

OPEN POSSIBILITIES.

Hardware Testing at Hyperscale



OCP
GLOBAL
SUMMIT

NOVEMBER 9-10, 2021



TEST AND
VALIDATION

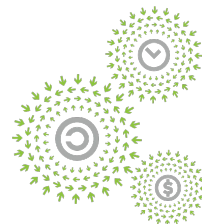
Hardware Testing at Hyperscale

Dan Frame, SWE Manager, Google

Paul Ng, QA Lead, Meta

Vincent Matossian, SWE Manager, Meta

Yuanlin Wen, SWE, Google



OPEN
PLATINUM™

OPEN POSSIBILITIES.



Presenters



TEST AND
VALIDATION



Paul Ng
QA Lead, Meta



Vincent Matossian
Software Engineering
Manager,
Meta



Dan Frame
Software Engineering
Manager,
Google

OPEN POSSIBILITIES.



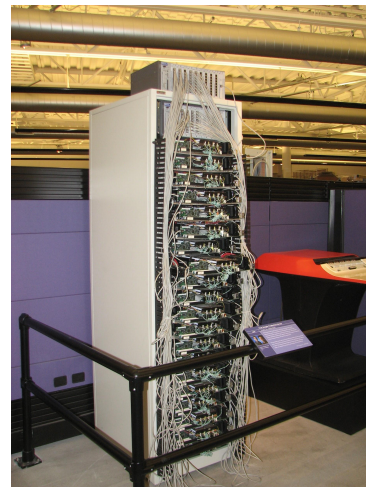
The Early Days of DC Hardware Testing

Before the days of Hyperscale:

- Server Counts were an insignificant fraction of what they are today.
- All machines were basically homogenous
 - It was a CPU centric world, and CPU was generally x86-64 based.
- SKU proliferation was low, and almost everything was “designed in house” for the Hyperscalers specific needs.
- Most of our testing and validation was done at one integration facility with a common set of infrastructure.



TEST AND
VALIDATION



OPEN POSSIBILITIES.



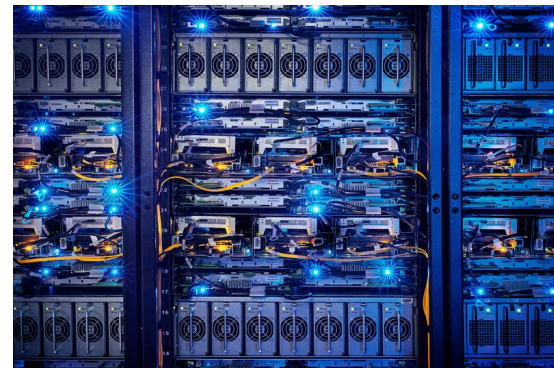
Enter the Hyperscale Era

- Machine counts have well exceeded linear growth.
- The proliferation of different machine types has continued to grow
- Several different instruction sets to target (x64, AArch64, RISC-V, etc)
- No longer only CPU-centric, there are many types of off-loads and accelerators that need testing.
- Increasingly, DC designs are becoming partnerships across many different organizations with different environments
- Tests and Diagnostics are no longer developed 100% internally. We use a variety of different diagnostics both internal and externally developed. Many tests and repair processes are proprietary with documented interfaces.
- The New Product Introduction (NPI) cycle has shortened, and elimination of duplicate work for testing/validation has become essential to be competitive.

OPEN POSSIBILITIES.



TEST AND
VALIDATION



Hardware Diagnostics - Low Volume/Early Life Cycle



TEST AND
VALIDATION

Hardware Bringup

System Integration Testing

Reliability Testing

Why?

First Boot, initial
debug/design verification

Verify Hardware and Software
quality/compatibility during
development

Estimate Hardware Longevity
Estimates and Reliability (MTBF,
MTDL, etc), Thermal Limits and
Design Issues

What?

Power Sequencing, Boot Up,
Bus Training

Hardware
diagnostic/performance/stress/load
testing for software development
life-cycle.

Stress Testing, Voltage/Frequency
Margining
Environmental and Thermal Testing

Who?

Hardware/Software Engineers

Software Engineers

Hardware and Quality/Reliability
Engineers

Where?

Design Partners and Hyperscalers
Lab Bench, Simulators

Design Partners and Hyperscalers
Dedicated CI Environment

Labs, Environmental Chambers,
Shock and Vibe, etc

How?

Manual Execution
No/Light Automation
Ad Hoc Execution

Usually integrated into Continuous
Integration/Continuous Release
Environment and toolchain.

Long Tests
Highly Automated

OPEN POSSIBILITIES.



Hardware Diagnostics - Volume Applications



TEST AND
VALIDATION

Manufacturing

Data Center Operations

RMA/Reverse Logistics

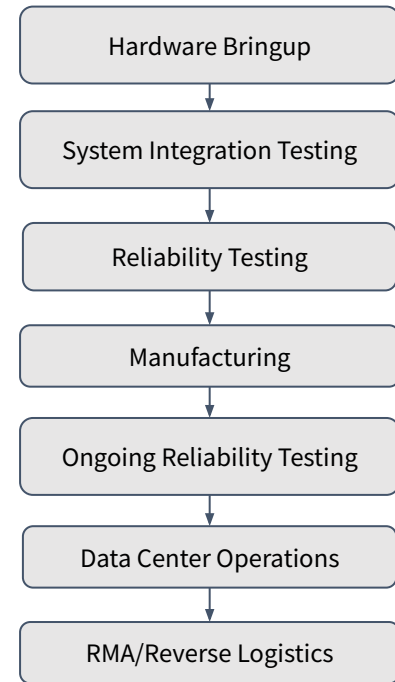
Why?	Verify Components, Provisioning and Assembly Processes	Verify Components, Provisioning and Assembly Processes	Verify Components, Provisioning and Assembly Processes
What?	Test All Components, Interconnects, and Assemblies	Test All Components, Interconnects, and Assemblies	Test Components
Who?	Manufacturing Engineers	Data Center Operations	Hardware/Vendor Engineering
Where?	Contract/Original Design Manufacturers	Data Centers, Colo Facilities	Contract/Original Design Manufacturers
How?	Highly Automated Test Executives with tight shop floor control integration Indict to Component/BUS Level	High Automated Test Executives High Security Requirements Tight integration with Work Flow Management Systems Indict to FRU Level	Various Levels of Automation Indict to Component or FRU Level

OPEN POSSIBILITIES.



Multiple Use-Cases, Multiple Requirements

- Different Execution Environments
 - Many Different Test Executives and Sequencers used for different testing scenarios
 - Different Security Requirements
 - Different Operating Systems
 - Different Data Schemas
- Different Test Use Cases
 - Long-Running vs. Short Running Tests
 - Component level vs. FRU level Root-Cause
 - FRU Level vs. System Level vs. Rack Level vs. Multi-Rack Testing
- Different Users, Engineers, and Stakeholders
 - Differing Skill-sets
 - Differing Preferred Toolsets
 - Development Languages
 - Continuous Integration Environments
 - Data Collection/Analysis Needs



OPEN POSSIBILITIES.



What are the new challenges we need to solve?



TEST AND
VALIDATION

- Acceleration/re-use of diagnostic development and integration efforts at all stages of the product life-cycle.
- Diagnostic portability across multiple products, environments, and use-cases.
- Reproduction of test and validation issues across multiple hardware and software partners.
- Simple sharing of component vendor tests to accelerate RMA and root-cause analysis.

