NLP Based Auto Test Suite Generator (ATSG) for NVMe Protocol

Presenter(s):

Karthik Balan

Associate Director,

Samsung Semiconductor India Research

Clinton A Beetham

Senior Staff Engineer,

Samsung Semiconductor India Research

Sai Shashank Y

Senior Staff Engineer,

Samsung Semiconductor India Research

Disclaimer

This presentation and/or accompanying oral statements by Samsung representatives collectively, the "Presentation" is intended to provide information concerning the SSD and memory industry and Samsung Electronics Co., Ltd. and certain affiliates (collectively, "Samsung").While Samsung strives to provide information that is accurate and up-to-date, this Presentation may nonetheless contain inaccuracies or omissions. As a consequence, Samsung does not in any way guarantee the accuracy or completeness of the information provided in this Presentation.

This Presentation may include forward-looking statements, including, but not limited to, statements about any matter that is not a historical fact; statements regarding Samsung's intentions, beliefs or current expectations concerning, among other things, market prospects, technological developments, growth, strategies, and the industry in which Samsung operates; and statements regarding products or features that are still in development. By their nature, forward-looking statements involve risks and uncertainties, because they relate to events and depend on circumstances that may or may not occur in the future. Samsung cautions you that forward looking statements are not guarantees of future performance and that the actual developments of Samsung, the market, or industry in which Samsung operates may differ materially from those made or suggested by the forward-looking statements in this Presentation. In addition, even if such forward-looking statements are shown to be accurate, those developments may not be indicative of developments in future periods.

Outline

- Background
- Objective
- Functional Flow
- Test Policy
- AI Processing & Binding Layer
- Test Script Generator
- Test Package
- Conclusion

Background

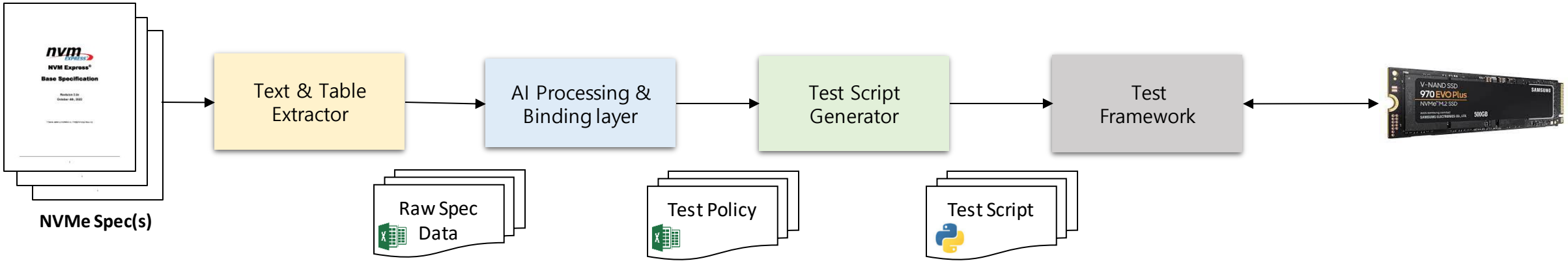
- Rapidly evolving NVMe Spec, *challenges* Test Dev within short time to market
- Manual Test Dev comes with drawbacks like,
  - *Increased* Test Development Time
  - Prone to *human errors* (missing key statements across specifications)
  - Limited *Guarantee* on uniform Test Coverage across features/commands
- Manual Documentation & Maintenance of Tests is *tedious* job

Objective

- *Automatic* Test Suite Generation for NVMe Protocol
- Standard and Uniform Test Coverage *across* NVMe features
- *Customizable* Test Execution Package for NVMe SSDs

Functional Flow

- Extracts Text and Table from Spec PDFs
- Process the Raw Spec Data → Requirements/ Test Policy
- Transforms Test Policy → Test Script
- Plug n Play Test Package



Test Policy (1/2)

Flash Memory Summit

Standard Policy for Uniformity in Test Coverage

- Basic to Advanced Approach

Div.01: Basic Command Test

Targets basic command response check

Div.02: Basic Function Test

The justification of basic processing is checked along with command response

Div.03: Reserved Field Test

Combination with Reserved Fields - defined and Undefined reserved fields is checked.

