KEYSIGHT

SAI Challenger Enhancements for DASH Testing

Chris Sommers, Mircea Dan Georghe - Keysight
Anton Putrya, PLVision
2022-09-14

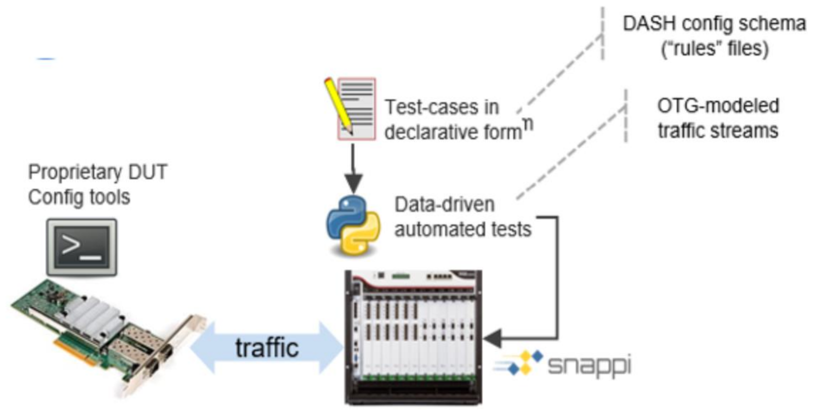
Why, What and When?

Why? *DASH stretches the limits of traditional testing!*

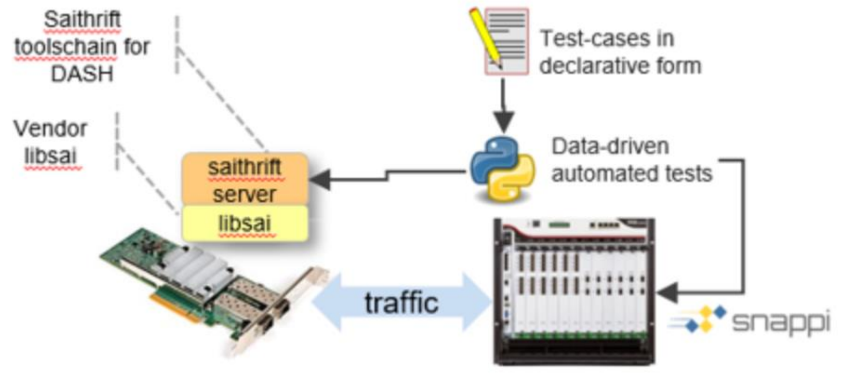
- Complex test-cases - many tables & interdependencies
- Huge table scale (millions of entries)
- Multiple APIs to test: SAI, sairedis, gNMI
- Performance testing of HW Targets (line rate)
- SW devs are increasingly expected to write test cases – how to make it easier?

Recap – DASH Test Maturity Stages

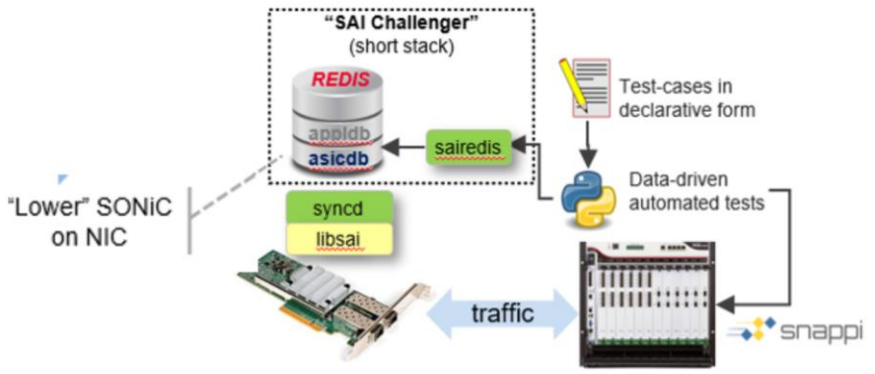
Data plane Testing Stage 2: Standardized, Automated Test Cases


DASH config schema ("rules" files)
Test-cases in declarative form
OTG-modeled traffic streams
Proprietary DUT Config tools
Data-driven automated tests
traffic
snappi

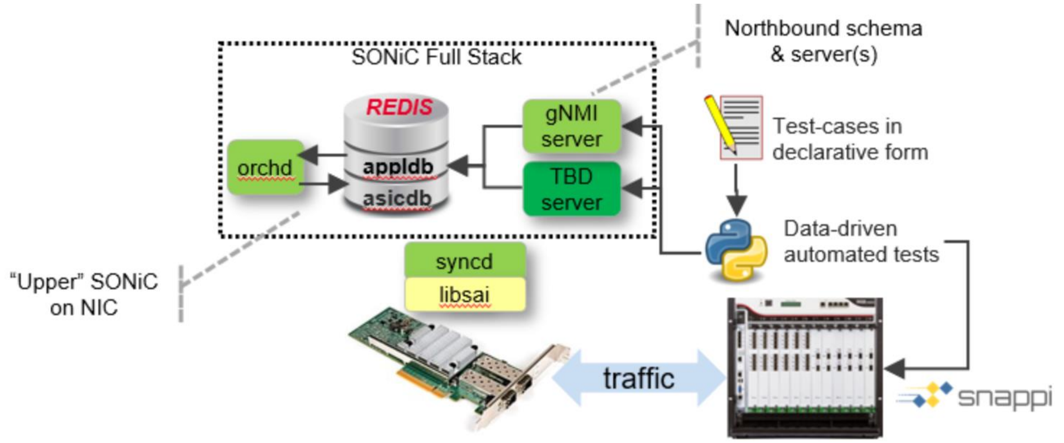
Data plane Testing Stage 3: DUT configuration via SAI-Thrift


