SCMl Test Suite

enabling self-testing of SCMl implementations

Saul Romero, Dean Arnold

Keynote room (Grand Peninsula D)
27/09/17
Introduction
**System Control and Management Interface (SCMI)**

- A set of operating system-independent software interfaces that are used in system management.
- SCMI provides two levels of abstraction, protocols and transports.
- Currently provides interfaces for Discovery (self-description), Power Domain, Performance, Clock and Sensor Management.
SCMI Test Suite

• *What:* The SCMI test suite is a C-source code project available in public for SCMI protocol conformance testing.

• *Why:* It needs to be available for partners and adopters to self-check if their implementation is conforming to ARM SCMI PDD.

• *How:* Delivers a library of tests to validate firmware’s implementation of SCMI protocol.

• [https://github.com/ARM-software/scmi-tests](https://github.com/ARM-software/scmi-tests)
Figure 1 Example system that implements SCMI.
SCMI Test Suite Design
Test Suite Design requirements and constraints

• Provide abstraction layer to make test suite agnostic to platform or transport layer implementation.
  • Source code delivery with clear interfaces for platform specific adaptation.
  • Needs to be able to run on any agent type – test agent on AP/MCP/BMC
  • No assumptions about whether the platform is ARMv7 or ARMv8.
• Able to run on baremetal and OS level.
• Discovery driven execution mode.
• Allow selective inclusion of test libraries to facilitate testing on memory constrained platforms.
• Allow extensibility and customisation of the test suite to accommodate custom protocols and custom commands as defined by the user.
• Not a system level test suite; Focus on validating individual commands in protocols.
Test Suite components

Test engine
1. Load test case and expected values.
2. Call generic or custom test function.
3. Call post-processing function.
4. Report results

Expected values, agent config

Platform interfaces
Transport Logger

Send/receive data and dump output

Platform code (SUT)

SCMI protocols

Custom

Base
- Discovery test suite
  - Test cases
- Test suite (command)
  - Test cases
- ... Test suites
Adaptation Layer Interfaces

The below interface is used by the test agent component to send a composed SCMI command to the platform and receive the response.

- `int arm_scmi_send_message(uint32_t message_header_send, size_t parameter_count, const uint32_t *parameters, uint32_t *message_header_rcv, int32_t *status, size_t *return_values_count, uint32_t *return_values)`

The below interface is used by the test agent component to receive either a delayed response or notification from the platform.

- `int arm_scmi_wait_for_response(uint32_t *message_header_rcv, int32_t *status, size_t *return_values_count, uint32_t *return_values, bool *message_ready, const uint32 timeout)`

The below interface is used by the test agent component to dump output from the test execution.

- `void arm_scmi_log_output(const char* output)`
Test execution flow and code structure
Test execution flow

User executes test agent entry function

Expected values, agent characteristics

Discover implemented platform protocols and supported Base commands/suites

Test supported suites for Base protocol

Discover implemented platform protocols

More protocols to test? (Yes/No)

Send test results in a C-structure to the user

Dump test results

Discover supported/implemented protocol commands/suites

Test supported suites for the protocol
Test results
test_case_id: _base_protocol_version_query_protocol_version_01
case: _base_protocol_version_query_protocol_version_01
case_id: 79459
definition: scmi-ospm
extra: ...
result: pass
test monitoring timeout: 600 seconds

[base]-[PROTOCOL_VERSION]-query_protocol_version-01: CONFORMANCE
MESSAGE_ID = 0x01
CHECK HEADER: PASSED [0x00000001]
CHECK STATUS: PASSED [SCMI_STATUS_SUCCESS]
RETURN@0x0 'ATTRIBUTES_RESERVSED': PASSED [0]
RETURN@0x0 'ATTRIBUTES_NUMBER_AGENTS': PASSED [2]
RETURN@0x0 'ATTRIBUTES_NUMBER_PROTOCOLS': PASSED [5]
[base]-[PROTOCOL_ATTRIBUTES]-query_protocol_attributes-01: CONFORMANCE

ok: test case found
test_case_id: _base_protocol_attributes_query_protocol_attributes_01

case: _base_protocol_attributes_query_protocol_attributes_01
case_id: 79460
definition: scmi-ospm
extra: ...
result: pass
test monitoring timeout: 600 seconds

MESSAGE_ID = 0x02
PARAMETERS win32_i[1] = [3]
CHECK HEADER: PASSED [0x00000002]
CHECK STATUS: PASSED [SCMI_STATUS_SUCCESS]
RETURN@0x0 'MESSAGE_ATTRIBUTES': PASSED [0x00000000]
CHECK STATUS' message_id' = 0x0003: PASSED [SCMI_STATUS_SUCCESS]
[base]-[PROTOCOL_MESSAGE_ATTRIBUTES]-query_vendor_disc_cmd_support-01: CONFORMANCE

ok: test case found
test_case_id: _base_protocol_message_attributes_query_vendor_disc_cmd_support_01

case: _base_protocol_message_attributes_query_vendor_disc_cmd_support_01
case_id: 79461
definition: scmi-ospm
extra: ...
result: pass
test monitoring timeout: 600 seconds

MESSAGE_ID = 0x02
PARAMETERS win32_i[1] = [4]
CHECK HEADER: PASSED [0x00000002]
INFO STATUS: INFO [SCMI_STATUS_SUCCESS]
RETURN@0x0 'MESSAGE_ATTRIBUTES': PASSED [0x00000000]
CHECK STATUS' message_id' = 0x0004: PASSED [SCMI_STATUS_SUCCESS]
[base]-[PROTOCOL_MESSAGE_ATTRIBUTES]-query_subvendor_disc_cmd_support-01: CONFORMANCE
Future extensions

Implement support for testing delayed responses and notifications.

Implement ability to invoke multiple commands in a single test case.

Test for newer protocols as SCMI specification evolves.
device_type: juno

job_name: juno-semi-baremetal
timeouts:
  job:
    minutes: 30
  action:
    minutes: 5

priority: medium
visibility: public

actions:
  - deploy:
    timeout:
      minutes: 10
to: vmem
    recovery_image:
      url: http://seg-sw.cambridge.arm.com/downloads/personal/deaarm01/juno-17.07-ncp-baremetal.zip
      compression: zip
  - boot:
    method: minimal

  test:
    monitors:
      - name: flash
        start: 'Powering up system...
        end: 'Releasing system resets...
        pattern: "<\?xml\?><\?xml-stylesheet\?><resultstatus\>(\w+)</resultstatus>\[PASSED|FAILED]\)
        fixupdict:
          PASSED: pass
          FAILED: fail

      name: semi-cspm
      start: 'START---
      end: 'END---
      pattern: "<\?xml\?><\?xml-stylesheet\?><resultstatus\>(\w+)</resultstatus>\[CONFORMANT|UNTESTED|NON CONFORMANT]\)
      fixupdict:
        CONFORMANT: pass
        NON CONFORMANT: fail
        UNTESTED: skip

      name: semi-psci
      start: 'START---
      end: 'END---
      pattern: "<\?xml\?><\?xml-stylesheet\?><resultstatus\>(\w+)</resultstatus>\[CONFORMANT|UNTESTED|NON CONFORMANT]\)
      fixupdict:
        CONFORMANT: pass
        NON CONFORMANT: fail
        UNTESTED: skip
Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!
감사합니다
धन्यवाद