2023 OFA Virtual Workshop

Booting your OS across the NVMe® over Fabrics (NVMe-oF™) Transport

NVM Express® Boot Specification + Boot over NVMe/TCP Reference Implementation

Phil Cayton

(Intel) – Co-chair of the NVM Express Boot Task Group

Agenda

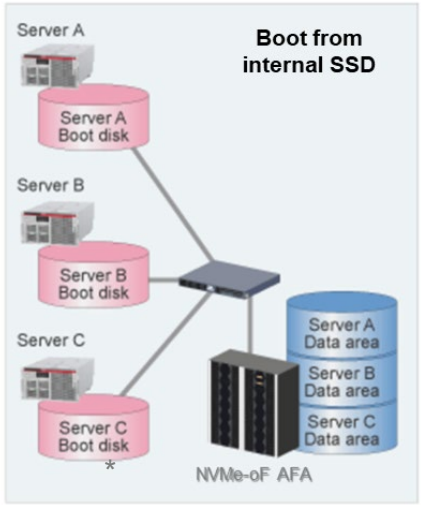
- Standardizing Booting from NVMe® and NVMe-oF™ Namespaces
- NVM Express® (NVMe) Boot Specification Overview
- Ecosystem Cooperation: UEFI and DMTF
- NVMe-oF Boot (UEFI-Based Example)
- Reference Implementations & Future Enhancements
- Q&A

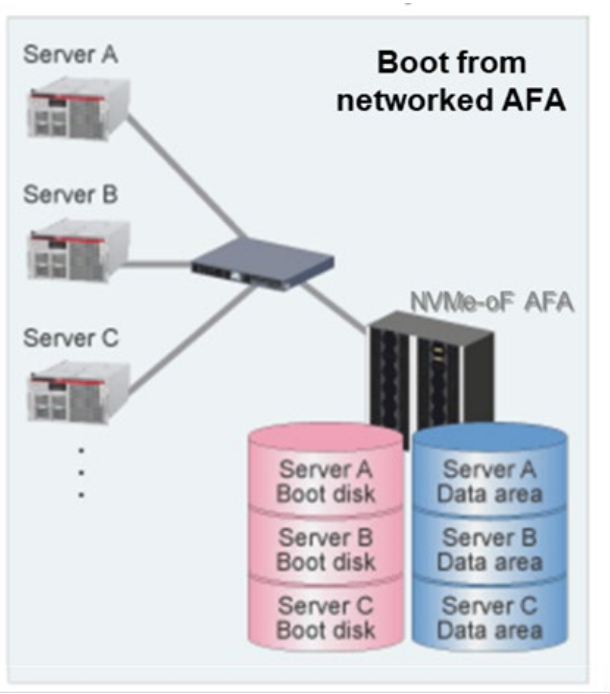
Why Does NVMe® Technology Need a Boot Specification

Currently successful storage networking technologies such as Fibre Channel and iSCSI have standardized solutions that allow attached computer systems to boot from OS images stored on storage nodes.

The lack of a standardized capability in the NVMe-oF™ transport specification presented a barrier for adoption.

This was a missing requirement for a networked storage technology.



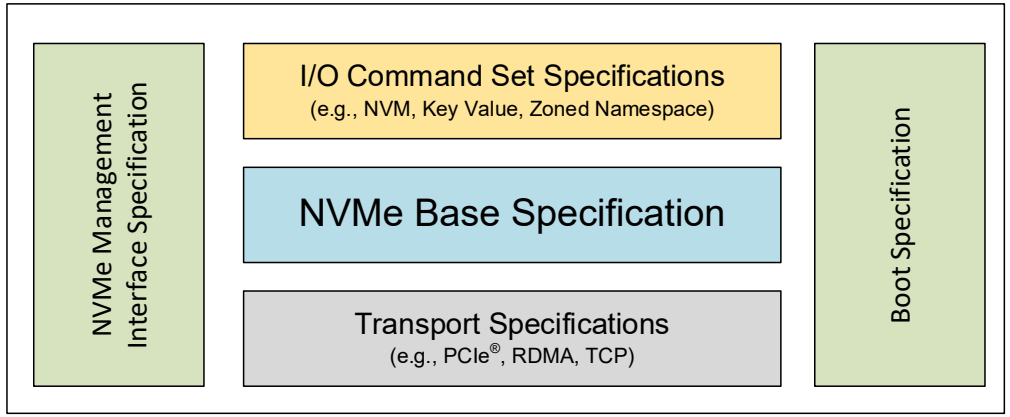


*AFA = All Flash Array Storage System

Standardize Booting from NVMe® and NVMe-oF™ Namespaces

NVM Express® Boot Specification

- Published on NVMExpress.org* 12/2022
- Defines constructs & guidelines for booting from NVM Express interfaces over supported transports
- Version 1.0 defines extensions to the NVMe interface for booting over NVMe/TCP transport
  - Normative content describes
    - General concepts for NVMe/NVMe-oF boot
    - Mechanism for boot device enumeration and configuration handoff from Pre-OS to OS environments (ACPI tables)
  - Informative content Introduces
    - Boot stages and flow in a UEFI pre-OS environment
    - Implementation and adoption guidelines and best-practices
      - NVMe-oF boot configuration in the Pre-boot environment
      - Mechanics for consumption of ACPI tables by the OS
      - OS and fabric transport specifics