Saithrift toolschain for DASH
Test-cases in declarative form
Vendor libsai
saithrift server
libsai
Data-driven automated tests
traffic
snappi

Data plane Testing Stage 4: DUT configuration via SAI-Redis


"SAI Challenger" (short stack)
REDIS
appldb
asicdb
sairedis
Test-cases in declarative form
"Lower" SONiC on NIC
syncd
libsai
Data-driven automated tests
traffic
snappi

Data plane Testing Stage 5: DUT configuration via SONiC Northbound API



KEYSIGHT https://github.com/Azure/DASH/blob/main/test/docs/dash-test-maturity-stages.md

Why, What and When?

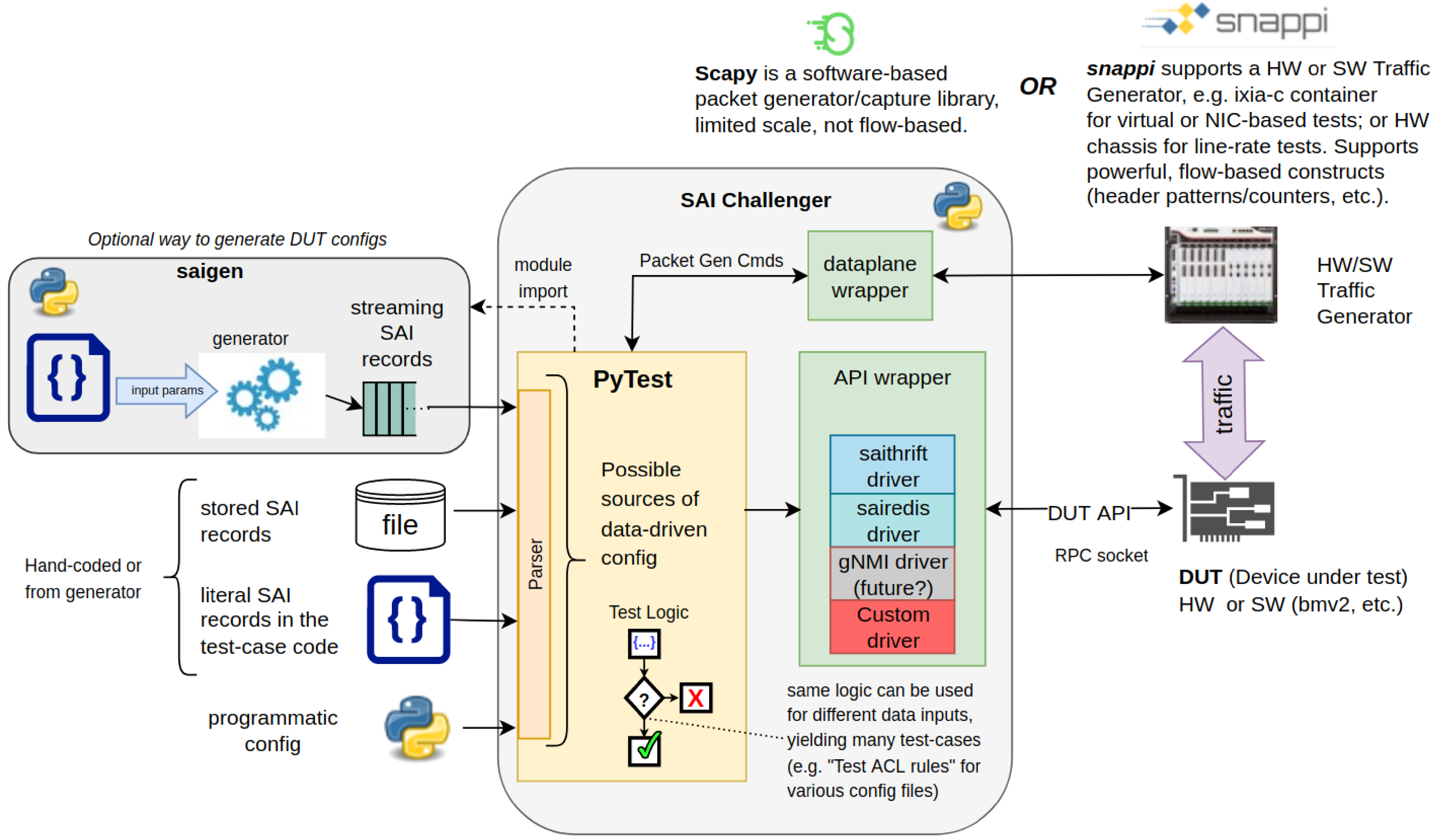
What? *GitHub contributions to SAI and DASH*