Div.04: Single Parameter Test

Combination with single parameter range which corresponds to an item below is checked.

Div.05: Multi Parameter Test

Combination with multi parameter range which corresponds to an item below is checked.

Div.06: Combination Test Relevant to Self Function

Combination with the self function which corresponds to an item below is checked.

- Classified into Divisions and Sections
  - Inclusive design for Negative Tests

Div.01: Basic Command Test

Sec.01: Positive

HLTC.001: M_Identify_CNS_0h

Test IDENTIFY CNS = 0h for SCT=0h, SC: 0h for all valid specification parameters listed below

- Check point

1.Identify command

Check item	Expected value
CQ Status field	SCT=0x0, SC= 0x0

-Specification parameter

1.Identify Command

Specification item	Specification val	Remarks
NSID	**Active_NSID, Broadcast_NSID**	Multi_Range
CNS	**0x0**	

Sec.02: Negative

HLTC.001: M_Identify_CNS_0h

Test IDENTIFY CNS = 0h for SCT=0h, SC: 2h, SC = 0Bh for all Invalid specification parameters listed below

- Check point

1.Identify command

Check item	Expected value
CQ Status field	SCT=0x0, SC= 0x2

-Specification parameter

1.Identify Command

Specification item	Specification val	Remarks
NSID	**NSID_0, Inactive_NSID**	Multi_Range
CNS	**0x0**	

Test Policy (2/2)

- Auto Generated Tabular Matrix
  - All Permutations and Combinations

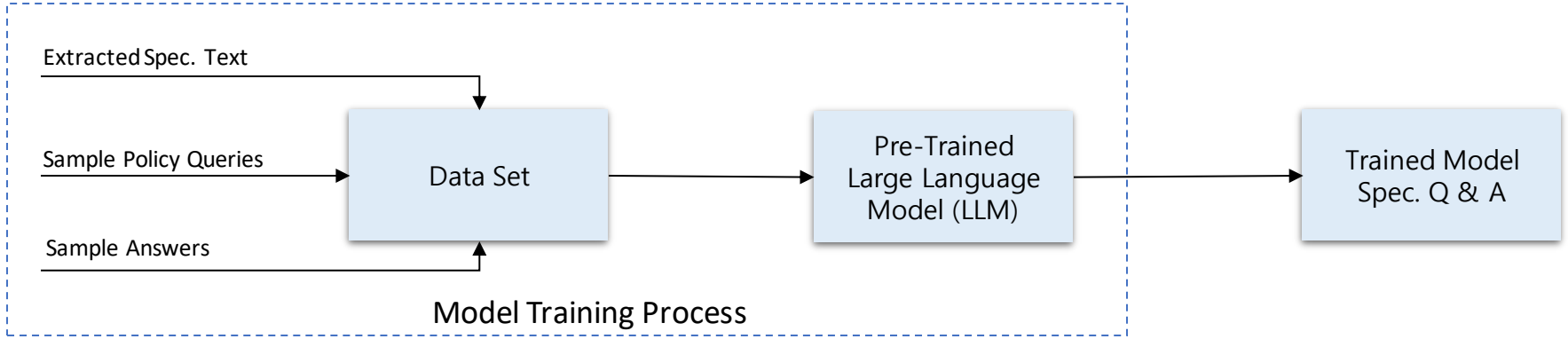
Field	Definition	Test_Value 1	Test_Value 2	Test_Value 3	Test_Value 4	Test_Value 5	
CDW15	Reserved	00000000h	Random				Random = 00000001h-FFFFFFFFh
CDW14	Reserved	00000000h	Random				Random = 00000001h-FFFFFFFFh
CDW13	Reserved	00000000h	Random				Random = 00000001h-FFFFFFFFh
CDW12	Reserved	00000000h	Random				Random = 00000001h-FFFFFFFFh
CDW11	Reserved	00000000h	Random				Random = 00000001h-FFFFFFFFh
CDW10	CNTID	0000h	0001h	Random			Random = 0002h-FFFFh
	Reserved	00h	Random				Random = 01h-FFh
	CNS	00h	01h	02h	03h	10h	
CDW8,9	PRP2	-					
CDW6,7	PRP1	-					
CDW4,5	MPTR	00000000000	Random				Random = 0000000000000001h-FFFFFFFFFFFFFFFFh
CDW2,3	Reserved	00000000000	Random				Random = 0000000000000001h-FFFFFFFFFFFFFFFFh
CDW1	NSID	00000001h	00000002h-N	00000000h	FFFFFFFFh	Random	Random = NNh-FFFFFFFEh
CDW0	CID	-					
	PSDT	00b	Random				Random = 01b-11b
	Reserved	0000b	Random				Random = 0001b-1111b
	FUSE	00b	01b	10b	11b		
	OPC	06h					

