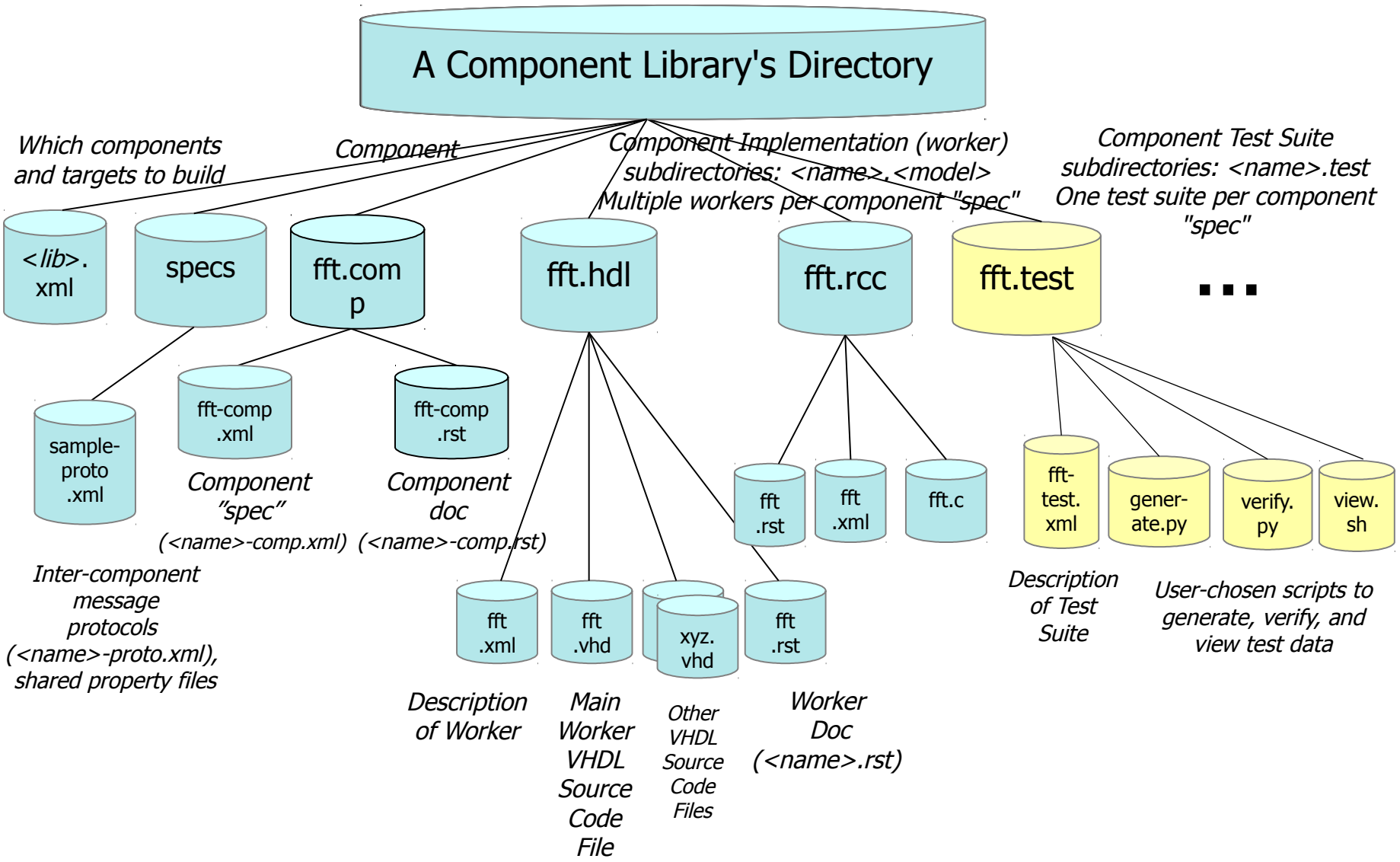
OpenCPI Unit Test Framework: Overview

- What it does:
  - Allows *functional* testing of all workers implementing a single component spec (OCS)
  - Stands alone from any user application
  - User provides "golden" input and output files or custom verification scripts
- Framework interactions tested as well
  - HDL "backpressure" randomly enforced to ensure worker properly respects flow control
  - HDL input metadata manipulations ensure worker can handle protocol edge conditions, *e.g.,* start-of-message without data until a later clock

OpenCPI Unit Test Framework: Overview (cont.)