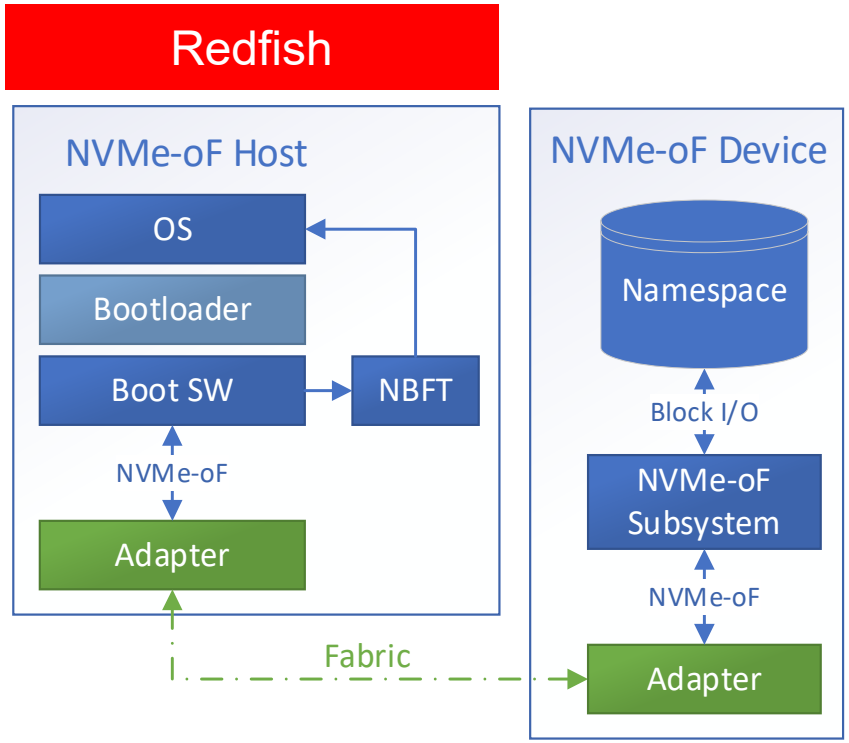


*https://nvmexpress.org/specification/nvme-boot-specification/

Ecosystem Cooperation to Enable Standardization

Collaboration with the following ecosystem and industry partners was key

1. UEFI Forum:
   - ACPI Specification (6.5*): Adds ACPI NVM Express® Boot Firmware Table (NBFT) to ACPI.org
   - UEFI System Specification (2.10*): Adds device path extension for NVMe-oF™ boot
2. DMTF: Adds standardization for Redfish NVMe-oF transport ‘secrets registry’ in the 2021.4 release
3. NVMe® Boot Spec 1.0 – introduces standardization of booting over NVMe and NVMe-oF transport (starting with Booting over NVMe/TCP transport)
4. Public reference implementation: The code for booting over NVMe-oF transport is based on open-source frameworks.

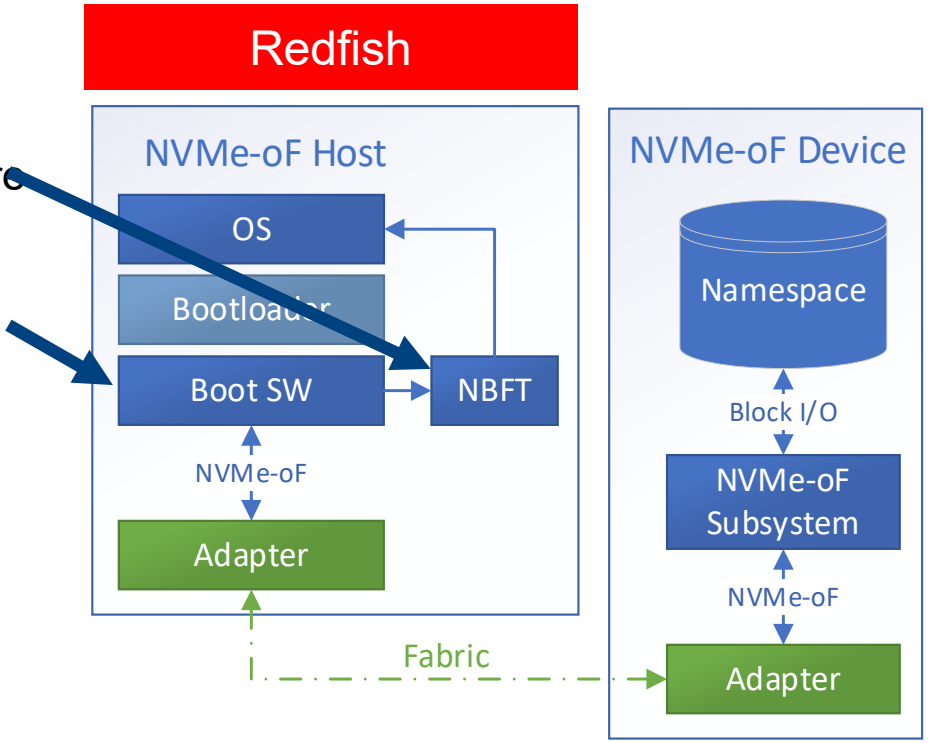


*See references slide for publication locations

Ecosystem Cooperation to Enable Standardization

Collaboration with the following ecosystem and industry partners was key

1. UEFI Forum:
   - ACPI Specification (6.5*): Adds ACPI NVM Express® Boot Firmware Table (NBFT) to ACPI.org
   - UEFI System Specification (2.10*): Adds device path extension for NVMe-oF™ boot
2. DMTF: Adds standardization for Redfish NVMe-oF transport ‘secrets registry’ in the 2021.4 release
3. NVMe® Boot Spec 1.0 – introduces standardization of booting over NVMe and NVMe-oF transport (starting with Booting over NVMe/TCP transport)
4. Public reference implementation: The code for booting over NVMe-oF transport is based on open-source frameworks.



*See references slide for publication locations

UEFI Collaboration

- Added to the ACPI XSDT Signature Table*
- NVMe® over Fabrics Device Path extension to support for NVMe-oF™ boot from UEFI System Spec**

Mnemonic	Byte Offset	Byte Length	Description
Type	00	1	Type 3 – Messaging Device Path
Sub-Type	01	1	Sub-Type 34 - NVMe-oF Namespace Device Path
Length	02	2	Length of this Structure in bytes. Length is 20+n bytes where n is the length of the SubsystemNQN
NIDT	04	1	Namespace Identifier Type (NIDT), for globally unique type values defined in the CNS 03h NIDT field (1h, 2h, or 3h) by the NVM Express® Base Specification®.
NID	05	16	Namespace Identifier (NID), a globally unique val-ue defined in the Namespace Identification De-scriptor list (CNS 03h) by the NVM Express® Base Specification in big endian format.
SubsystemNQN	21	n	Unique identifier of an NVM subsystem stored as a null-terminated UTF-8 string of n-bytes in compli-ance with the NVMe Qualified Name in the NVM Express® Base Specification. Subsystem NQN is used for purposes of identification and authentica-tion. Maximum length of 224 bytes.