- OCP **SAI-Challenger** with Keysight-sponsored enhancements (for *any* SAI device)
- Keysight DASH Config generator can feed test cases for large-scale tests
- Increased developer productivity – focus on declarative configuration **data** and **test logic**, not API plumbing!
- Enhancements for multi-APIs, flexible traffic generators (SW or HW)

When? *Underway now!*

- “Early Preview” Demo today for community – looking for feedback
- First “release” ~ October

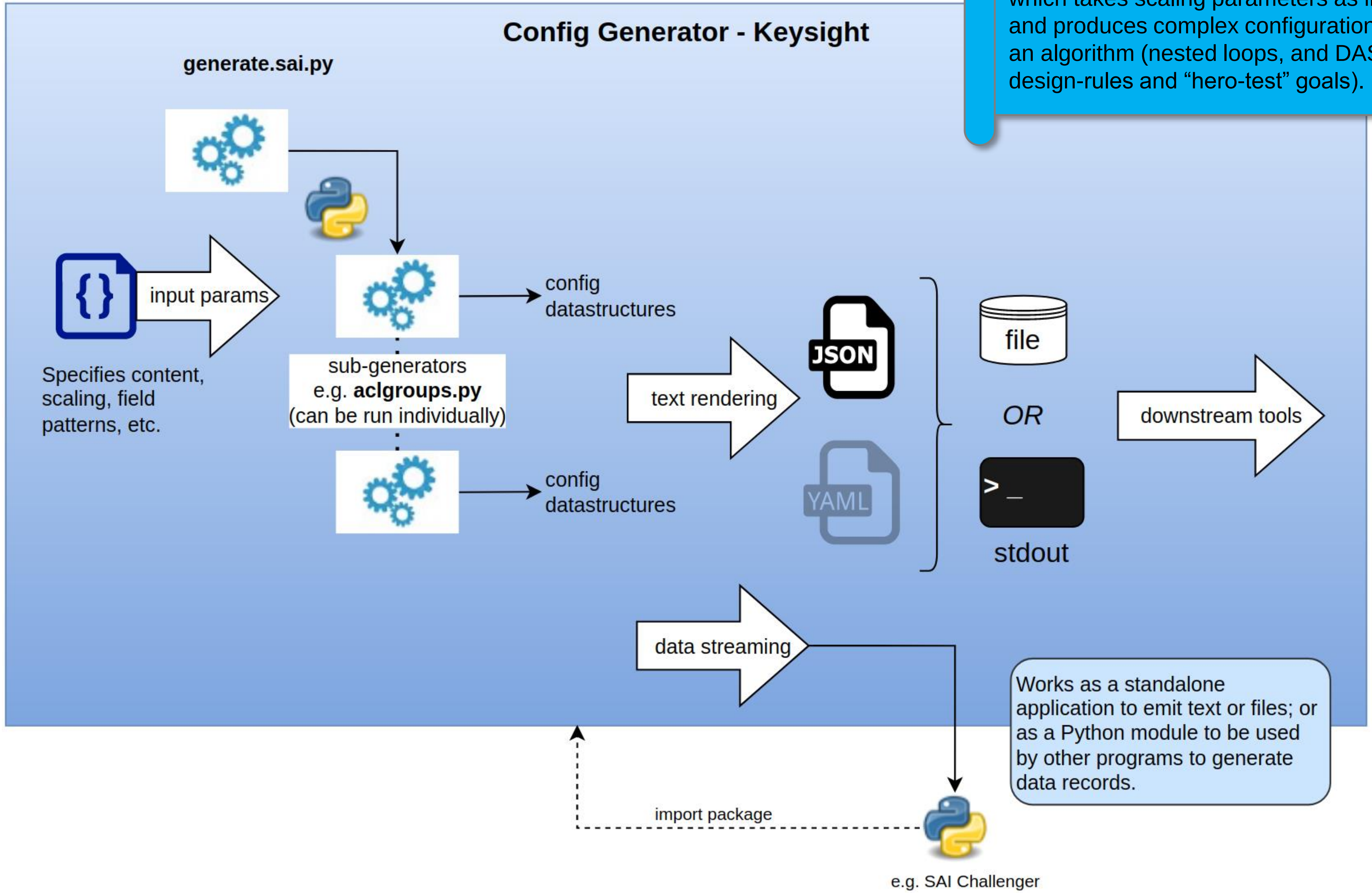
Framework at a Glance



Choice of SW/HW traffic generators
snappi
Scapy is a software-based packet generator/capture library, limited scale, not flow-based.
OR
snappi supports a HW or SW Traffic Generator, e.g. ixia-c container for virtual or NIC-based tests; or HW chassis for line-rate tests. Supports powerful, flow-based constructs (header patterns/counters, etc.).
SAI Challenger
Optional way to generate DUT configs
saigen
module import
Packet Gen Cmds
dataplane wrapper
HW/SW Traffic Generator
streaming SAI records
generator
input params
PyTest
API wrapper
traffic
saithrift driver
sairedis driver
gNMI driver (future?)
Custom driver
Possible sources of data-driven config
Parser
stored SAI records
file
DUT API
RPC socket
DUT (Device under test) HW or SW (bmv2, etc.)
Hand-coded or from generator
literal SAI records in the test-case code
Test Logic
{...}
?
X
programmatic config
same logic can be used for different data inputs, yielding many test-cases (e.g. "Test ACL rules" for various config files)