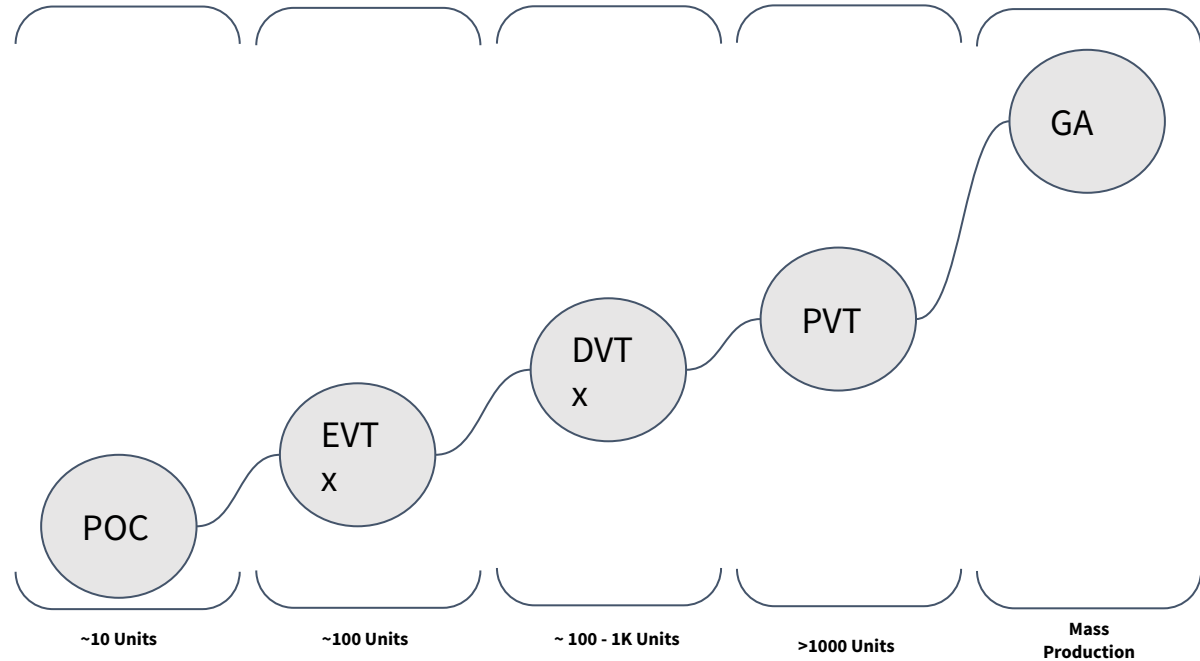
OPEN POSSIBILITIES.



Hardware Testing Applications

Testing requirements continue to change at each stage as volumes continue to increase in the product development life-cycle...

As the DUT counts increase, so does pressure for test time optimization and high fault isolation to aid repair cycles.



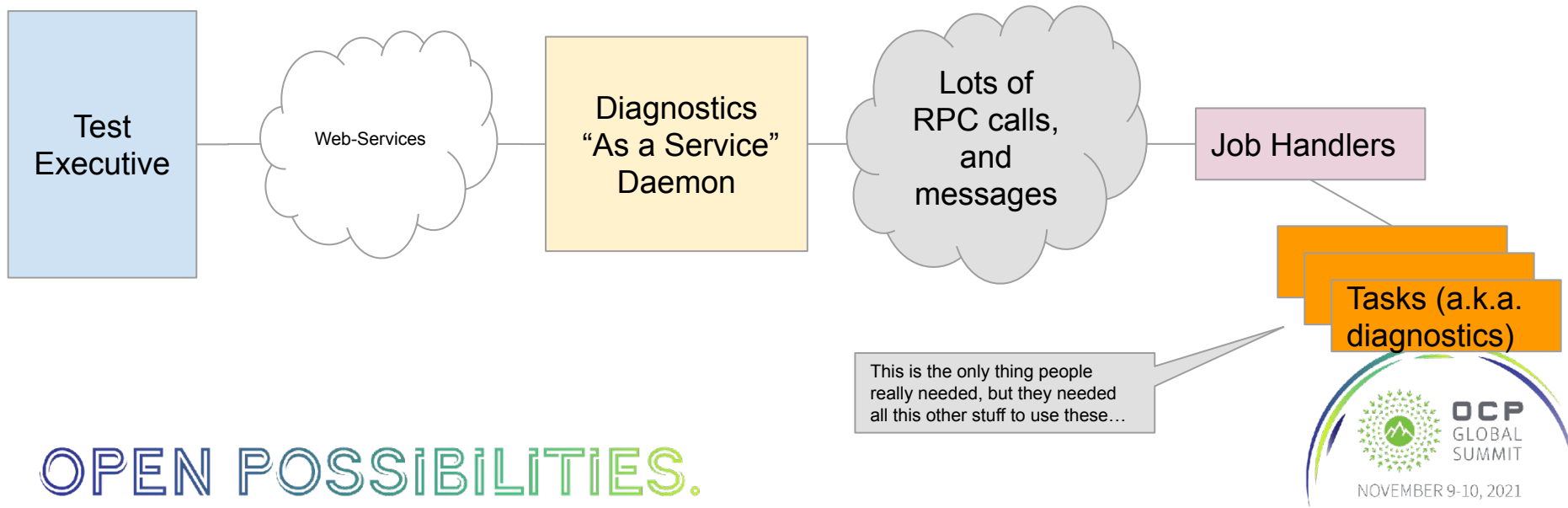
OPEN POSSIBILITIES.

How We Got Here



TEST AND
VALIDATION

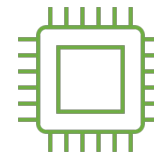
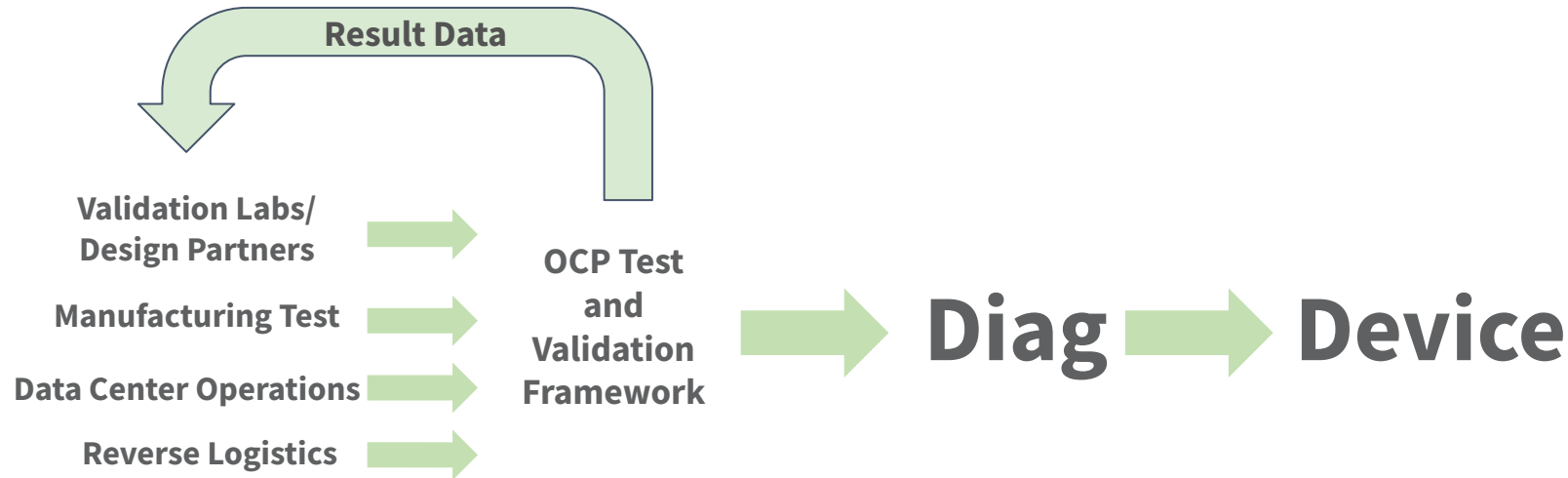
Before the proposed OCP Diagnostic standard, our diagnostics were very tightly integrated with our test framework, and it made portability very difficult. In order to run our diagnostics, it meant exporting a great deal of our internal Infrastructure.



How Test Execution Could Be Structured



TEST AND
VALIDATION



OPEN POSSIBILITIES.



OCP Diagnostic and Validation Framework



TEST AND
VALIDATION

This framework provides multi-language support for the following features...

- Proven Data Model for Diagnostic Output
- API's to easily produce that output.
- Streaming Results For Long Running Tests
- Simple, Powerful Parameter Management
- An optional Device Communication Library
- An optional Hardware Abstraction Layer

OPEN POSSIBILITIES.



How does it fit in different environments?



TEST AND
VALIDATION

Test Environment/Executive

- Provides sequencing for tests.
- Typically integrates with PLM and control systems.
- Records test results to some persistent store (database, etc)
- Provides arguments to a diagnostic
- May control the lifecycle of a diagnostic.
- May be responsible for installing a diagnostic payload onto a machine under test.
- Typically has final determination of pass/fail or at least the ability to override that.
- May transform OCP diagnostic output to an internal/alternative representation.

OCP Diagnostic

- Parses input arguments
- Performs actual testing either on or off the device under test.
- Provides a consistent output format.
- Provides pass/fail result which can be overridden by a test executive.

The OCP Diagnostic framework is NOT a test executive.