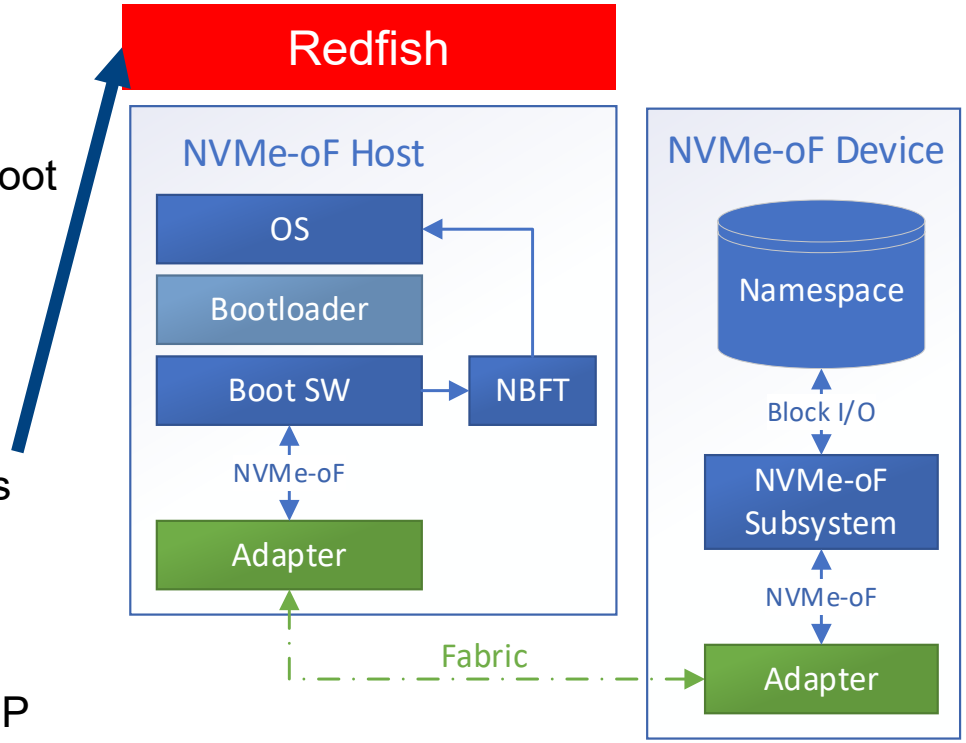
*https://uefi.org/specs/ACPI/6.5/05_ACPI_Software_Programming_Model.html

**https://uefi.org/specs/UEFI/2.10/10_Protocols_Device_Path_Protocol.html#nvme-over-fabric-nvme-of-namespace-device-path

Ecosystem Cooperation to Enable Standardization

Collaboration with the following ecosystem and industry partners was key

1. UEFI Forum:
   - ACPI Specification (ECR into 6.5*): Adds ACPI NVM Express® Boot Firmware Table (NBFT) to ACPI.org
   - UEFI System Specification (ECR into 2.10*): Adds device path extension for NVMe-oF™ boot
2. DMTF: Adds standardization for Redfish NVMe-oF transport 'secrets registry' in the 2021.4 release
3. NVMe® Boot Spec 1.0 – introduces standardization of booting over NVMe and NVMe-oF transport (starting with Booting over NVMe/TCP transport)
4. Public reference implementation: The code for booting over NVMe-oF transport is based on open-source frameworks.



*See references slide for publication locations

DMTF Collaboration

Adds standardization for NVMe-oF™ transport 'secrets registry' for 2021.4

Property	Type	Attributes	Notes
KeyString	string	*read-only required on create (null)*	The string for the key.
KeyType	string (enum)	*read-only required on create (null)*	The format of the key. *For the possible property values, see KeyType in Property details.*
NVMeoF {	object	*(null)*	NVMe-oF specific properties.
HostKeyId	string	*read-write (null)*	The identifier of the host key paired with this target key.
NQN	string	*read-only required on create (null)*	The NVMe Qualified Name (NQN) of the host or target subsystem associated with this key.
OEMSecurityProtocolType	string	*read-only (null)*	The OEM security protocol that this key uses.
SecureHashAllowList [ ]	array (string (enum))	*read-only (null)*	The secure hash algorithms allowed with the usage of this key. *For the possible property values, see SecureHashAllowList in Property details.*
SecurityProtocolType	string (enum)	*read-only (null)*	The security protocol that this key uses. *For the possible property values, see SecurityProtocolType in Property details.*

KeyType: The format of the key.

string	Description
NVMeoF	An NVMe-oF key.

SecureHashAllowList: The secure hash algorithms allowed with the usage of this key.

string	Description
SHA256	SHA-256.
SHA384	SHA-384.
SHA512	SHA-512.