- Look up Table for re-usability
  - Global Constants, like NSID variations

Specification Item	Specification Value
NSID_0	0h
Active_NSID	Global_NSID
Inactive_NSID	Unallocated_NSID
Invalid_NSID	>NN to <FFFFFFFEh
NSID_FFFFFFFEh	0xFFFFFFFE
NSID_Broadcast	0xFFFFFFFF

AI Processing & Binding Layer (1/2)

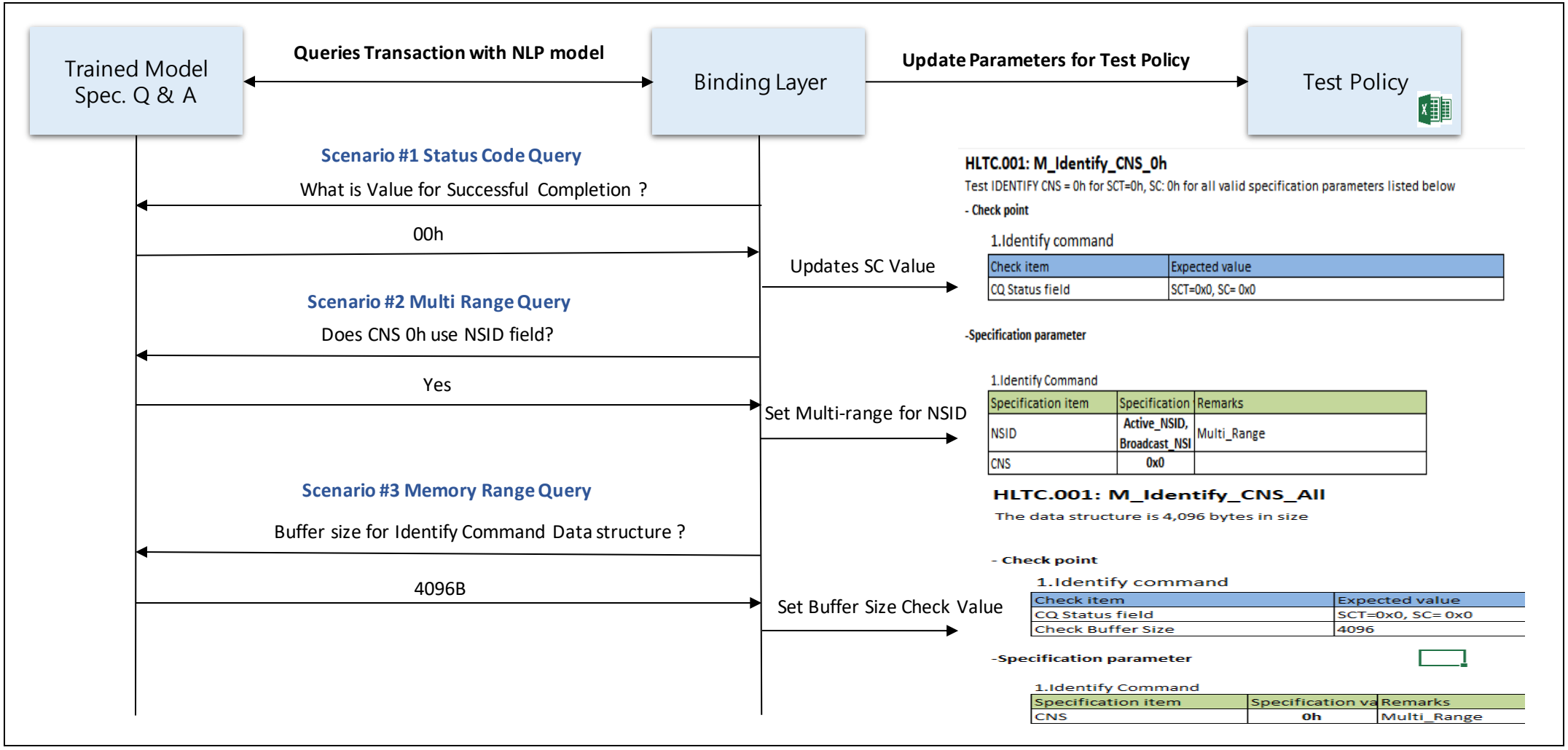
- Train TAPAS model to answer the queries
- Queries based on Test Policy for NVMe Spec Coverage