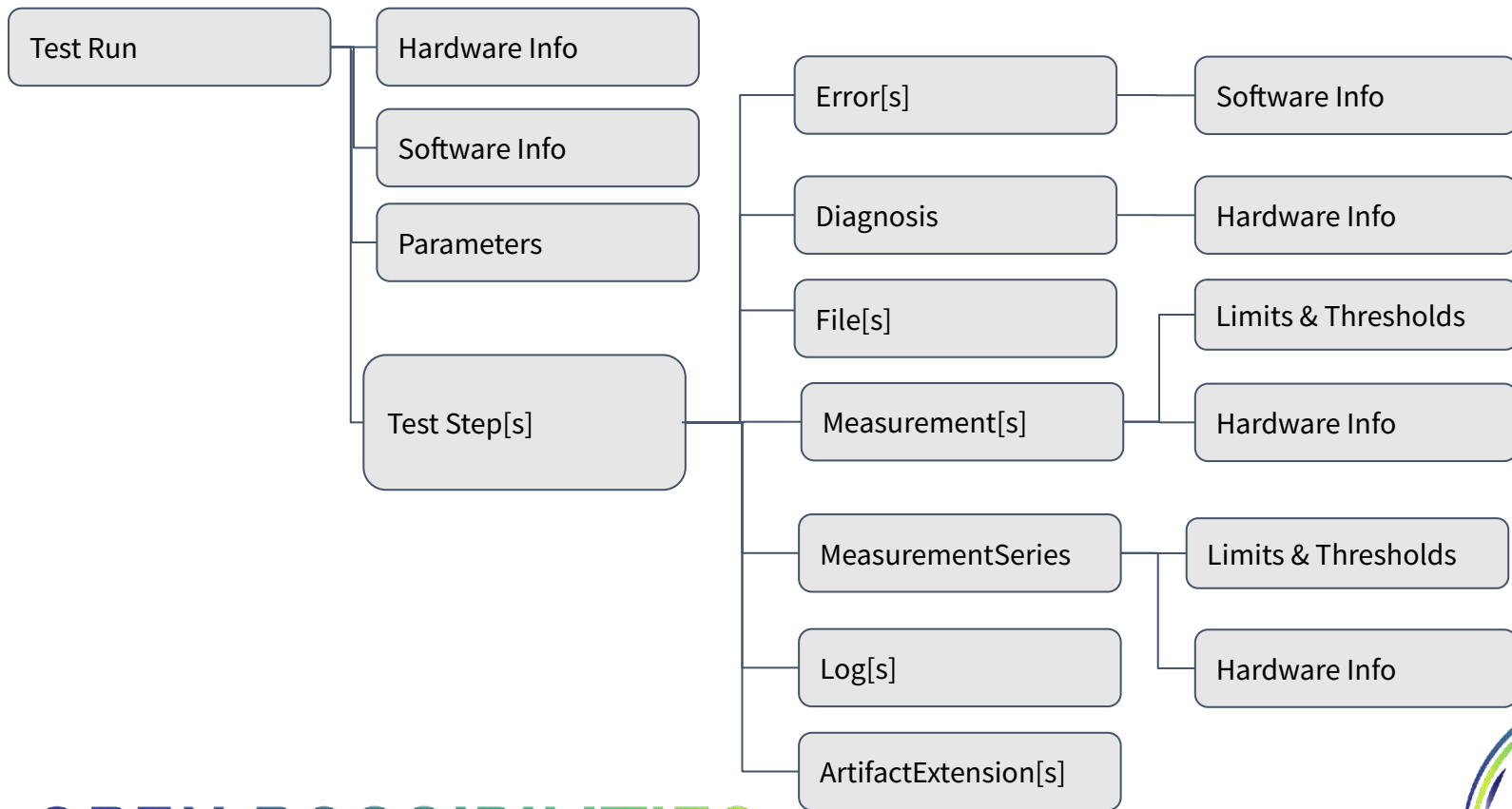
A test executive typically has dozens of integration points in an organization (i.e. ERP, MES, Data Collection, etc).

By contrast, the diagnostic or test typically only has two integration points, so portability is best achieved at interacting at this level.

OPEN POSSIBILITIES.



OCP Diagnostics - Result Model



TEST AND
VALIDATION

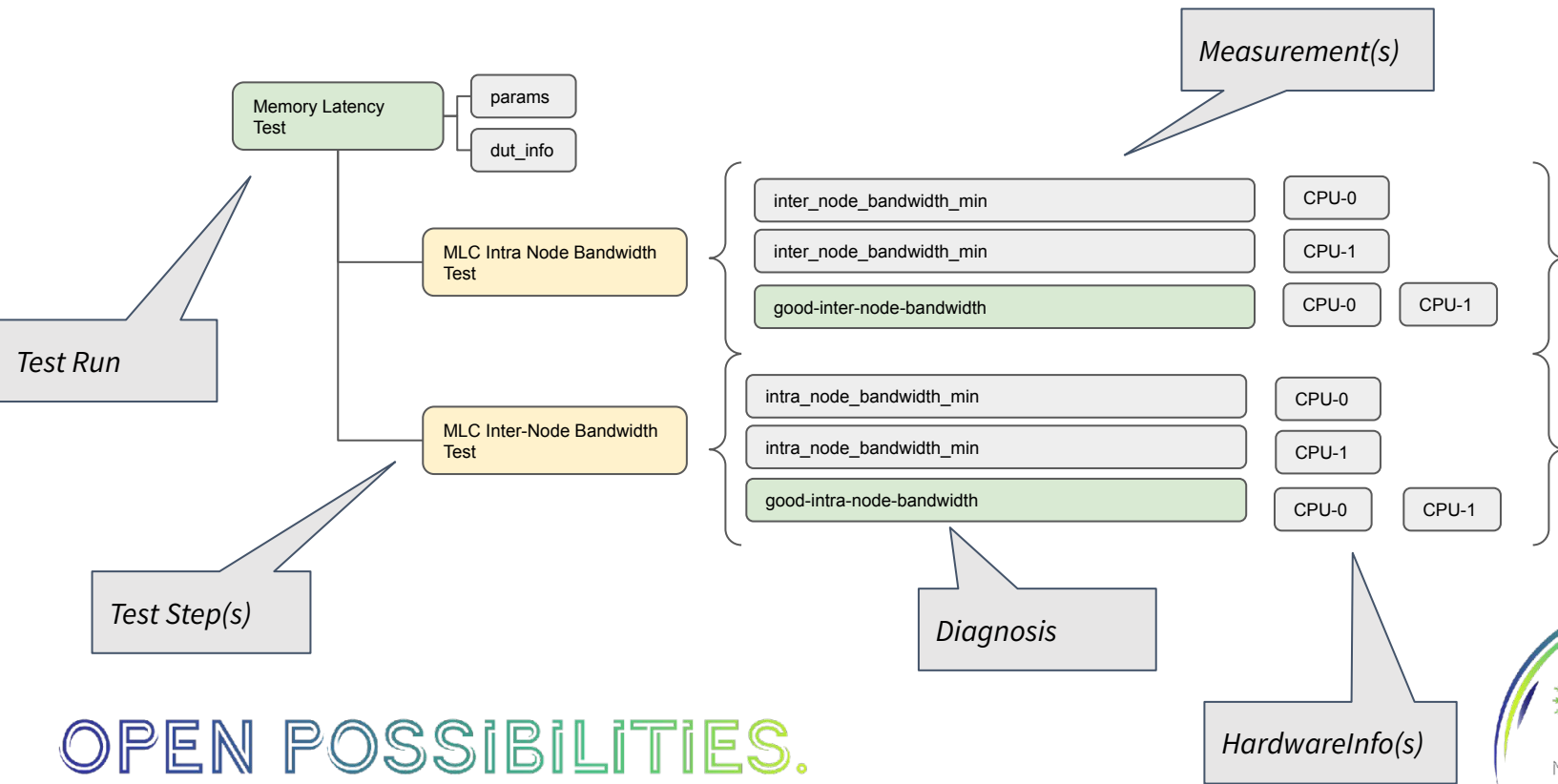
OPEN POSSIBILITIES.



OCP Diagnostics - Result Model Example



TEST AND
VALIDATION



OPEN POSSIBILITIES.



OCP Diagnostics - Result API's - Test Runs



TEST AND
VALIDATION

// Intended use is to have one TestRun object per OcpDiag Test.

```
class TestRun : public internal::LoggerInterface {  
public:  
    ~TestRun() override { End(); }
```

// Returns a TestRun object if successful. This is meant to be called only
// once per test, and will fail if called a second time. `name`: a descriptive
// name for your test.

```
static absl::StatusOr<TestRun> Init(std::string
```

// Emits a TestRunStart artifact and registers the DutInfos.
// No additional DutInfos can be registered after this point.

```
virtual void StartAndRegisterInfos(  
    absl::Span<const DutInfo> dutinfos,  
    const proto2::Message& params = google::protobuf::Empty());
```

// Emits a TestRunEnd artifact and returns overall result.
virtual third_party::OcpDiag::results_pb::TestResult End();

// Skips and ends the Test.