- Based on a single component specification
  - <component>.test/
- Creates individual parameterized test cases based on
  - HDL hardware, simulation, and RCC target platforms
  - Multiple workers
    - one OCS implemented by many workers (authoring models and/or different implementations)
  - <worker>-build.xml - worker build configurations
    - Build-time/Compile-time (property parameters)
  - <component>-test.xml
    - Run-time settable property values from OCS and worker description (OWD)
    - Run-time test properties (optional)
    - Customized cases (ability to override the default of "all combinations" of the above)
- Supports
  - Application workers (focus of this training)
  - Device workers with emulators

Component Directory Layout: Test Suite



Create Component Unit Test Suite

- **$ ocpidev create test <component>**
  - Creates <component>.test/ containing skeleton files
    - <component>-test.xml
    - generate.py (optionally generate input test data; one script per input port)
    - verify.py (optionally verify output test data; one script per output port)
    - view.sh (optionally view data)
- **OpenCPI GUI**
  - Creates the same files as **`ocpidev`**

Unit Testing: Five Phases

- **Generate** – creates the "gen" sub-directory
  - generates input data, OAS, OHAD, case configuration(s)
- **Build** (HDL workers only; no-op for RCC workers)
  - builds HDL subassemblies for target platforms (bitstream/executable)
- **Prepare** – creates the "run" sub-directory
  - examines available built workers and platforms; creates execution scripts
- **Execute**
  - executes tests for all workers, configurations, test cases, and platforms
- **Verify**
  - verifies results from the execution of test cases on workers and platforms

OpenCPI GUI allows various combinations of these

Unit Testing Step 1: Generate

- Discovers
  - OpenCPI Component Specification (OCS) associated with this test directory
  - Workers in the same component library that implement this OCS
  - Build configurations (parameter values) for each worker per each <worker>-build.xml or <worker>.build
- Derives
  - Test cases that are "baselined" for build (parameter) configurations from any worker
  - The actual tests appropriate for all build (parameter) combinations vs. the actual worker build configurations they apply to
    - **default** is all, but may be reduced by the property element in the Unit Test Suite Description XML file
- Generates
  - OpenCPI Application Specification (OAS) XML files for launching unit tests
  - OpenCPI HDL Assembly Description (OHAD) XML files that are built for HDL platforms (hardware and simulation)
  - If necessary, generates input and/or property value files

Unit Testing Step 2: Build (HDL Only)

- Builds the required artifacts and executables for running tests, especially the generated HDL test assemblies for different workers and their build configurations
- This step requires specifying the target platforms on which testing should occur

Unit Testing Step 3: Prepare

- Discovers available
  - Execution platforms, both local and remote (network)
  - Built artifacts that can be executed on available platforms
- Generates
  - Test scripts to perform all feasible tests on all available platforms

Unit Testing Steps 4 and 5: Execute and Verify

- Interleaved (default) or Sequential
- Execute
  - All scripts are run per subcase
    - Scripts can access parameter properties and run-time properties of the subcase being tested
- Verify
  - Optionally execute view.sh to view test input/output data
    - " ocpidev run test ... --view "
  - Simulation data
    - Deleted upon successful verification (PASSED) by default
    - " ocpidev run test ... --keep-simulations " retains output data after verification, regardless of results

Unit Test Suite Description XML File

- <component>-test.xml
  - Specifies test *cases* and defaults that apply to *all* test cases
- input
  - Defines a prepared input file or generator script
- output
  - Defines an output file or verify script
- property
  - Defines property values for all test cases (OCS, OWD, or test)
- case
  - Used to define a non-default test case when needed
  - Necessary when the automatic parameterization of the default test case is invalid or excessive for testing the worker(s)
  - When no case element is defined, the default is used (common)

Unit Test Suite Description XML File: Example 1

(in <ocpi-install-path>/projects/tutorial/components/complex_mixer.test/)

```
<!-- This is the test xml for testing component "complex_mixer" -->

<Tests UseHDLFileIo='true'>

  <!-- Here are typical examples of generating for an input port and verifying results at an output port-->

  <Input Port='in' Script='generate.py 100 12.5 32767 16384'></Input>

  <Output Port='out' Script='verify.py 100 16384' View='view.sh'></Output>

  <!-- Set properties here. Use Test='true' to create a test-exclusive property. -->

  <Property Name='phs_inc' Values='8192'></Property>

  <Property Name='enable' Values='0,1'></Property>

  <Property Name="data_select" Values="0,1"></Property>

</Tests>
```