KEYSIGHT

Generator Details - Recap

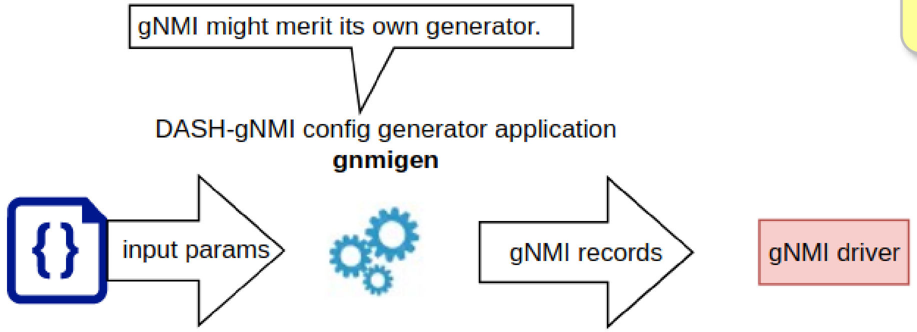


The config generator is like a “wizard” which takes scaling parameters as inputs and produces complex configurations using an algorithm (nested loops, and DASH design-rules and “hero-test” goals).

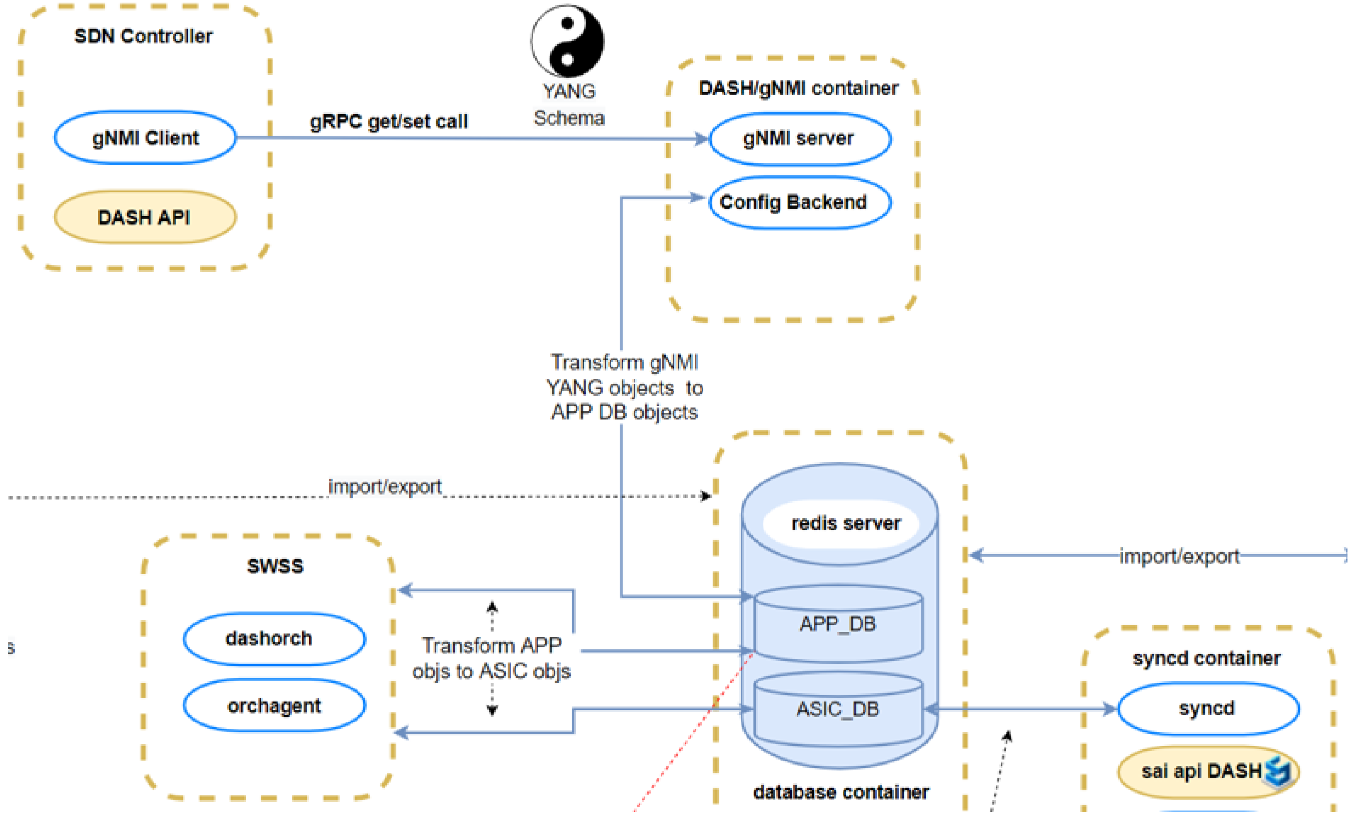
Works as a standalone application to emit text or files; or as a Python module to be used by other programs to generate data records.

gNMI Northbound API testing

- DASH gNMI and SAI schema are not 1:1 equivalent. The is some transformation in the dash-orch.
- May need modified gNMI DASH generator.
- Algorithms are same but output rendering modified to match gNMI schema.