// Should be part of, or followed by a return statement.

```
virtual third_party::OcpDiag::results_pb::TestResult Skip();
```

// Emits an Error artifact, associated with the TestRun.

// This is intended for scenarios where a software error occurs

// before the test officially starts (i.e. the TestRun::StartAndRegisterInfos
// method has not yet been called. For example, when gathering host
// information with the hardware interface).

// Once the test has started, prefer to use TestStep::AddError(...).

```
virtual void AddError(absl::string_view symptom, absl::string_view message);
```

// Emits a Tag artifact, associated with the TestRun

```
virtual void AddTag(absl::string_view tag);
```

// Returns the current overall TestRun status

```
virtual third_party::OcpDiag::results_pb::TestStatus Status() const;
```

// Returns the current overall TestRun result

```
virtual third_party::OcpDiag::results_pb::TestResult Result() const;
```

// If true, it is ok to start creating TestSteps.

```
virtual bool Started() const;
```

// Returns true if the TestRun has ended (i.e. any of End(), Skip(), or
// fatal error have been called)

```
virtual bool Ended() const;
```

// Emits a Log artifact of Debug severity, associated with the TestRun.

```
void Debug(absl::string_view msg) override;
```

// Emits a Log artifact of Info severity, associated with the TestRun.

```
void Info(absl::string_view msg) override;
```

// Emits a Log artifact of Warn severity, associated with the TestRun.

```
void Warn(absl::string_view msg) override;
```

// Emits a Log artifact of Error severity, associated with the TestRun.

```
void Error(absl::string_view msg) override;
```

// Emits a Log artifact of Fatal severity, associated with the TestRun.

// Note: this may have downstream effects, such as terminating the program.

```
void Fatal(absl::string_view msg) override;
```

OPEN POSSIBILITIES.



OCP Diagnostics - Result API's - Test Steps



TEST AND
VALIDATION

```
// TestStep is a logical subdivision of a TestRun.
class TestStep : public internal::LoggerInterface {
public:
    ~TestStep() override { End(); }

    // Factory to create a TestStep. Emits a TestStepStart artifact if successful.
    static absl::StatusOr<TestStep> Begin(TestRun*, std::string name);

    // Emits a Diagnosis artifact. A FAIL type also sets TestRun result to FAIL,
    // unless an Error artifact has been emitted before this.
    virtual void AddDiagnosis(third_party::OcpDiag::results_pb::Diagnosis::Type,
                             std::string symptom, std::string message,
                             absl::Span<const HwRecord>);

    // Emits an Error artifact associated with this TestStep.
    // Also Sets TestRun status to ERROR.
    virtual void AddError(absl::string_view symptom, absl::string_view message,
                          absl::Span<const SwRecord>);

    // Emits a standalone Measurement artifact.
    // Acceptable Value kinds if using ValidValues limit: NullValue, number,
    // string, bool, ListValue.
    // Acceptable Value kinds if using Range limit: number, string.
    virtual void AddMeasurement(
        third_party::OcpDiag::results_pb::MeasurementInfo,
        third_party::OcpDiag::results_pb::MeasurementElement,
        const HwRecord* hwrec);

    // Emits a File artifact
    virtual void AddFile(third_party::OcpDiag::results_pb::File);
```

```
// Emits an ArtifactExtension artifact
virtual void AddArtifactExtension(std::string name,
                                  const proto2::Message& extension);

// Emits a Log artifact of Debug severity, associated with the TestStep.
void Debug(absl::string_view msg) override;
// Emits a Log artifact of Info severity, associated with the TestStep.
void Info(absl::string_view msg) override;
// Emits a Log artifact of Warn severity, associated with the TestStep.
void Warn(absl::string_view msg) override;
// Emits a Log artifact of Error severity, associated with the TestStep.
void Error(absl::string_view msg) override;
// Emits a Log artifact of Fatal severity, associated with the TestStep.
// Note: this may have downstream effects, such as terminating the program.
void Fatal(absl::string_view msg) override;

// Emits a TestStepEnd artifact
virtual void End();

// Skips and ends the step.
virtual void Skip();

// Returns true if End() or Skip() have been called
bool Ended() const;

// Returns current TestStep status
third_party::OcpDiag::results_pb::TestStatus Status() const;
```

OPEN POSSIBILITIES.



OCP Diagnostics - Result API's - MeasurementSeries



TEST AND
VALIDATION

// A collection of related measurement elements.

```
class MeasurementSeries {  
public:  
    virtual ~MeasurementSeries() { End(); }
```

// Factory method to create a MeasurementSeries. Emits a

// MeasurementSeriesStart artifact if successful.

```
static absl::StatusOr<MeasurementSeries> Begin(  
    TestStep*, const HwRecord&,  
    third_party::OcpDiag::results_pb::MeasurementInfo);
```

// Emits a MeasurementElement artifact with valid range limit.

// Acceptable Value kinds: string, number

```
virtual void AddElementWithRange(  
    google::protobuf::Value,  
    third_party::OcpDiag::results_pb::MeasurementElement::Range
```

// Emits a MeasurementElement artifact with valid values limit.

// Acceptable Value kinds: NullValue, number, string, bool, ListValue.

```
virtual void AddElementWithValues(  
    google::protobuf::Value,  
    absl::Span<const google::protobuf::Value> valid_values);
```

// Emits a MeasurementElement artifact without a limit.

// Acceptable Value kinds: NullValue, number, string, bool, ListValue.

```
virtual void AddElement(google::protobuf::Value value);
```

// Emits a MeasurementSeriesEnd artifact unless already ended.

```
virtual void End();
```

// Returns true if End() has already been called

```
virtual bool Ended() const;
```

OPEN POSSIBILITIES.



Diagnostic Output - JSON

The OCP Diagnostic Framework by default returns results as executed as streaming JSON output.



TEST AND
VALIDATION

Why JSON?

- Highly Portable, Self-Describing - No Metadata needed.
- Human readable and machine readable.
- Many visualization/validation tools available
- Widely known/expertise across all diagnostic functions.
- JSONL provides a format for streaming large amounts of JSON for long-running tests that require periodic updates.

Limitations of JSON

- Not Performant/High Level of Transmission Redundancy/Computationally expensive to parse
- Requires an intermediate schema for streaming long-running tests with real-time updates. Some of our use-cases for testing have very long durations (i.e. weeks)

We have selected portability over efficiency for the simplified integration, but internally all data is represented by a strongly typed, efficient protocol buffer implementation.

OPEN POSSIBILITIES.



