An Auto generated test framework for improving SAI interoperability
An Auto generated test framework for improving SAI interoperability

Ravi Vantipalli, Sr. SW Engineer, Intel
Rita Hui, Principal SW Engg. Manager, Microsoft
Agenda

• Motivation
• Goals
• State of SAI testing
• Thrift auto-generation framework
• PTF tests
• Call to Action
## Motivation

<table>
<thead>
<tr>
<th>SAI</th>
<th>Functional Definition</th>
<th>Quality</th>
<th>Updates</th>
</tr>
</thead>
</table>
| • Vendor agnostic API with over 50 objects and 1000+ attributes | • Lack of examples on API usage  
• Scope for ambiguities in API attribute functionality | • Non uniform test coverage  
• Lack of standard tests for a common bar for all vendors | • Lack of workflow for adding new tests for change/updates to SAI  
• Can result in poor quality / disparity in SAI implementation |
Goals

Quality

• Publish a unit and functional test plan for most SAI objects
• Increase SAI testing with a goal towards achieving a high and uniform bar for vendor implementations
• Provide a pseudo control-plane using SAI RPC (also auto-generated)

Extensibility

• Provide and fast and easily understood wrapper to invoke and test any libSAI implementation
• Simplify the effort it takes to add new tests going forward
# State of SAI API Testing

<table>
<thead>
<tr>
<th>Directory</th>
<th>Contents</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic_router</td>
<td>Simple test to illustrate L3 route setup</td>
<td>not been updated in over 3 years</td>
</tr>
<tr>
<td>sai_ut</td>
<td>C++ unit tests for L2/L3 objects using google-test</td>
<td>not been updated in over 3 years</td>
</tr>
<tr>
<td>saithrift</td>
<td>PTF based functional tests for L2, L3, Tunnel and Mirror</td>
<td>updates are very few and far between</td>
</tr>
<tr>
<td></td>
<td>As of today, ~125 testcases present in “saithrift”</td>
<td></td>
</tr>
</tbody>
</table>
Anatomy of a PTF test

PTF Server
- Unit test
  - saivlan.py
- Python Helper
  - sai_adapter.py

DUT
- libsai
- Vendor SAI implementation
- RPC Server
  - sai_rpc_server.cpp
  - sai.thrift
Current: To add a new test

Add an entry in `switch_sai.thrift`

```
add_sai_thrift_object_id_t
add_sai_thrift_create_debug_counter(1: list<sai_thrift_attribute_t> thrift_attr_list);
```

Add RPC server method for new entry in
`switch_sai_rpc_server.cpp`

```
add_sai_thrift_object_id_t
add_sai_thrift_create_debug_counter(const std::vector<sai_thrift_attribute_t>& thrift_attr_list)
{
    <function body>
}
```

Add a python wrapper in `switch.py` (if applicable)

Write the new test

We believe this is an impediment to writing more tests
Proposed: To add a new test

- **sai.thrift**
  
  ```
  sai_thrift_object_id_t sai_thrift_create_switch(1: list<sai_thrift_attribute_t> attr_list);
  sai_thrift_status_t sai_thrift_remove_switch();
  sai_thrift_status_t sai_thrift_set_switch_attribute(1: sai_thrift_attribute_t attr);
  sai_thrift_attribute_list_t sai_thrift_get_switch_attribute(1: sai_thrift_attribute_list_t a
  ttr_list);
  ```

- **sai_rpc_server.cpp**
  
  C++ thrift backend implementation for above thrift APIs are auto-generated

- **sai_adapter.py**
  
  Python wrapper for above thrift APIs, compliant to SAI CRUD semantics

- **All the above files auto-generated and readily available for every new object and attribute**

- **Just write the test**
SAI Thrift Generation

**SAI Meta**
- Utilize existing SAI meta to collect information about object/attributes
- Extend SAI Meta perl scripts to auto-generate wrapper files

**Code Generation**
- Provide new template files for auto-code generation
- New extended perl script + template files output .thrift, C++ wrapper + python thrift interface

**Test Creation**
- Write new tests using Auto-generated code
- Primary focus on tests rather than on boiler-plate code
- SAI Object definition/update need regeneration allows quick addition/update of tests
SAI PTF Tests

- Python based framework for unit and function testing
- SONiC community tests extensively using PTF

SAI Tests

- ~1000 documented functional testcases for SAI
- CRUD UTs for all supported objects and their attributes
- Generated using auto-generated thrift APIs

SAI PTF Tests

- Help describe API usage with 100s of examples
- Coverage for all supported objects
- Extensible
SAI PTF Tests

- The test cases include
  - CRUD unit tests
  - Functional tests

- The functional tests conform to **SAI behavior model**
- Allow for different implementations to have a consensus on API behavior
- Decrease cases of ambiguity for a given API
- Establish a common bar of quality for all SAI implementations
SONiC Community Tests

• All of these tests can run in a SONiC community test environment
• Build the libsaithrift using ENABLE_RPC functionality in SONiC
• Work in progress to define a community test topology to run new tests
• More details will be published soon
Call to Action

- We invite all vendors and partners to review and contribute testcases
- Help us review the design of the thrift wrapper
- Help is review the test cases
- Start adding your own PTF tests
Thank you!
Open Discussion