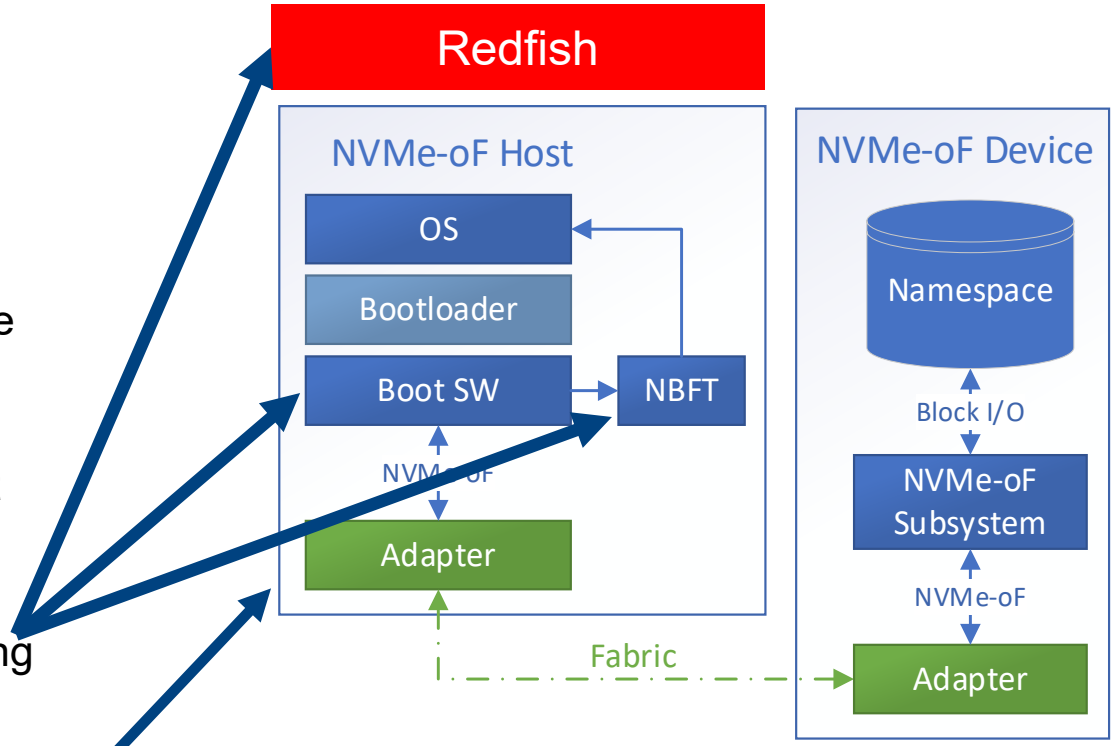
SecurityProtocolType: The security protocol that this key uses.

string	Description
DHHC	Diffie-Hellman Hashed Message Authentication Code Challenge Handshake Authentication Protocol (DH-HMAC-CHAP).
OEM	OEM.
TLS_PSK	Transport Layer Security Pre-Shared Key (TLS PSK).

Ecosystem Cooperation to Enable Standardization

Collaboration with the following ecosystem and industry partners was key

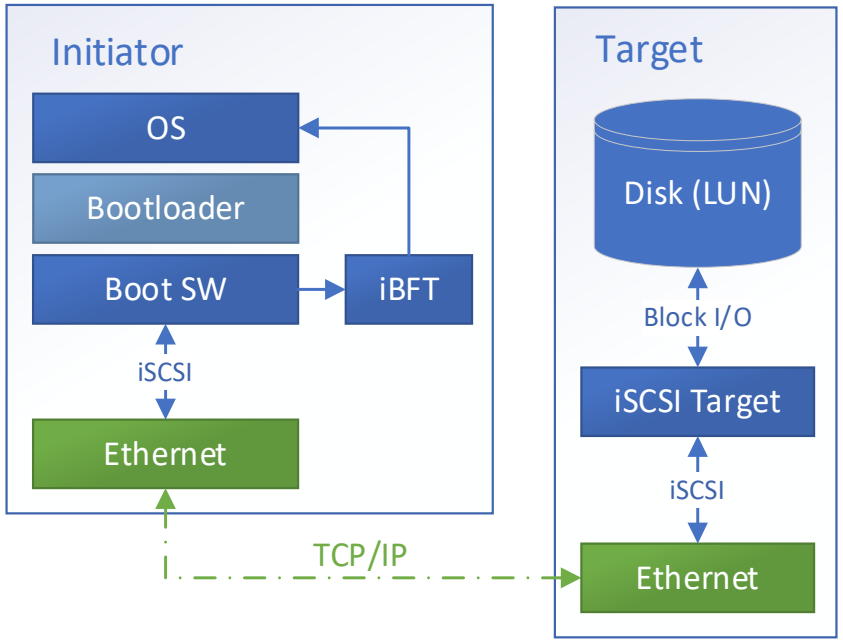
1. UEFI Forum:
   - ACPI Specification (ECR into 6.5*): Adds ACPI NVM Express® Boot Firmware Table (NBFT) to ACPI.org
   - UEFI System Specification (ECR into 2.10*): Adds device path extension for NVMe-oF™ boot
2. DMTF: Adds standardization for Redfish NVMe-oF transport 'secrets registry' in the 2021.4 release
3. NVMe® Boot Spec 1.0 – introduces standardization of booting over NVMe and NVMe-oF transport (starting with Booting over NVMe/TCP transport)
4. Public reference implementation: The code for booting over NVMe-oF transport is based on open-source frameworks.