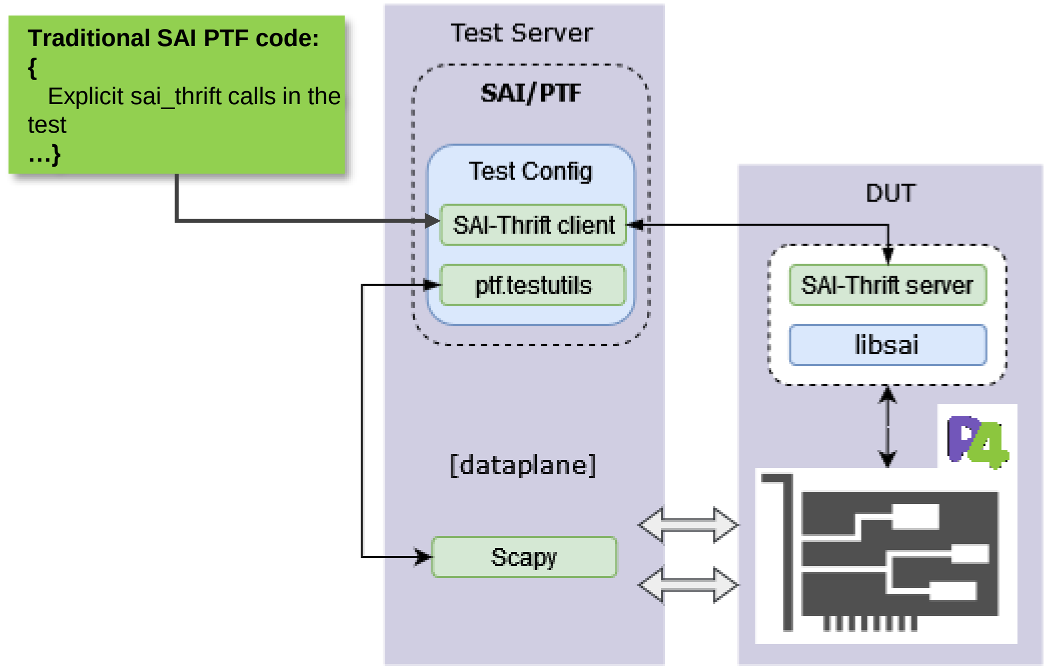


gNMI might merit its own generator.
DASH-gNMI config generator application
gnmigen
input params
gNMI records
gNMI driver

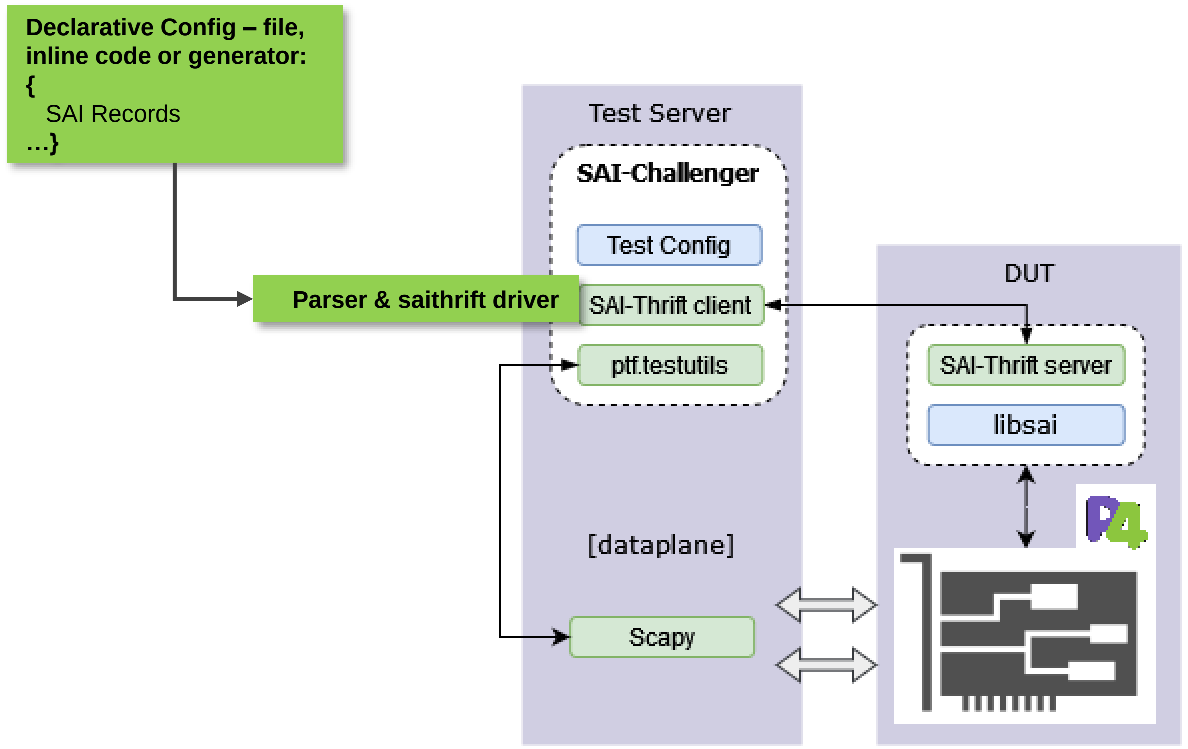


Demo Time!