Unit Test Suite Description XML File: Example 2

(in <ocpi-install-path>/projects/assets/components/dsp_comps/cic_int.test/)

```
<!-- This is the test xml for testing component "cic_int" -->
<tests timeout='20000'  useHDLFileIo='true'>
  <input port='in' stressorMode='full' script='generate.py 32767' messagesize='8192'/>
  <output port='out' script='verify.py 32767' view='view.sh'/>
  <property name='TARGET_FREQ' test='true'/>
  <property name='VIVADO_ILA_p' values='0'/>
  <!-- Exclude higher R value test cases on simulators at to reduce test time -->
  <case>
    <!-- R=2048 is not excluded when TARGET_FREQ=0 to test the case of the -->
    <!-- output port producing multiple EOMs after EOF on input in simulation -->
    <property name='TARGET_FREQ' value='0'/>
    <property name='R' values='8191,8192' exclude='*sim'/>
  </case>
  <case>
    <property name='TARGET_FREQ' value='1'/>
    <property name='R' values='2048,8191,8192' exclude='*sim'/>
  </case>
</tests>
```

Unit Test Suite Description XML File (cont.)

Tests element Attribute	Data Type	Description
Spec	string	Overrides the default behavior where the OCS is inferred from the name of the <component>.test directory, and found in the ../specs/<component>-spec.xml file.
UseHdlFileIO	boolean	Applies only when HDL workers are being tested on simulation platforms. When true, file I/O is handled in the simulator using file read/write HDL workers (faster). When false (default), file I/O is handled using file read/write RCC workers outside the simulator.
ExcludeWorkers	string	A list of comma-separated workers that should *not* be tested, even if they implement the spec of this test suite.
OnlyWorkers	string	A list of comma-separated workers that should be the *only* ones tested, and others found to implement the same spec will be ignored.
ExcludePlatforms	string	A list of comma-separated platforms that should *not* be tested. Any other available platforms that have built artifacts will be used.
OnlyPlatforms	string	A list of comma-separated platforms that should be the *only* ones tested. Any other available platforms will be ignored.
Duration	integer	An amount of time the application should run before being considered successfully ***finished***. **Cannot be used with *Timeout.***
Timeout	integer	An amount of wall clock time in seconds after which the execution of a test subcase is considered a failure. **Cannot be used with *Duration.***

Unit Test Suite Description XML File (cont.)

Tests element child element	Attribute	Type	Description
input	name	string	Optionally specifies the name of this input source. Not necessary if applied to all cases, but useful when shared across multiple cases. More common to use the port attribute if it applies only to a specific port. **Must specify either name or port.**
	port	string	Optionally specifies the name of the port that this input source will always apply to. If there is only one input source for a port, it will be used for all cases. **Must specify either name or port.**
	script	string	Used to indicate a command to execute to produce data. Can be a command name followed by command arguments, where the last argument is implicitly the file to be written into. **Must have execute permissions.**
	file	string	Used to specify the name of a file to be used as the source of data.
	messageSize	integer	A positive number that specifies the buffer size/size of messages (in Bytes).
	messagesInFile	boolean	Indicates that the input data contains message boundaries and opcodes.

```
<input port="in" script="generate.py 32768"/>
<input port="in" file="../../../applications/FSK/idata/Os.jpeg"/>
```

Unit Test Suite Description XML File (cont.)

Tests element child element	Attribute	Type	Description
output	name	string	Optionally specifies the name of this output source. Not necessary if applied to all cases, but useful when shared across multiple cases. More common to use the port attribute if it applies only to a specific port. **Must specify either name or port.**
	port	string	Optionally specifies the name of the port that this output source will always apply to. If there is only one output source for a port, it will be used for all cases. **Must specify either name or port.**
	script	string	Used to indicate a command to execute to validate data. Can be a command name followed by command arguments, where the last arguments are implicitly the output filename from the given port followed by each input port filename in the order the ports are declared in the OCS. **Must have execute permissions.**
	file	string	Used to specify the name of a golden file used to compare to output data.
	view	string	Optionally "view" the data for the port in some viewing or plotting window. Takes all of the same arguments as the verification script mentioned above.

Unit Test Suite Description XML File (cont.)