*See references slide for publication locations

Standardize Booting from NVMe® and NVMe-oF™ Namespaces

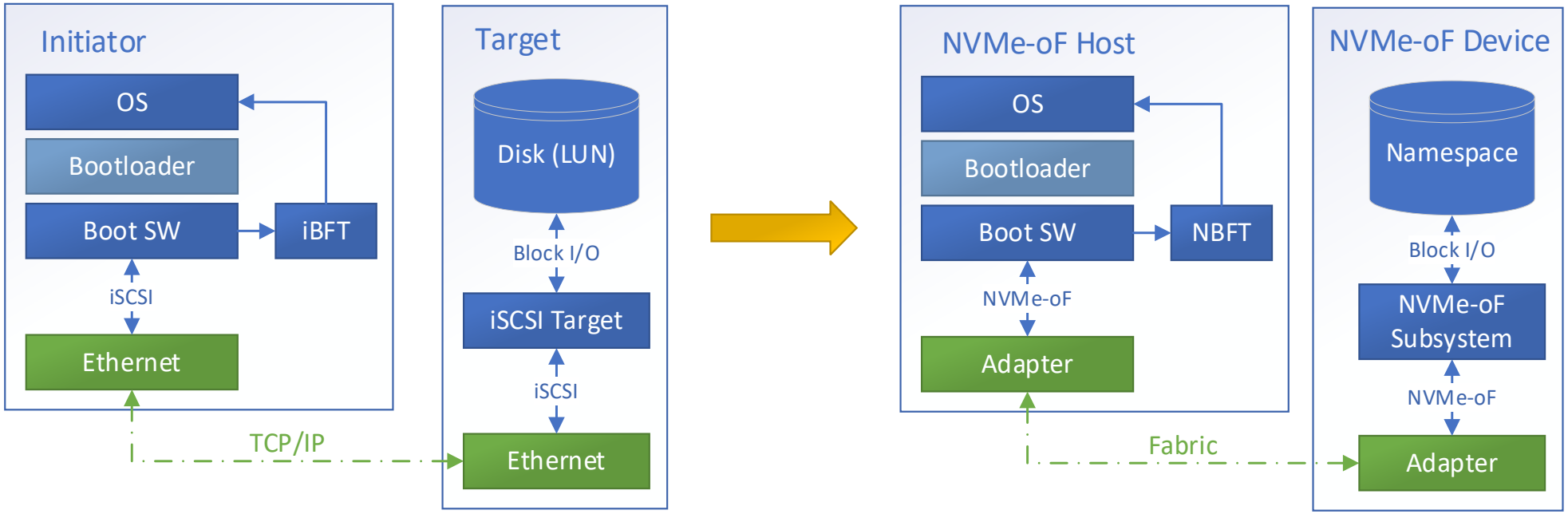
NVMe/TCP boot enabled standardization to leverage past iSCSI lessons and ecosystem enablement



Standardize Booting from NVMe® and NVMe-oF™ Namespaces

NVMe/TCP boot enabled standardization to leverage past iSCSI lessons and ecosystem enablement

Boot from NVMe/TCP transport main concepts are similar to booting from iSCSI



Delta between booting from iSCSI and booting from NVMe-oF

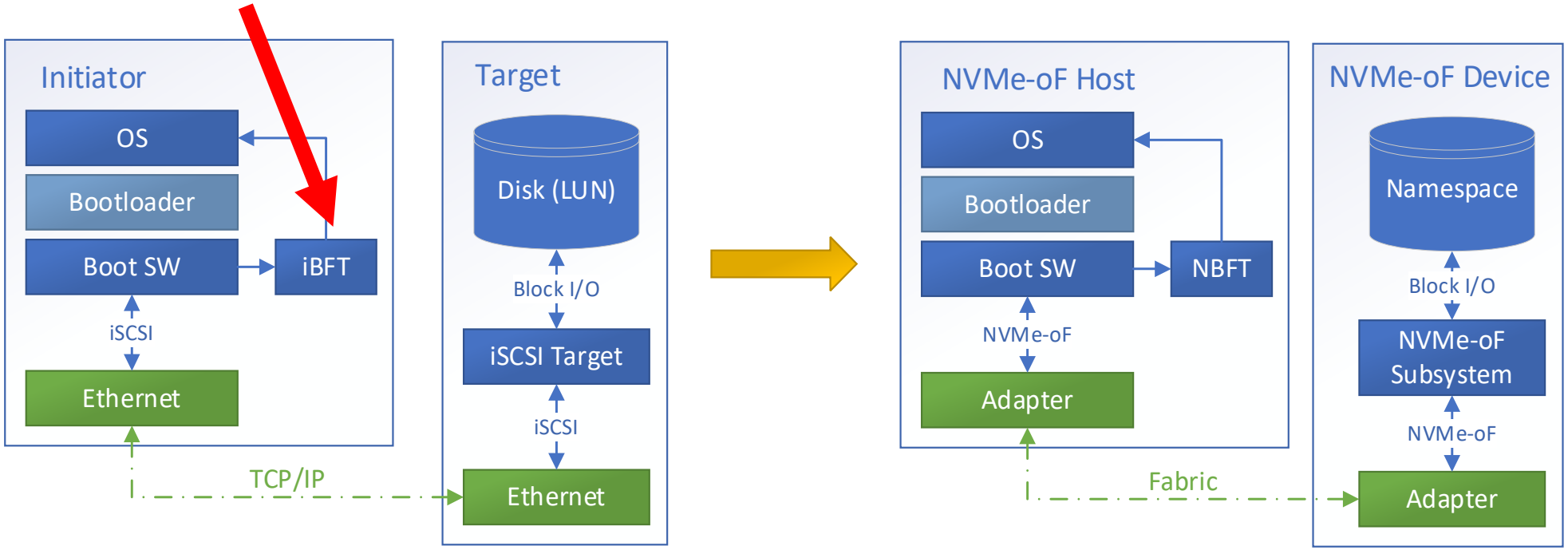
Standardize Booting from NVMe® and NVMe-oF™ Namespaces

NVMe/TCP boot enabled standardization to leverage past iSCSI lessons and ecosystem enablement

Boot from NVMe/TCP transport main concepts are similar to booting from iSCSI

iSCSI enabled boot and OS handover through a mechanism called the “iSCSI Boot Firmware Table” (iBFT)

iBFT contains information to be shared between BIOS / pre-boot environments and the OS