Sample Query (Training Data)	Sample Answers (Training Data)
Which DWORDs are used in the command ?	Command DWORD 10, Command DWORD 11, and Command DWORD 14
Which Figure to Refer to get values of the DWORD?	Figure 273
Supported Values in the given Table?	02h, 03h, 05h, 07h, 08h, 10h, 11h, 12h, 1Ah, 1Bh
What should be the value if the field is not used?	0h

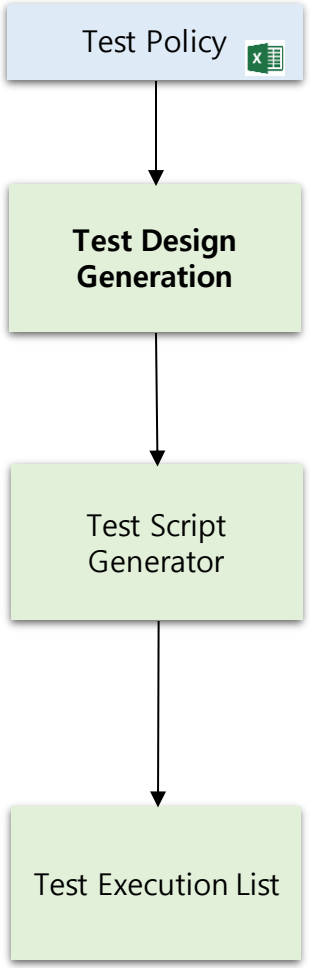
AI Processing & Binding Layer (2/2)

Queries Trained Model, updates Test Policy Document with Test Parameter



Test Script Generator (1/3)

Test Design -> Test Script -> Test Execution List



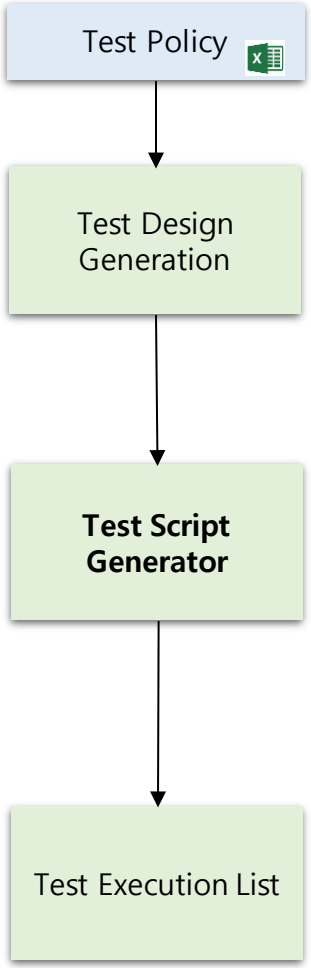
Translates Test Design Tables to Test Case Steps

Script_Name	Input Parameter	Expected Output	Test Purpose	Test Steps

01_01_M_Identify_CNS_0h_001	1.Identify Command
	NSID : Active_NSID, Broadcast_NSID
	CNS : 0x0
		1.Identify Command
		CQ Status field:SCT=0x0, SC= 0x0
			Test IDENTIFY CNS = 0h for SCT=0h, SC: 0h for all valid specification parameters listed below	def setup(self):
				
				def run(self):
				1-*issue_identify_cmd(cns = 0x0, nsid = self.identify_command_1_nsid, ctrlid = 0, cn_spec_id = 0, uuidIndex = 0, csi = 0, sct=0x0 , sc= 0x0, cdw0_res=0,cdw10_res=0,cdw11_res=0,cdw12_res=0,cdw13_res=0,cdw14_res=0,cdw15_res=0)
				
				def teardown(self):