Demo Setups – Traditional PTF vs. Declarative

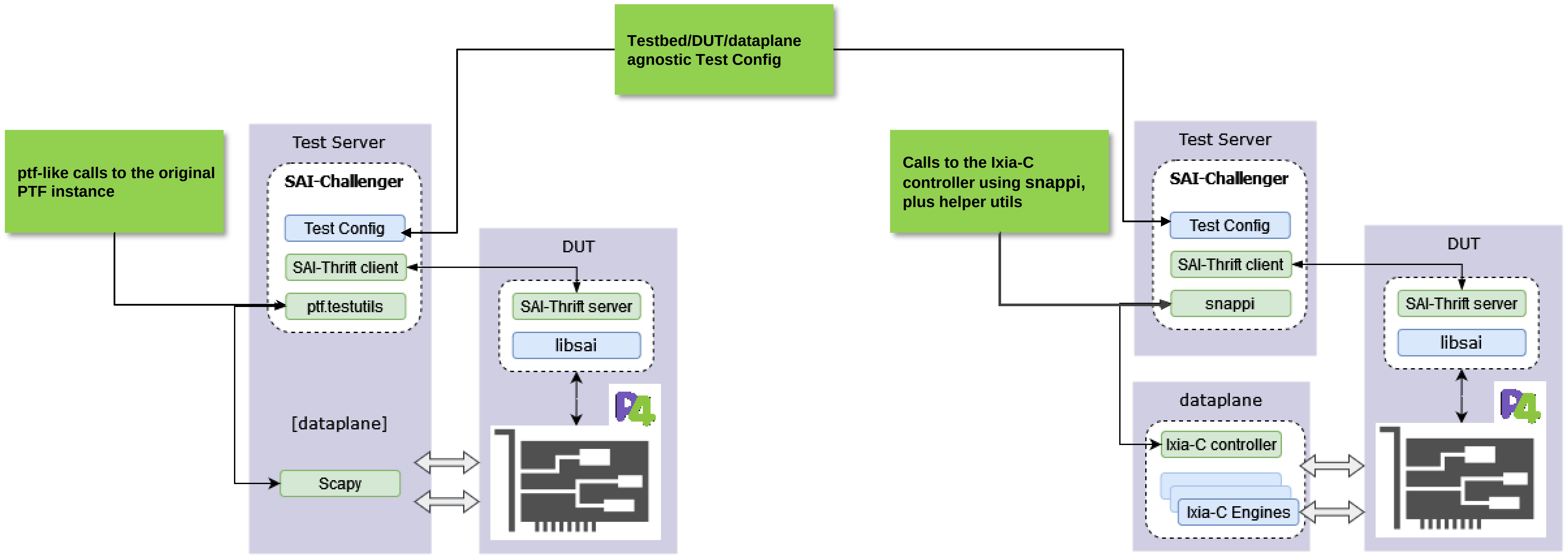


1. Traditional SAI PTF: API-specific config with PTF(Scapy) Dataplane



2. New: Declarative, abstract config with snappi/ptf Dataplane

Demo Setups –PTF & snappi based



1. Traditional PTF-based setup with new abstractions

2. New Ixia-C-based setup with snappi Dataplane

KEYSIGHT

Deliverables- ~ October 2022

- ***Everything will be upstreamed to OpenCompute/SAI and Azure/DASH***
- SAI challenger docker containers, integrated into DASH workflows including CI automation
- A few initial test-cases
- Tests will be verified against behavioral model using SW traffic generators.
- When vendors provide HW devices with SAI support (saithrift server, sai-redis), we’ll test with HW packet testers.

Other Possibilities – Not In Scope*

- gNMI data formats, generators, API drivers, test-cases

Community owner needed! Cool project!

- Port some libraries over to PTF to share the advances:
  - Data-driven SAI-Thrift tests (parser, driver)
  - Scapy/snappi wrappers for flexible SW/HW packet generation (caveats with PTF’s design)
- Single pane of glass – SAI challenger *could* run all existing PTF cases in SAI repo
- Integrating bmv2 into SONiC for full-stack emulation - worth the effort?
  - At a minimum, requires building a syncd daemon using bmv2 which would allow sairedis testing
  - Underlay for bmv2? Or wait for more compete SW switch e.g. p4DPDK w/ full SONiC stack?

**Not in project scope – volunteers or sponsor needed*

Community Call to Action

- Feedback on this sneak preview/demo. Is it appealing and useful?
- Need bmv2 progress to run meaningful DASH test cases – define the MVP, close the gaps.
- Who wants to work on gNMI test components?
- After framework developed – let’s write more test-cases!

Q&A, Feedback?