Delta between booting from iSCSI and booting from NVMe-oF

Standardize Booting from NVMe® and NVMe-oF™ Namespaces

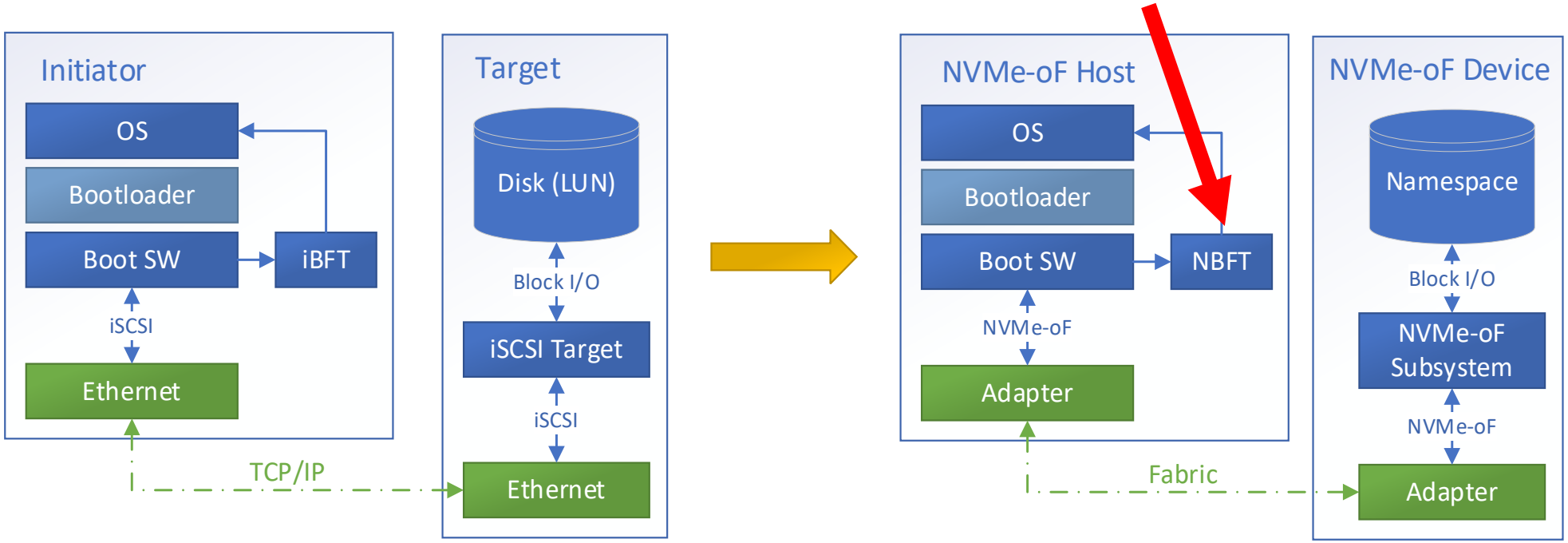
NVMe/TCP boot enabled standardization to leverage past iSCSI lessons and ecosystem enablement

Boot from NVMe/TCP transport main concepts are similar to booting from iSCSI

iSCSI enabled boot and OS handover through a mechanism called the “iSCSI Boot Firmware Table” (iBFT)

iBFT contains information to be shared between BIOS / pre-boot environments and the OS

NVMe technology needs a similar configuration mechanism, NBFT (NVMe Boot Firmware Table)

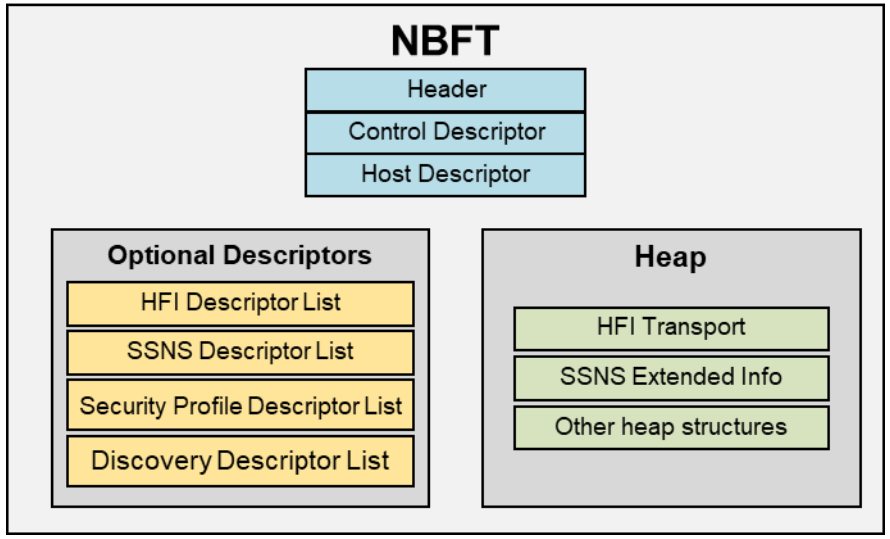


Delta between booting from iSCSI and booting from NVMe-oF

NBFT: Pre-OS to OS Configuration Handoff Mechanism

Information presented to the OS using ACPI XSDT Table at OS boot provides

- local Pre-OS -> OS agnostic configuration communications medium; independent from UEFI, UBOOT, …
- standardized means of passing configuration & connection context from pre-OS Boot environment to an administratively configured OS runtime