Test Script Generator (2/3)

Test Design -> **Test Script** -> Test Execution List

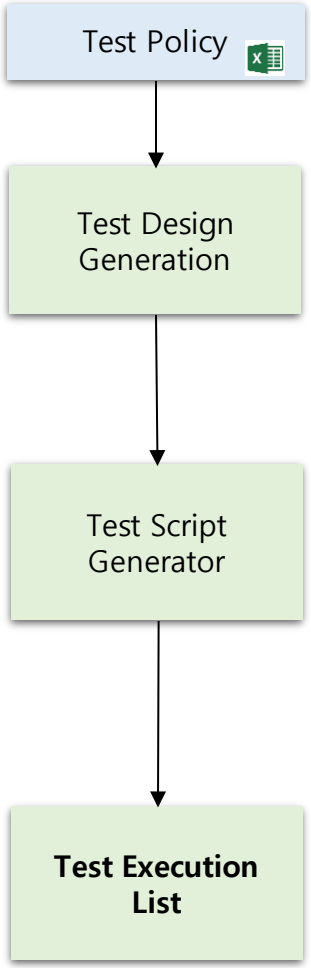


Converts Test Case Steps → Python Test Scripts

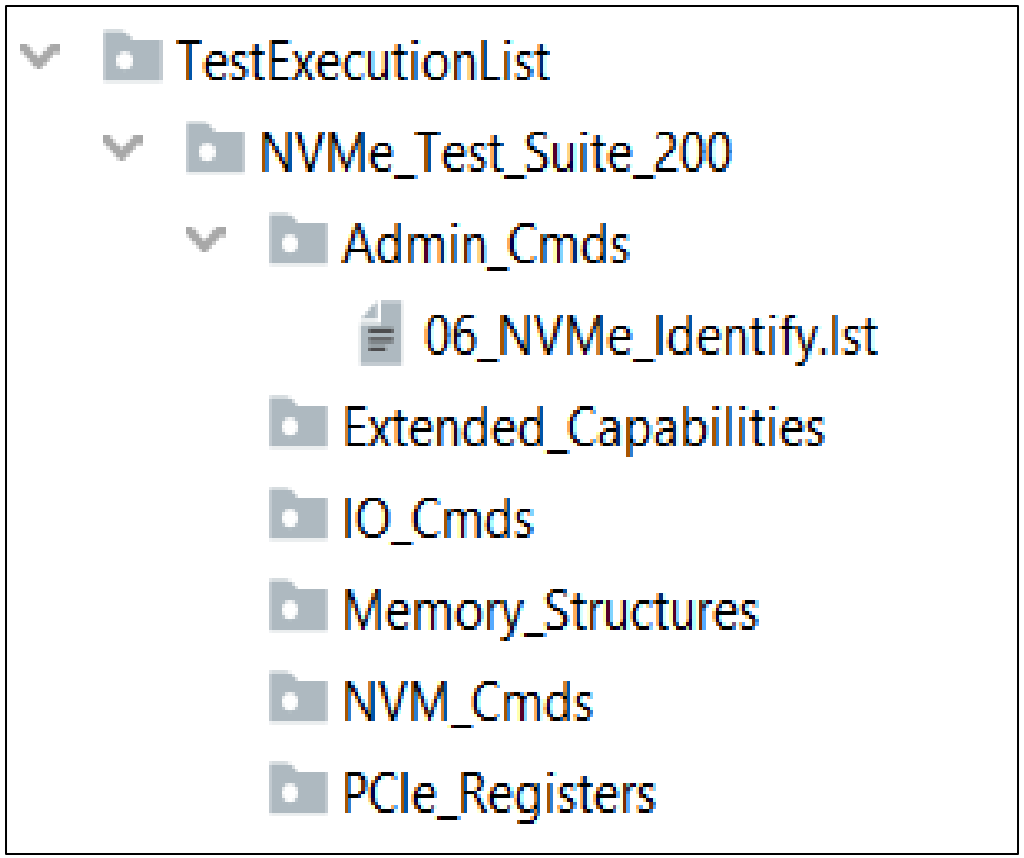
```
"""

*Test Purpose :   Test IDENTIFY CNS = 0h for SCT=0h, SC: 0h for
    all valid specification parameters listed below

* LLTC Details
* 1.Identify Command
LLTC1 : NSID = Active_NSID
LLTC2 : NSID =  Broadcast_NSID


* @Date[DD-MM-YYYY]          @Name                    @Remarks
*  28-06-2023                ATSG                     First Writing
"""

from library_path import *
class _01_01_M_Identify_CNS_0h_001(NVMeInterfaceLib):

    def __init__(self):
},},1:{"self.identify_command_1_":{"NSID": 0xFFFFFFFF
},},}

    def setup(self, dev=None):
        """
            1. Create object
        """
        self.port = self.main_port
        if self.main_ec == False:
            LOG.error("Port ID[{0}] Command lib object creation failed".format(self.port))
            return False
        return True

    def run(self):

    def teardown(self):


if __name__ == "__main__":
    obj = _01_01_M_Identify_CNS_0h_001()
    test_runner_v(obj,obj.test_param)
```