Diagnostic Output - JSON



TEST AND
VALIDATION

```
{"testRunArtifact":{"testRunStart":{"name":"mlc","version":"399834856","parameters":{"@type":"type.googleapis.com/meltan.mlc.Parameters","interNodeBandwidthMin":0,"intraNodeBandwidthMin":0,"interNodeLatencyMax":0,"intraNodeLatencyMax":0,"useDefaultThresholds":true,"dataCollectionMode":false},"dutInfo":{"hostname":"","hardwareComponents":[{"hardwareInfoId":"0","arena":"","name":"cpu0","fruLocation":{"devpath":"/phys/CPU0","odataId":"","blockpath":"","serialNumber":"cpu0_serial"},"partNumber":"cpu0_part","manufacturer":"MFG","mfgPartNumber":"","partType":"cpu"}],"softwareInfos":[{"softwareInfoId":"1","arena":"","name":"system_daemon","version":"20210902.0-external-nightly-0"}]},"sequenceNumber":0,"timestamp":"2021-09-30T03:09:44.678957932Z"}
{"testStepArtifact":{"testStepStart":{"name":"Measure Internode Bandwidth"},"testStepId":"1","sequenceNumber":1,"timestamp":"2021-09-30T03:12:40.667365379Z"}
{"testStepArtifact":{"measurement":{"info":{"name":"inter_node_bandwidth_min","unit":"MB/sec","hardwareInfoId":"0"},"element":{"index":0,"measurementSeriesId":"NOT_APPLICABLE","range":{"minimum":49500,"maximum":"Infinity"},"value":115649.4},"testStepId":"1","sequenceNumber":2,"timestamp":"2021-09-30T03:12:40.667907305Z"}
{"testStepArtifact":{"measurement":{"info":{"name":"inter_node_bandwidth_min","unit":"MB/sec","hardwareInfoId":"1"},"element":{"index":0,"measurementSeriesId":"NOT_APPLICABLE","range":{"minimum":49500,"maximum":"Infinity"},"value":115704.2},"testStepId":"1","sequenceNumber":3,"timestamp":"2021-09-30T03:12:40.668283952Z"}
{"testStepArtifact":{"diagnosis":{"symptom":"good-inter-node-bandwidth","type":"PASS","msg":"Measured value 115649.4 \u003e= minimum bandwidth threshold 49500","hardwareInfoId":["0","1"],"testStepId":"1","sequenceNumber":4,"timestamp":"2021-09-30T03:12:40.668557351Z"}
{"testStepArtifact":{"testStepEnd":{"name":"Measure Internode Bandwidth","status":"COMPLETE"},"testStepId":"1","sequenceNumber":5,"timestamp":"2021-09-30T03:12:40.668732179Z"}
{"testStepArtifact":{"testStepStart":{"name":"Measure Intranode Bandwidth"},"testStepId":"2","sequenceNumber":6,"timestamp":"2021-09-30T03:12:40.668890997Z"}
{"testStepArtifact":{"measurement":{"info":{"name":"intra_node_bandwidth_min","unit":"MB/sec","hardwareInfoId":"0"},"element":{"index":0,"measurementSeriesId":"NOT_APPLICABLE","range":{"minimum":139500,"maximum":"Infinity"},"value":180296.1},"testStepId":"2","sequenceNumber":7,"timestamp":"2021-09-30T03:12:40.669171538Z"}
{"testStepArtifact":{"measurement":{"info":{"name":"intra_node_bandwidth_min","unit":"MB/sec","hardwareInfoId":"1"},"element":{"index":0,"measurementSeriesId":"NOT_APPLICABLE","range":{"minimum":139500,"maximum":"Infinity"},"value":180585.5},"testStepId":"2","sequenceNumber":8,"timestamp":"2021-09-30T03:12:40.669462376Z"}
{"testStepArtifact":{"diagnosis":{"symptom":"good-intra-node-bandwidth","type":"PASS","msg":"Measured value 180296.1 \u003e= minimum bandwidth threshold 139500","hardwareInfoId":["0","1"],"testStepId":"2","sequenceNumber":9,"timestamp":"2021-09-30T03:12:40.669685368Z"}
{"testStepArtifact":{"testStepEnd":{"name":"Measure Intranode Bandwidth","status":"COMPLETE"},"testStepId":"2","sequenceNumber":10,"timestamp":"2021-09-30T03:12:40.669851968Z"}
{"testRunArtifact":{"testRunEnd":{"name":"mlc","status":"COMPLETE","result":"FAIL"},"sequenceNumber":11,"timestamp":"2021-09-30T03:12:40.672711573Z"}}
```

Test Run Start

Test Step Start

Measurement

Measurement

Diagnosis

Test Step End

Test Step Start

Measurement

Measurement

Diagnosis

Test Step End

Test Run End

OPEN POSSIBILITIES.



OCP Diagnostic Parameter Model



TEST AND
VALIDATION

Due to the requirements to re-use diagnostics in multiple use-cases and environments, the ability to parameterize and configure the diagnostics at execution time rather than build time is essential.

In addition, some diagnostics have many different parameters, including complex-types and lists of values.

As a result, the ability to provide simple help to the consumers of the diagnostics, default parameters, and the ability to override those default parameters necessitates a powerful parameter model that allows developers to focus on the test challenge at hand, rather than the plumbing required to capture parameters and integrate with other test environments.

Parameters to OCP diagnostics can be specified as CLI arguments, or supplied via StdIn depending on the best approach for different users. This also provides the ability to leverage configuration files for very large parameter sets that are infrequently changing.

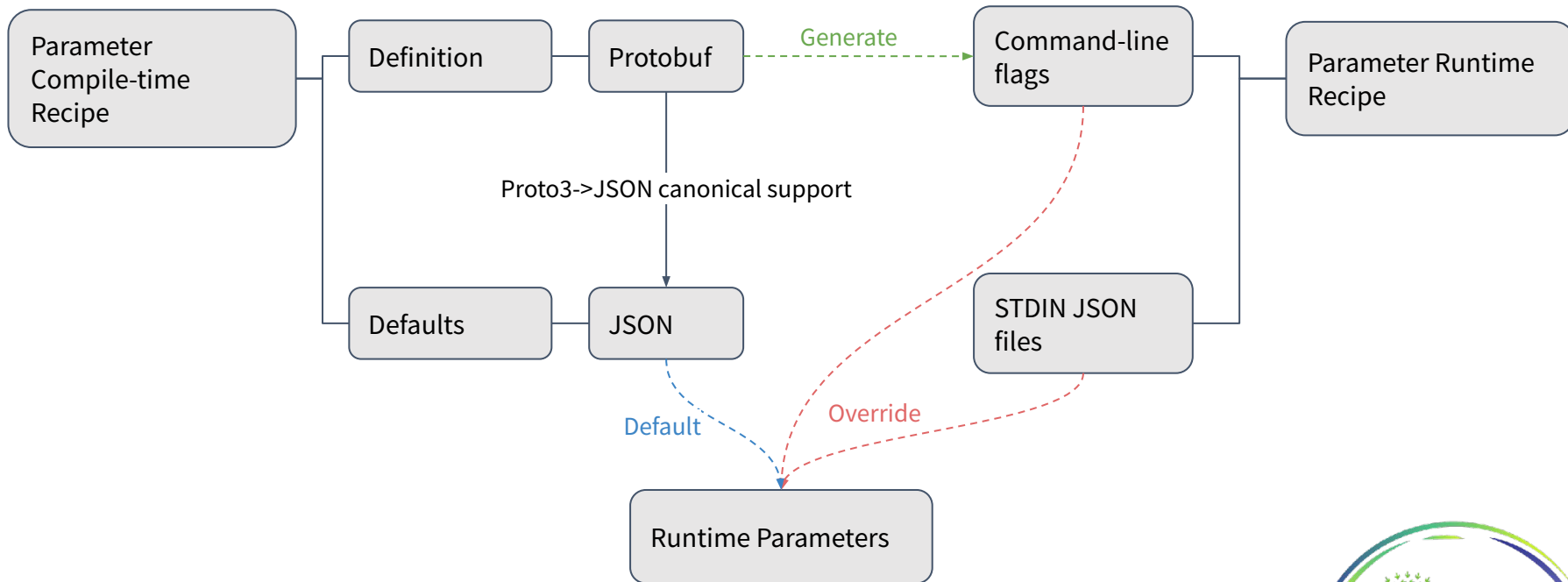
OPEN POSSIBILITIES.



OCP Diagnostic Parameter Model



TEST AND
VALIDATION



OPEN POSSIBILITIES.

Parameter Definition & Defaults

--help can be used to print parameter flags.

\$./mlc --help

Usage: ./mlc [options]	Name	Description
	<code>--inter_node_bandwidth_min</code>	Minimum inter-node bandwidth required. Type: float Default: 0
	<code>--intra_node_bandwidth_min</code>	Minimum intra-node bandwidth required. Type: float Default: 0
	<code>--inter_node_latency_max</code>	Maximum inter-node bandwidth allowed. Type: float Default: 0
	<code>--intra_node_latency_max</code>	Maximum intra-node bandwidth allowed. Type: float Default: 0
	<code>--use_default_thresholds</code>	Whether to use default thresholds Type: bool Default: true
	<code>--data_collection_mode</code>	If this is true, the test won't compare the bandwidth or data with any thresholds. Type: bool Default: false

// File: mlc/params.proto

syntax = "proto3";

package OcpDiag.mlc;

message Params {

// Minimum inter-node bandwidth required.

float inter_node_bandwidth_min = 1;

// Minimum intra-node bandwidth required.

float intra_node_bandwidth_min = 2;

// Maximum inter-node latency allowed.

float inter_node_latency_max = 3;

// Maximum intra-node latency allowed.

float intra_node_latency_max = 4;

// Whether to use default thresholds.

bool use_default_thresholds = 5;

// If this is true, the test won't compare the bandwidth or data with any thresholds.

bool data_collection_mode = 7;

}

File: mlc/params.json

{

"use_default_thresholds" : true,

"data_collection_mode" : false,

}

OPEN POSSIBILITIES.