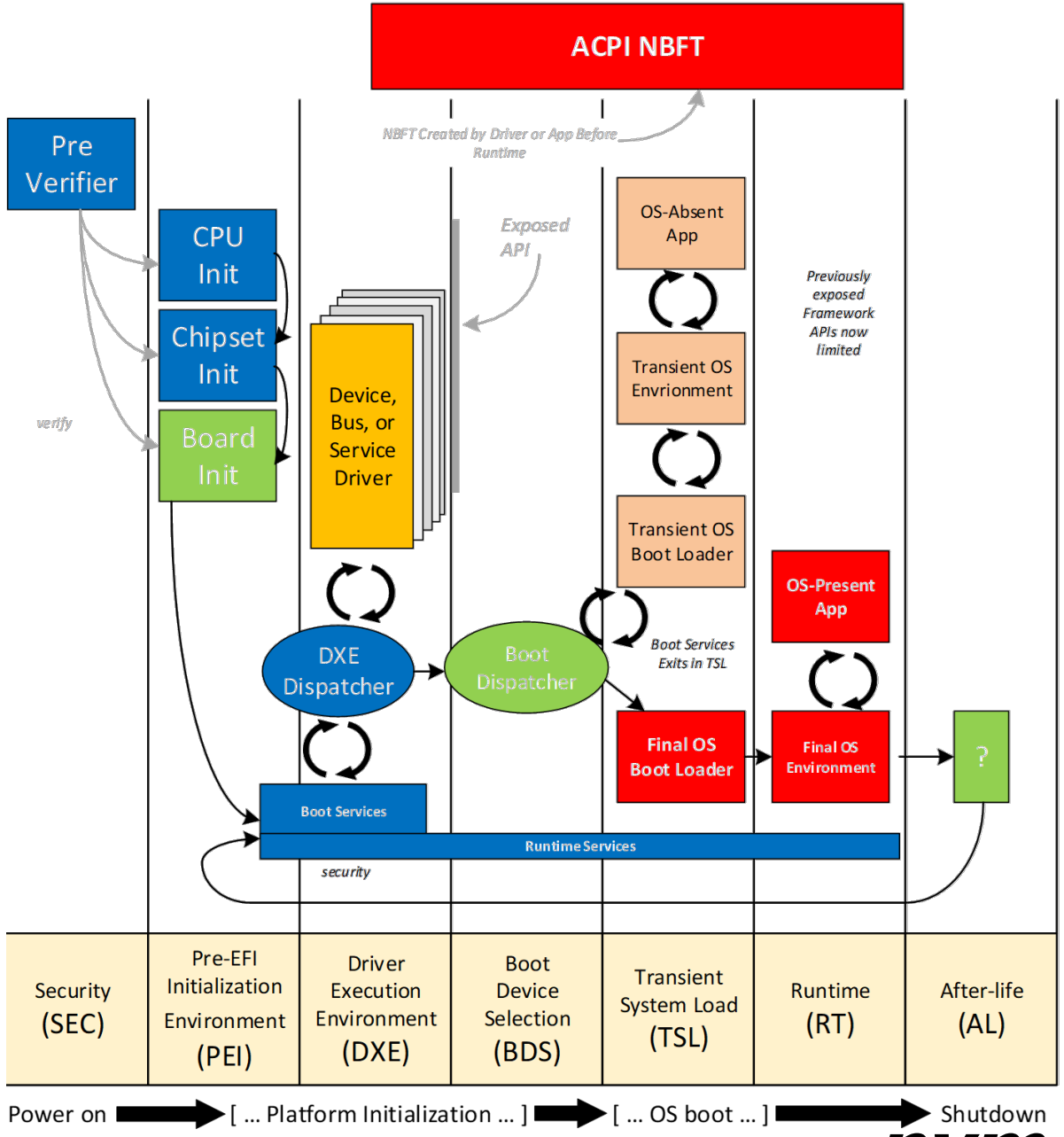
Element	Description
Header	An ACPI structure header with some additional NBFT specific info.
Control Descriptor	Indicates the location of host, HFI, SSNS, security, and discovery descriptors.
Host Descriptor	Host information.
HFI Descriptor	An indexable table of HFI Descriptors, one for each fabric interface on the host.
Subsystem Namespace Descriptor	An indexable table of SSNS Descriptors.
Security Descriptor	An indexable table of Security descriptors.
Discovery Descriptor	An indexable table of Discovery Ddescriptors.
HFI Transport Descriptor	Indicated by an HFI Descriptor, corresponds to a specific transport for a single HFI.
SSNS Extended Info Descriptor	Indicated by an SSNS Descriptor if needed.

https://nvmexpress.org/specification/nvme-boot-specification/

UEFI Boot Phases

The seven phases in a UEFI boot sequence*

1. Security (SEC)
2. Pre-EFI Initialization (PEI)
3. Drive Execution Environment (DXE)
4. Boot Device Selection (BDS)
5. Transient System Load (TSL)
6. Runtime (RT)
7. After Life (AL)



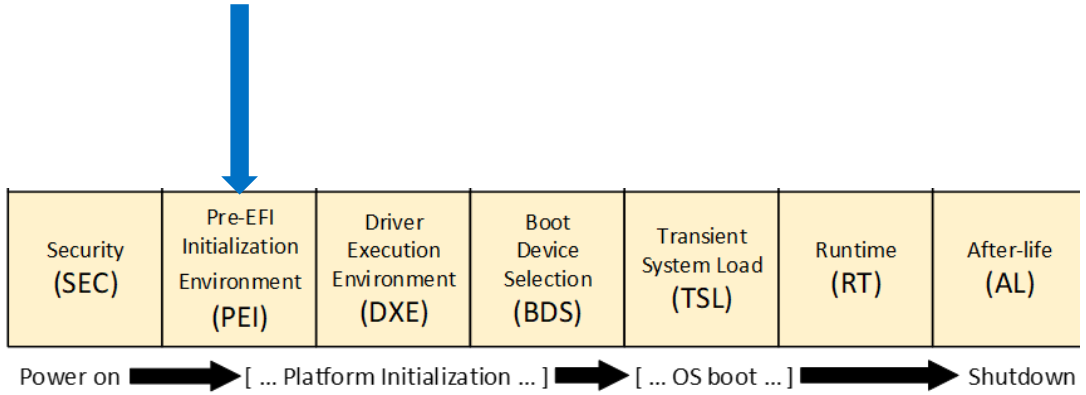
*Tianocore: EDK2 Build Specification

Configuring NVMe-oF™ Boot (UEFI-based example): Pre-Operating System Boot

Boot Attempt configuration is stored in UEFI variables.