Test Script Generator (3/3)

Test Design -> Test Script -> **Test Execution List**



Logical Grouping of Scripts for Execution

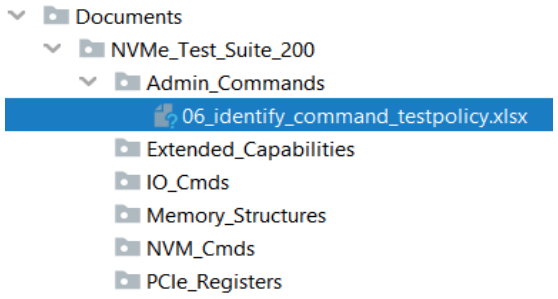


Test Package

Package contains – Test Design Document, Test Script, Test Execution List, User Config. File

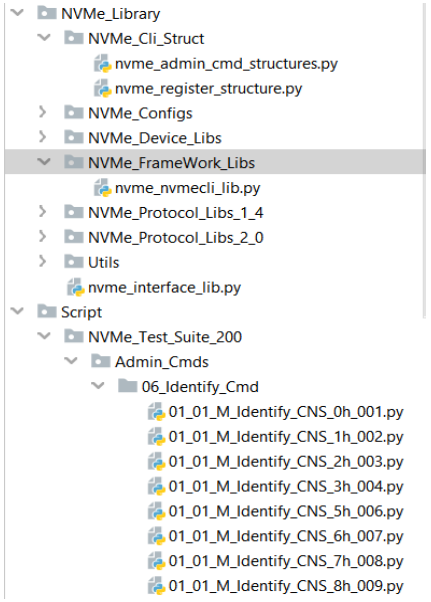
Test Design Document

- Test Design Excel (.xslx)
- Detailed Test Flow



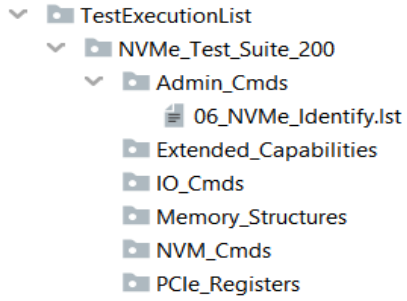
Test Scripts

- Python Test Script
- Organized as per Features/Commands



Test Execution List (.lst)

- Logical Collection of Tests
- Categorized as per Spec Version



User Configurable File

- Enable user selection
- Protocol, Application Selectable

```
APP = "NVMe_CLI"
DEVICE_TYPE = "Generic"
SPEC_VERSION = 200
Failure_Analysis = False
LST = "TRUE"
```

Conclusion

- Automated Solution
  - Faster Test Development for Rapid changing Specification
  - E2E Automation helps to reduce Human Errors
  - Pluggable over Open Source Tools (like nvme-cli, PyTest)
- Uniform Test Coverage
  - Standard Policy classifies coverage to Positive, Negative, Boundary, etc.
  - Policy applicable for both features and commands
- Customizable Test Execution
  - Prefixed Config feature
    - Division, Sections, Mandatory, Optional, etc.
  - Easy to use in TDD (Test Driven Development)