Tests element child element	Attribute	Type	Description
property	name	string	Required. Identifies a property defined in the OCS or OWD, or declares a new test property.
	test	boolean	Optionally indicates that the property is a test property and not a property of the worker(s) being tested. Must be a unique name not already used in the OCS/OWD. Used to generate other test cases not defined simply by the values of worker properties.
	value		Specifies a single value to be tested.
	values		Specifies a comma-separated sequence of values to be tested.
	valueFile	string	Specifies the name of a file containing a single value to be tested. Multiple lines in the file are considered elements of a sequence or array value.
	valuesFile	string	Specifies the name of a file containing multiple values to be tested. Multiple lines are considered separate values to be tested.
	generate	string	Specifies a command to execute to create a file containing a value to be tested. Used when a property depends on others in a complex way. The last argument is implicitly the file to be written into. Must have execute permissions.

```
<property name="ref" value="0x1B26"/>
<property name="data_select" values="0,1"/>
```

```
<property name="taps" generate="gen_lpf_taps.py"/>
<property name="samplesPerBaud" test="true" value="25"/>
```

Unit Test Suite Description XML File (cont.)

Tests element child element	Attribute	Type	Description
case	Name	string	Optionally specifies the name of the test case. If omitted, the name of the case is **case** followed by a case number (zero-based) with at least two digits. May be used when specifying that only certain cases (rather than all) should be executed or verified.
	ExcludeWorkers	string	A list of comma-separated workers that should *not* be tested, even if they implement the spec of this test suite.
	OnlyWorkers	string	A list of comma-separated workers that should be the *only* ones tested, where others found to implement the same spec will be ignored.
	ExcludePlatforms	string	A list of comma-separated platforms that should *not* be tested. Any other available platforms that have built artifacts will be used.
	OnlyPlatforms	string	A list of comma-separated platforms that should be the *only* ones tested. Any other available platforms will be ignored.

Unit Test Suite Description XML File (cont.)

Case element child element	Description
input	Overrides the default input per port. If no **input** is defined for an input port, the default **input** is used as defined for the port under the top-level **tests** element. The **port** attribute may be used, but if the **name** attribute names an input source at the top-level that already has a **port** attribute, it may be omitted. When the **name** attribute is used both **file/script** attributes are disallowed.
output	Overrides the default output per port. If no **output** is defined for an output port, the default **output** is used as defined for the port under the top-level **tests** element. The **port** attribute may be used, but if the **name** attribute names an output source at the top-level that already has a **port** attribute, it may be omitted. When the **name** attribute is used both **file/script** attributes are disallowed.
property	Overrides default value(s). If no **property** element is present for a property under a **case** element, then the value(s) defined at the top-level are used. For parameter properties, the default value(s) tested are derived from value(s) defined in all worker(s)' build configurations, but this automatic default may be overridden (limited) by a **property** element either at the top-level or under a **case** element. A single test **case** can have multiple values for any property, where sub-cases are automatically generated for all combinations of property values for the test case whether specified at the top level as defaults sets of values or specified for the test case in the **case** element.

Additional Configurations

- The test XML also allows more advanced configurations, including:
  - File I/O with embedded opcodes (**MessagesInFile** XML attribute)
  - Manipulating incoming ports' metadata to help ensure worker handshaking compliance (**StressorMode** XML attribute)
  - Disabling backpressure on output ports (**disableBackpressure** XML attribute)
    - **NOT** recommended; *HDL worker implementations have had problems respecting backpressure in the past*

Controlling Unit Testing

- Test XML attributes, OpenCPI GUI and/or **`ocpidev build/run`** command-line options are available for controlling test suite phases and other aspects of building and executing unit tests
  - HdlPlatform(s) – controls the Build phase
  - OnlyPlatform(s), ExcludePlatforms(s) – controls Build/Prepare/Execute/Verify phases
  - View – causes the "view" script to be run whenever verification is requested
  - Keep-Simulations – causes the contents of the simulations directory to be retained after successful execution on simulation platforms
  - Case – specifies a wildcard pattern indicating which cases/subcases should be executed or verified

Test Suite Scripts

- All Parameter and Initial/Writable run-time property values are exposed to each script as environment variables
  - OCPI_TEST_<myprop>
  - Output scripts also have access to final values of Writable and Volatile properties
- Scripts may be parameterized by these values for the subcase being generated
- Exit status of zero indicates success, while a non-zero exit status indicates failure
  - Framework prints green/red colors for PASSED/FAILED based on the exit status (terminal settings permitting)
  - May write other informational messages to STDERR to be logged
- Scripts must be executable and found in the <component>.test directory
- C/C++
  - getenv("OCPI_TEST_myprop")
- Python3
  - os.environ.get("OCPI_TEST_myprop")
  - #!/usr/bin/env python3
- Bash shell
  - ${OCPI_TEST_myprop}
  - #!/bin/bash --noprofile