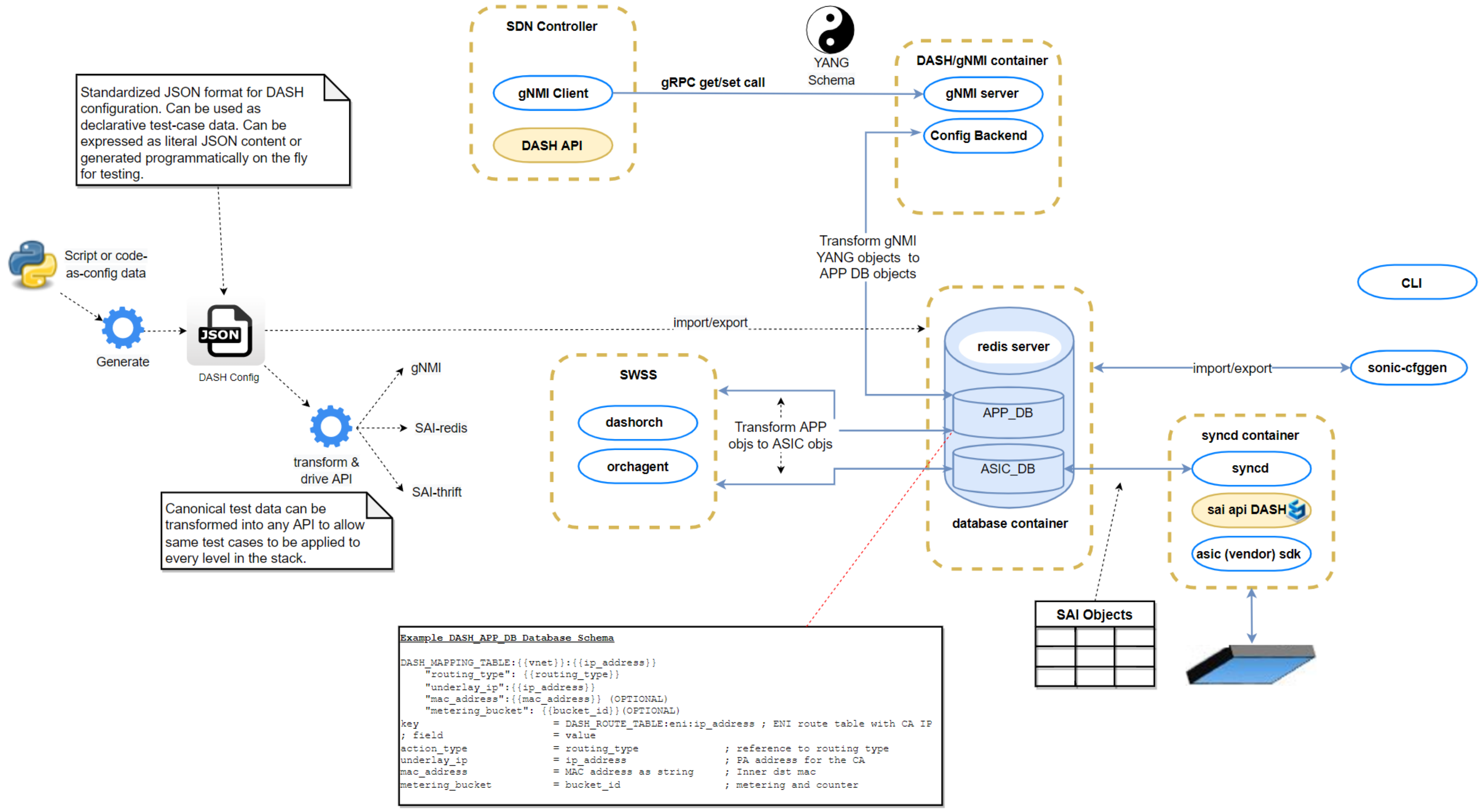
KEYSIGHT

Backup Slides

Test Methodology Evolution

What	**Existing tools**	**…Adding New Tools**
Test framework	Unittest	Pytest
DUT APIs	saithrift	saithrift, sairedis, gNMI (future)
Dataplane	PTF (Scapy based)	PTF and/or **snappi** (open traffic generator)
Coding style	Concrete use of sai_thrift APIs	Abstract, config data-driven; underlying APIs taken care of by framework
Granularity	Direct access to each API and data type, allows arbitrary API usage	Config + helpers hide the API details (but also discourages direct access)
Expertise	Requires intimate knowledge of APIs and data types	Config data easy to understand, API knowledge not required
Packet testing: Speed & Scale	“Packet-at-a-time” testing, speed is limited	Packet-at-a-time or flow based, speed up to full line rate
Config scaling	Ad-hoc coding, limited by ingenuity & Scapy limitations	Built-in handling of large static configs or on-the-fly config generator

Recap – Schema Relationships



Standardized JSON format for DASH configuration. Can be used as declarative test-case data. Can be expressed as literal JSON content or generated programmatically on the fly for testing.

Canonical test data can be transformed into any API to allow same test cases to be applied to every level in the stack.