"ocpdia_test_pkg" Bazel Build Rule



TEST AND
VALIDATION

```
# mlc/BUILD
```

```
load("//third_party/OcpDiag/lib:OcpDiag.bzl",  
      "ocpdia_test_pkg")
```

```
# Parameter definition.
```

```
proto_library(  
    name = "params_proto",  
    srcs = ["params.proto"],  
)
```

```
cc_proto_library(  
    name = "params_cc_proto",  
    deps = [":params_proto"],  
)
```

```
# Test binary.
```

```
cc_binary(  
    name = "mlc_bin",  
    srcs = ["mlc_main.cc"],  
    deps = [  
        ":params_cc_proto",  
    ],  
)
```

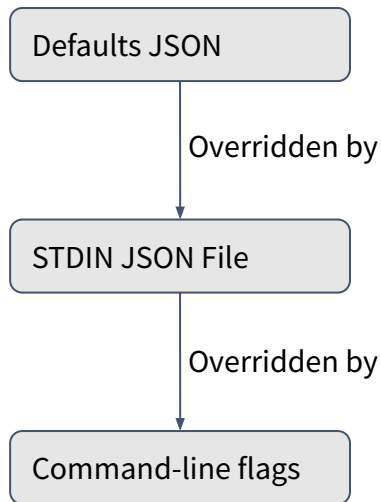
```
# Test executable
```

```
ocpdia_test_pkg(  
    name = "mlc",  
    binary = ":mlc_bin",  
    json_defaults = "params.json",  
    params_proto = ":params_proto",  
)
```

OPEN POSSIBILITIES.



Parameter Overrides



Note: "**--dry_run**" flag can be used to sanity check parameter override combinations.

Parameter override

```
$ ./mlc --dry_run
```

```
{  
  "use_default_thresholds" : true,  
  "data_collection_mode" : false,  
}
```

```
$ cat param_override.json
```

```
{  
  "use_default_thresholds" : false,  
  "inter_node_bandwidth_min" : 100,  
}
```

```
$ ./mlc --dry_run < param_override.json
```

```
{  
  "inter_node_bandwidth_min" : 100,  
  "use_default_thresholds" : false,  
  "data_collection_mode" : false,  
}
```

```
$ ./mlc --dry_run < param_override.json
```

```
--inter_node_bandwidth_min=200
```

```
{  
  "inter_node_bandwidth_min" : 200,  
  "use_default_thresholds" : false,  
  "data_collection_mode" : false,  
}
```

OPEN POSSIBILITIES.

OCP Diagnostics - Communication Interface



TEST AND
VALIDATION

Diagnostics are typically invoked and sequenced from a control computer that is separate from the device under test. This control computer may be testing dozens, or even hundreds of DUT's in parallel depending on the environment. Different environments have different security needs. For instance, a manufacturing test environment may have different policies for remote execution than a tightly controlled production environment in a data center.

As such, the OCP Diagnostic framework includes an API to assist with common tasks that includes a simple interface for extending it into new environments, with new requirements. By default, an SSH based implementation is provided for users as part of the core framework.

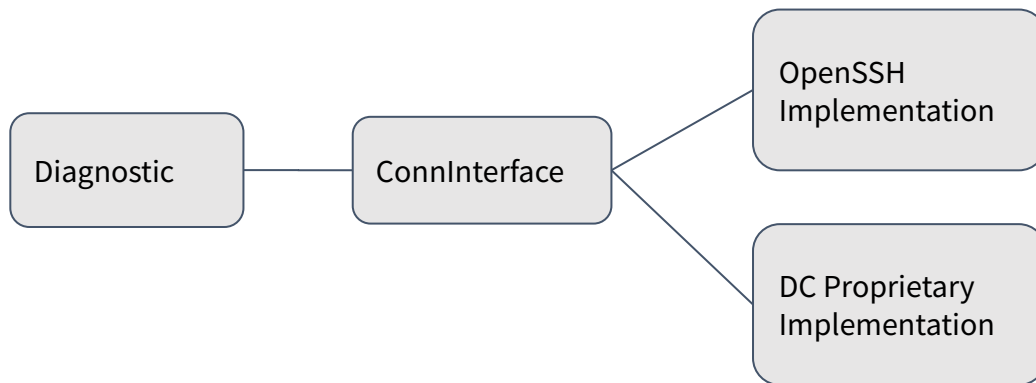
OPEN POSSIBILITIES.



OCP Diagnostics - Communication Interface



TEST AND
VALIDATION



OPEN POSSIBILITIES.



OCP Diagnostics - Communication Interface APIs

```
// Class ConnInterface provides a remote connection to the specified machine
// node. It provides the file read/write operations, and the capability to
// launch a remote command on the machine node.
```

```
class ConnInterface {
public:
```

```
    // Options to configure a command.
```

```
    // The following arguments specify an absolute file path for redirecting
    // stdout/stderr. Whenever the stdout/stderr is redirected, the
    // corresponding field in "CommandResult" will be empty.
    std::string stdout_file;
    std::string stderr_file;
```

```
};
```

```
// The exit code and the command's output to stdout and stderr.
```

```
struct CommandResult {
```

```
    // set to -127 by default.
```

```
    // exit_code = 0 means OK. follows the python-style exit codes.
```

```
    int exit_code = -127;
```

```
    std::string stdout;
```

```
    std::string stderr;
```

```
};
```

```
// ReadFile reads a file from the machine node, and returns the full file
// content on success, or the error status when applicable.
```

```
virtual absl::StatusOr<absl::Cord> ReadFile(absl::string_view file_name) =
0;
```

```
// WriteFile writes the given data to the file on the machine node and
returns
```

```
// the status.
```

```
virtual absl::Status WriteFile(absl::string_view file_name,
                                absl::string_view data) = 0;
```

```
// RunCommand runs a remote command on the machine node, and
returns the
```

```
// command output on success, or the error status when applicable.
```

```
// If the command's stdout/stderr is redirected by setting the
```

```
CommandOption
```

```
// option, the corresponding field in "CommandResult" will be empty.
```

```
virtual absl::StatusOr<CommandResult> RunCommand(
    absl::Duration timeout, const absl::Span<absl::string_view> args,
    const CommandOption& options) = 0;
```

```
};
```

OPEN POSSIBILITIES.



OCP Diagnostics - Hardware Interface



TEST AND
VALIDATION

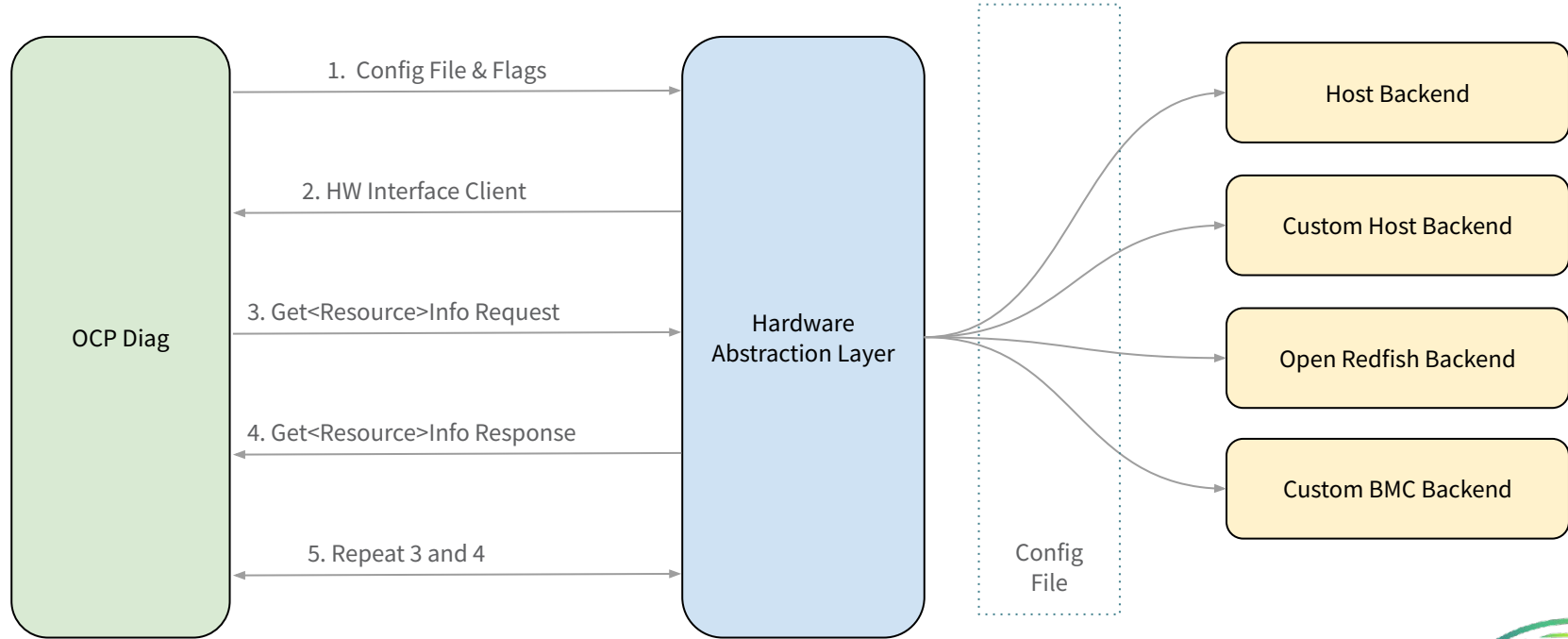
In some scenarios, the way that a diagnostic interrogates DUT hardware may not be consistent in different environments. This can be due to the execution environment of a diagnostic, or may be due to the need for a diagnostic to reference a unique hardware identifier to interface with shop-floor control systems or workflow systems for operations.