Test Suite view.sh Examples

- Should return "immediately" by launching viewers in the background

```
#!/bin/bash --noprofile
# Plot the time/fft of input file ($2)
${OCPI_PROJECT_DIR}/scripts/plotAndFft.py $2 complex 32768 100 &
# Plot the time/fft of output file ($1)
${OCPI_PROJECT_DIR}/scripts/plotAndFft.py $1 complex 32768 100 &
```

```
#!/bin/bash --noprofile
# Plot the time/fft of output file ($1), and use property value ($
{OCPI_TEST_num_zeros}) to calculate the number of real samples
${OCPI_PROJECT_DIR}/scripts/plotAndFft.py $1 real `echo "$
{OCPI_TEST_num_zeros}+1*100" | bc` 1 &
```

Unit Testing Generate/Build Summary

- Create the test suite via "ocpidev create test -S <OCS>" or GUI
- Modify the <component>-test.xml file with input/output/property/case
- Prepare input data files and/or input data generator scripts
- Prepare output data gold files and/or output verification scripts
- Execute the Generate phase
- Examine the gen/cases.txt report to verify the generated subcases
- For HDL workers, execute the Build phase to build HDL assemblies
- All Generate/Build results are found in the gen/ directory

Test Suite Limitations

- Supports application workers and device workers
  - **Otherwise, test suites are "hand crafted"**
- Test suites only use ***ocpirun*** – no mechanism for an ACI
  - Simple "delayed" properties are available
- Fixed application XML (OAS)
  - Application worker **MUST** propagate zero-length messages on opcode 0
    - **OR** set the top-level *duration* / *timeout* attributes of the <component>-test.xml
  - Any file I/O limited to capabilities of **`ocpi.core.file_read`** and **`ocpi.core.file_write`**

Executing Tests on Remote Test Systems

- Extends the OpenCPI unit test framework beyond the current development system
- Remote platforms specified by **OCPI_REMOTE_TEST_SYSTEMS**
  - Colon-separated list of remote test systems with four fields, each separated by "="
  - **Host name/IP address = SSH user name = SSH password = Project mount path on remote system**
  - Example: 10.11.12.13=root=root=/mnt/myproj:10.11.12.14=root=root=/mnt/myproj
- Remote test systems are accessed via **SSH**
- Project's directory on the development system **MUST** be NFS mounted by the remote test system and its mounted directory name must match the project's directory, as indicated by the fourth field above
- Remote and development systems should use the same release version

Tests Execution and Verification

- Execute the Prepare phase
  - By **default**, all local and remote platforms are considered
    - RCC, HDL hardware, and HDL simulators
  - Ability to filter which platforms are used for execution
    - **OnlyPlatform(s)="xsim, centos7"** and **ExcludePlatform(s)="xsim, e3xx"**
  - Discovers available RCC-built artifacts in the component library, and **local** HDL artifacts built during the build phase from generated **HDL test assemblies**
  - From the combination of available platforms and available artifacts, it determines which sub-cases can be run on which platforms and generates test execution scripts accordingly in each run/<platform> sub-directory
  - Execution of unit tests overrides any user-specified OCPI_LIBRARY_PATH
    - Limits the artifact search to local RCC artifacts in the component library, the gen/assemblies directory for HDL artifacts, and the installed CDK on both local and remote systems for both file_read/file_write when RCC implementations are used

Tests Execution and Verification (cont.)

- Execute the Execute phase
  - Test applications are executed for all possible cases/sub-cases per platform
  - Ability to filter which platforms are used for execution
    - **OnlyPlatform(s)="xsim, centos7"** and **ExcludePlatform(s)="xsim, e3xx"**
  - The recorded results, per sub-case and platform, are
    - Output data from each output port
    - Final values of all properties, including volatile properties
    - A log file of the actual test execution

Tests Execution and Verification

- Execute the Verify phase
  - By **default**, all output results are verified
    - RCC, HDL hardware, and HDL simulators
  - Ability to filter which platforms are used for execution
    - **OnlyPlatform(s)="xsim, centos7"** and **ExcludePlatform(s)="xsim, e3xx"**
  - Relies on previous execution of appropriate sub-cases for all platforms
    - Performs verification using results previously recorded in the Execute phase
  - The Verify phase by itself does not involve execution or access to local or remote platforms and may be performed off-line
  - The View option enables running the view scripts during verification
    - This occurs per sub-case before each verification
  - The Verify phase will fail if any sub-cases fail by returning non-zero exit status
    - Each sub-case individually reports PASSED/FAILED status