<u>Example DASH_APP_DB Database Schema</u>

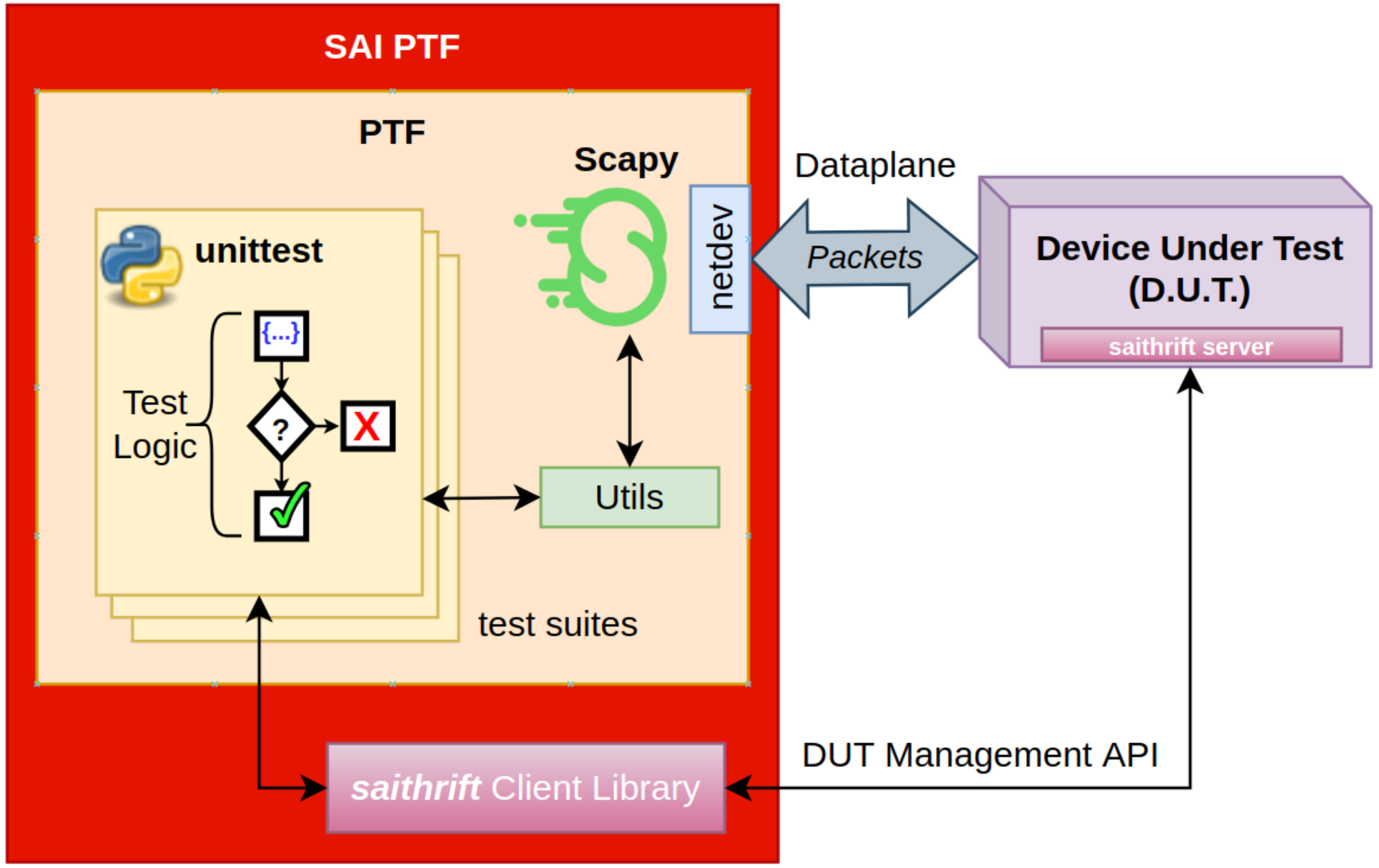
```
DASH_MAPPING_TABLE:{{vnet}}:{{ip_address}}
    "routing_type": {{routing_type}}
    "underlay_ip":{{ip_address}}
    "mac_address":{{mac_address}} (OPTIONAL)
    "metering_bucket": {{bucket_id}}(OPTIONAL)
key                      = DASH_ROUTE_TABLE:eni:ip_address ; ENI route table with CA IP
; field                  = value
action_type              = routing_type             ; reference to routing type
underlay_ip              = ip_address               ; PA address for the CA
mac_address              = MAC address as string    ; Inner dst mac
metering_bucket          = bucket_id                ; metering and counter
```

KEYSIGHT

https://github.com/Azure/DASH/blob/main/documentation/general/design/dash-sonic-hld.md#figure---schema-relationships

Background: PTF/SAI-PTF

- **Unittest** is a Python framework for generic software unit tests. Developers write “test suites” with pass/fail outcomes.
- PTF was created to test dataplanes. It combines the unittest framework with Scapy, a popular software traffic generator/capture tool, plus various utilities to make dataplane tests easy to write. It does not include a DUT configuration API or transport.
- **SAI-PTF** is PTF with an Apache Thrift RPC transport layer plus Python client libraries for SAI configuration.



KEYSIGHT

KEYSIGHT

Thank you