As a result, we include an optional HW interface that provides a communication abstraction layer for a device under test. In many cases, this may not be necessary and the diagnostic can communicate to the hardware directly, but in other scenarios, the use of a shim can be beneficial.

OPEN POSSIBILITIES.

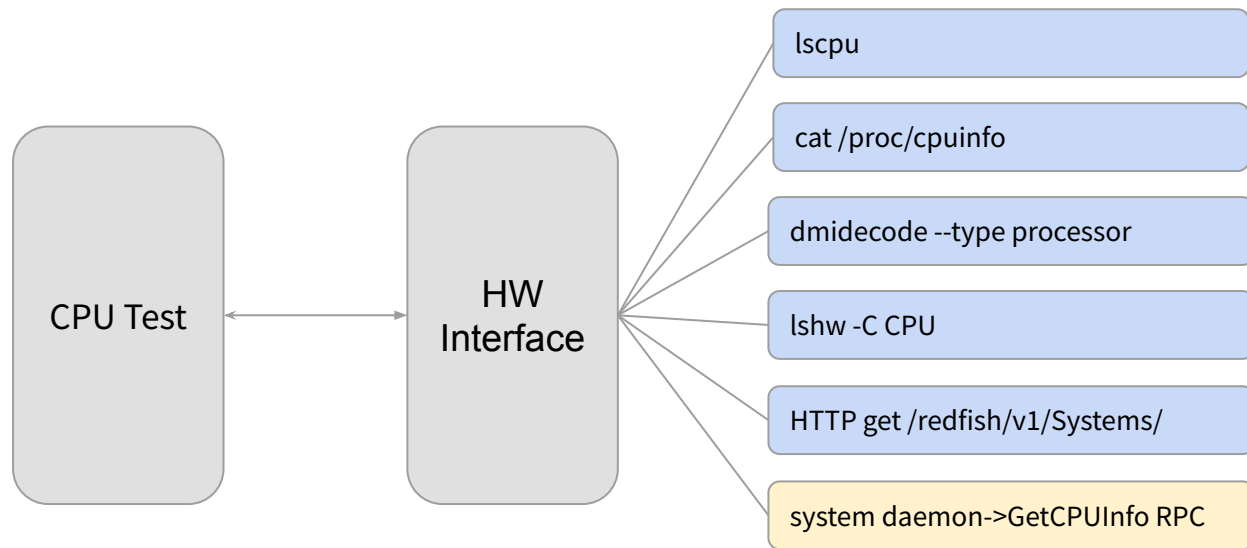


OCP Diagnostics - Hardware Interface



OPEN POSSIBILITIES.

OCP Diagnostics - Hardware Interface



**Different
Implementations**

Allows a single diagnostic to run in multiple OS's or different machine types cleanly.

Allows us to use a different interface between MFG and Production if required.

Provides a transition path to migrate from from proprietary interfaces to open OPC/DTMF standards (i.e. RedFish)

OPEN POSSIBILITIES.

OCP Diagnostics - Multiple Language Support



TEST AND
VALIDATION

The OCP Diagnostic Framework supports diagnostic development with common API's across these languages which are popular in the test development space

- Python
- C++
- Golang (Coming Soon)

OPEN POSSIBILITIES.



OCP Test and Validation Repository



TEST AND
VALIDATION

- JSON format example for implementation.
- Consists of tests that are OCP ready written by the community.
- Community driven tests that can be picked up and dropped into any test executive supporting the OCP diagnostic and validation framework format.

OPEN POSSIBILITIES.



Supported Platforms



TEST AND
VALIDATION

Open Source:

- OpenTAP Test Automation Project
- OpenTest Manufacturing Test Platform
- ConTest Test Automation Framework

Proprietary:

- Google's Burnin Data Center Test Platform
- Meta's FAVA Hardware Test Platform

Many more coming soon!

OPEN POSSIBILITIES.



OPEN POSSIBILITIES.

```

: ">00%" MouseLeft

```



NOVEMBER 9-10, 2021



Assets

BOM

Jobs

TestSuites

Analytics

23:42

ShopFloor

System

Server Info - 01-005218 III-db

Server Serial	Rack Serial	State	Last Seen	Position	Sub-position	Slot	Sled
TWLYX02800860	47962XX	idle	2021-09-29 23:11:48	A19	2	4	QTW

Eth0 IP4	OOB IP4	Eth0 IP6	OOB IP6	Eth0 MAC	OOB MAC	Debug
None	None	FE80::E42:A1FF:FEAD:67F0%enp94s0	FE80::E42:A1FF:FEAD:67EB%enp94s0	0C:42:A1:AD:67:F0	0C:42:A1:AD:67:EB	

System Configuration

Firmware Configuration

Auxiliary Configuration

Job Info UTC

Job Id	Task Id	Name	Start	End	Duration	Reruns	Status	Job	Status	Suite
7475	123370	OcpPcDiag	2021-09-29 22:58:04	2021-09-29 22:58:20	0:00:16	0	PASS	7475	PASS	ocpDiag
								7474	PASS	ocpDiag
								7473	PASS	ocpTestsuite
								7472	PASS	ocpTestsuite
								7471	PASS	ocpTestsuite

Index of /logs/TWLYX02800860/J7475T123370/

./		
favatest.log	29-Sep-2021 22:58	2259
favatest.log.DEBUG	29-Sep-2021 22:58	71707
results.json	29-Sep-2021 22:58	472
result.txt	29-Sep-2021 22:58	254182

FAVA by Meta

```

2021-09-29 22:58:05,059 INFO *****
2021-09-29 22:58:05,059 INFO * FAVA version: 1.12.8rc1-218786-124752-e92c280b181f
2021-09-29 22:58:05,059 INFO * Test name: OcpPcDiag
2021-09-29 22:58:05,059 INFO * Test Params: {'script': 'error_monitor --uid= --envelope_enabled=false --svelte_enable_envelope=false --hinterface_default_backend=host'}
2021-09-29 22:58:05,059 INFO * Controller: qer16
2021-09-29 22:58:05,059 INFO * Job ID: 7475
2021-09-29 22:58:05,059 INFO * Task ID: 123370
2021-09-29 22:58:05,059 INFO * Log dir: /shared/fava/logs/TWLYX02800860/J7475T123370/
2021-09-29 22:58:05,060 INFO * System serial: TWLYX02800860
2021-09-29 22:58:05,060 INFO * Worker serial: TWLYX02800860
2021-09-29 22:58:05,060 INFO * System part number: 01-005218
2021-09-29 22:58:05,060 INFO * Rack serial: 47962XX
2021-09-29 22:58:05,060 INFO * Rack part number: 15-0052XX
2021-09-29 22:58:05,060 INFO * Rack location: 001
2021-09-29 22:58:05,060 INFO * Rack type: N/A
2021-09-29 22:58:05,060 INFO * Testsuite: ocpDiag
2021-09-29 22:58:05,060 INFO * Steps: 02505
2021-09-29 22:58:05,060 INFO * Nodes: [{'TWLYX02800860', 'FE80::E42:A1FF:FEAD:67F0meth0', None}, {'FE80::E42:A1FF:FEAD:67EBmeth0', None}]
2021-09-29 22:58:05,065 INFO * OOB IPv4: FE80::E42:A1FF:FEAD:67EBmeth0 OOB IPv4: None
2021-09-29 22:58:05,065 INFO *****
2021-09-29 22:58:05,070 INFO Running OcpPcDiagTest
2021-09-29 22:58:05,072 INFO Running command: error_monitor --uid= --envelope_enabled=false --svelte_enable_envelope=false --hinterface_default_backend=host
2021-09-29 22:58:05,083 INFO Setup.
2021-09-29 22:58:06,115 INFO Starting test: 'Error Monitor'
2021-09-29 22:58:06,121 INFO Starting error monitoring.
2021-09-29 22:58:19,463 INFO No AER errors found for link with endpoint 0000:67:00:0
2021-09-29 22:58:19,520 INFO No AER errors found for link with endpoint 0000:64:00:0
2021-09-29 22:58:19,533 INFO Stopped error Monitoring.
2021-09-29 22:58:19,536 INFO Result: PASS
2021-09-29 22:58:19,539 INFO Duration: 0:00:14.563515

```

Reference our “lightning talk”,
or visit the OCP experience
center for more information.



OpenTest

OpenTest Start All Life Stations About All Running Stations Version **V3.2.5** Plugins Anonymous User

View All Process Stations

Action	Station	Status	Serial Number	Process Step	Elapsed Time	% Complete	Yield Statistics	Details
▶ Start	Test Station 1	Passed-DUT-1			0:00:01	100%	100.00% 1 1 0 0	
▶ Start	Test Station 2	Passed-DUT-2			0:00:02	0	100.00% 1 1 0 0	
■ Abort	Test Station 3	Running-DUT-3		OcpPcdiag	0:00:00	0	0.00% 0 0 0 0	
▶ Start	Test Station 4	Idle			0:00:00	100%	0.00% 0 0 0 0	

OpenTest Stations Station Debugger Version **V3.2.5** 100.00% 1 1 0 0

Serial Number Start Station

Process Plan OCP Diagnostic Demo Version 1.0.0 Time Elapsed: 00:00:01 % Completed: 100%

PASSED: DUT-1

User Instructions Previous Step Next Step

Default Instructions

Process Plan Results Process Symptoms Measurements Data Records Attachments Debug Output

Process Step	Step Name	Name	Description	Size
1.	OcpPcdiag	OcpPcdiag_test_results_RAW_usdp2of5.txt	Raw JSON Test Results of OcpPcdiag	154108

OpenTest Stations Station Debugger Version **V3.2.5** 100.00% 1 1 0 0

Serial Number Start Station

Process Plan OCP Diagnostic Demo Version 1.0.0 Time Elapsed: 00:00:01 % Completed: 100%

PASSED: DUT-1 Previous Step Next Step

User Instructions Default Instructions

Process Plan Results Process Symptoms Measurements Data Records Attachments Debug Output

Process Step	Step Name	Label	Device Type	Device ID	GPN	MPN	Location	Sublocation
1.1.	OcpPcdiag - monitor-link-0000:07:00:0	healthy-pcie-link	endpoint					3
1.1.	OcpPcdiag - monitor-link-0000:07:00:0	healthy-pcie-link	root_port					3
1.2.	OcpPcdiag - monitor-link-0000:04:00:0	healthy-pcie-link	endpoint					1
1.2.	OcpPcdiag - monitor-link-0000:04:00:0	healthy-pcie-link	root_port					1

Reference our “lightning talk”, or visit the OCP experience center for more information.



OpenTAP

The screenshot shows the OpenTAP Test Automation interface. The top menu bar includes File, Settings, Tools, View, and Help. The main window is divided into several panes:

- Test Steps:** A list of steps including Delay, Dialog, Log Output, Run Program, SCPI, Time Guard, Flow Control, OpenTap Plugins OCP, OCPStep, Operator Prompt, and Operator Prompt Dialog. Each step has an 'Add' button and an 'Add Child' button.
- Test Plan:** A tree view showing the test plan structure. The selected step is 'Load OCP Plan'.
- Test Step Settings:** A pane for configuring the selected test step.
- Log:** A pane showing the execution log, including errors, warnings, and information.

The bottom status bar shows 'OpenTap http://opentap.io'.

The screenshot shows the OpenTAP Test Automation interface after a test plan execution. The top menu bar includes File, Settings, Tools, View, and Help. The main window is divided into several panes:

- Test Steps:** A list of steps including Delay, Dialog, Log Output, Run Program, SCPI, Time Guard, Flow Control, OpenTap Plugins OCP, OCPStep, Operator Prompt, and Operator Prompt Dialog. Each step has an 'Add' button and an 'Add Child' button.
- Test Plan:** A tree view showing the test plan structure. The selected step is 'Error Monitor'.
- Test Step Settings:** A pane for configuring the selected test step.
- Log:** A pane showing the execution log, including errors, warnings, and information.

The bottom status bar shows 'OpenTap http://opentap.io'.

The log output shows the following summary:

```

21:02:47:095 TestPlan Test step menu finished. (1.16 ms)
21:02:47:095 Summary ..... Summary of test plan started 18/12/2021 21:02:47:
21:02:47:095 Summary Error Monitor 1.00 ms Pass
21:02:47:095 Summary monitor-link-0000-6700-0 418 us
21:02:47:095 Summary monitor-link-0000-64-00-0 409 us
21:02:47:095 Summary ..... Test plan completed successfully in 42.9 ms
21:02:47:138 CSV Resource "CSV" closed. [31.8 us]
  
```

The word "Passed" is displayed in green text.

The screenshot shows the AutoSave application interface. The top menu bar includes File, Home, Insert, Page Layout, Formulas, Data, Review, View, Help, and Team. The main window is divided into several panes:

- File:** A pane showing the file structure.
- Home:** A pane showing the home view.
- Insert:** A pane showing the insert view.
- Page Layout:** A pane showing the page layout view.
- Formulas:** A pane showing the formulas view.
- Data:** A pane showing the data view.
- Review:** A pane showing the review view.
- View:** A pane showing the view view.
- Help:** A pane showing the help view.
- Team:** A pane showing the team view.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Name	StepDuration	PlanName	ResultType	0	1	2	3	4	5	6	7	8	9	Step 1	Measurement	Series ID						
2	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	monitor-link-0000-6700-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	monitor-link-0000-64-00-0	0.000419	Untitled	OCF Test V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reference our “lightning talk”, or visit the OCP experience center for more information.



OPEN POSSIBILITIES.

Test Executive Support



TEST AND
VALIDATION

The test platforms we just highlighted are executing the same diagnostics via different communication methods, running on 3 different operating systems.

By implementing your diagnostic in the OCP framework, it's capable of running at:

- Hardware Validation Labs
- Original Design and Contract Manufacturing Partners
- Data Center Testing Systems at Major Hyperscalers

All of this requires no additional integration work, or specialized wrappers for each diagnostic.

If you add support for the OCP Diagnostic format to your test execution platform, you open up executing all OCP diagnostics with a single development effort.

OPEN POSSIBILITIES.



Where to Get it?

The latest version of the OCP Diagnostic Framework and documentation is available publicly at:

git clone <https://github.com/opencomputeproject/ocp-diag-core>



TEST AND
VALIDATION

OPEN POSSIBILITIES.



What's Next?

Over the coming months, we will be releasing many diagnostics based on this format focused on testing non-differentiated core server hardware including:

- Memory
- CPUs
- Storage
- Common Communication Buses
- Machine Check Error Monitoring
- Networking Interfaces
- Environmental/Thermal Monitors
- Power/Performance/Benchmark Monitors

We also will be including common interfaces for industry test executive's such as Keysight's OpenTAP test executive framework and other common open-source unit testing frameworks.

OPEN POSSIBILITIES.



Thanks!



TEST AND
VALIDATION

Special thanks to all the people who have participated in the project so far!

- Raveej Sharma – OCP
- Yuanlin Wen - Google
- Dharmesh Jani – Meta
- Daniel Alvarez Wise – Meta
- Tobias Fleig – Meta
- Adrian Enache – Meta
- Giovanni Colapinto - Meta
- Ron Minich – Google
- Ryan O’Leary – Google
- Kevin Byod – Google
- Brennen DiRenzo – Keysight
- Winston Liu - Keysight
- Jon Stroud - Keysight
- Alexander Wang – Keysight
- Christian Walters – 9Elements
- Jens Drehaus – 9Elements
- Jean-Marie Verdun – HPE
- Arun Koshy – HPE
- Gregg Shick – HPE
- Paula Kylas – HPE
- William Navas - HPE

OPEN POSSIBILITIES.



Call to Action



TEST AND
VALIDATION

- If you are interested, and would like to participate, please join the Test and Validation working group.
- We are looking feedback, diagnostic contributions, as well as re-usable interfaces to common test executives used at ODM's, Hyperscalers, and contract manufacturers
- Check us out at the Experience Center!

Where to participate: <https://github.com/opencomputeproject/ocp-diag-core>

Wiki with latest specification: <https://github.com/opencomputeproject/ocp-diag-core/wiki>

Project Wiki : https://www.opencompute.org/wiki/OCF_Test_and_Validation_Enablement_Initiative

OPEN POSSIBILITIES.



Thank you!



NOVEMBER 9-10, 2021