Administrator configures Pre-OS driver:

- target subsystem NQN
- target IP address
- target port #
- target namespace
- host NQN
- security related info

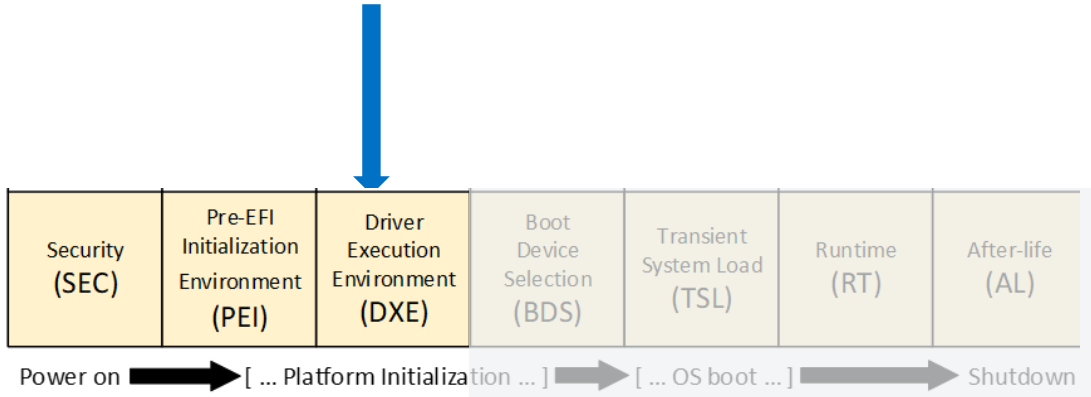


Configuring NVMe-oF™ Boot (UEFI-based example): Pre-Operating System Boot

New functionality

Driver Execution Environment phase: DXE driver supporting NVMe-oF boot is loaded and executed:

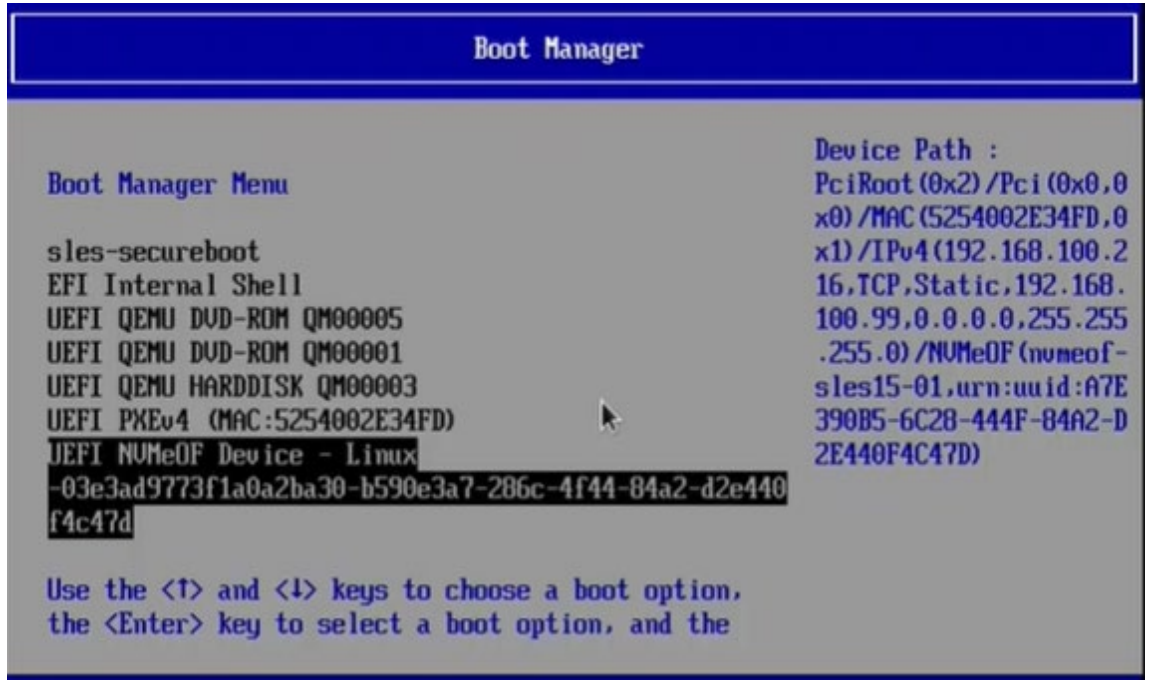
- reads configuration from UEFI variables
- sets up network (interfaces, routing, …)
- (optionally) retrieves authentication credentials
- (optionally) performs discovery and authentication
- connects to NVMe® subsystems provides namespaces to the UEFI Boot Manager as block devices
- stores the configuration in the NBFT: can later be accessed by the OS as an ACPI table

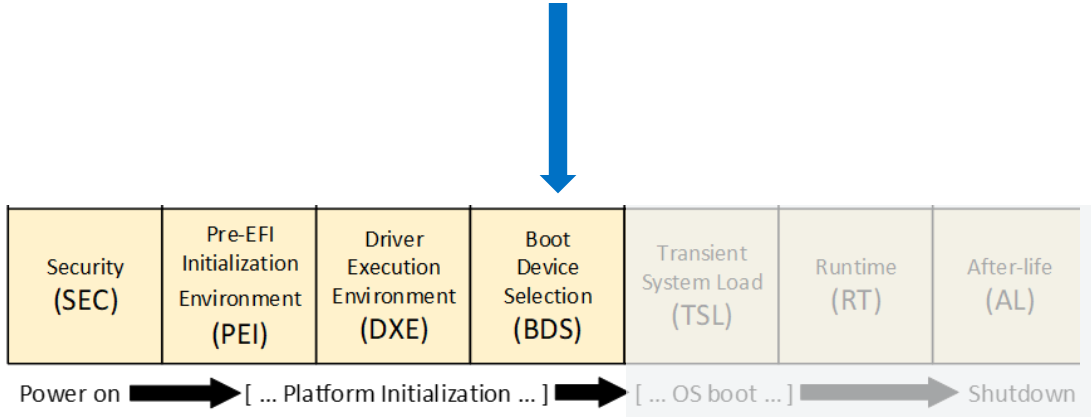


Configuring NVMe-oF™ Boot (UEFI-based example): Pre-Operating System Boot

Existing functionality

Boot Device Selection phase: The Namespace can then be selected as final boot device for OS boot



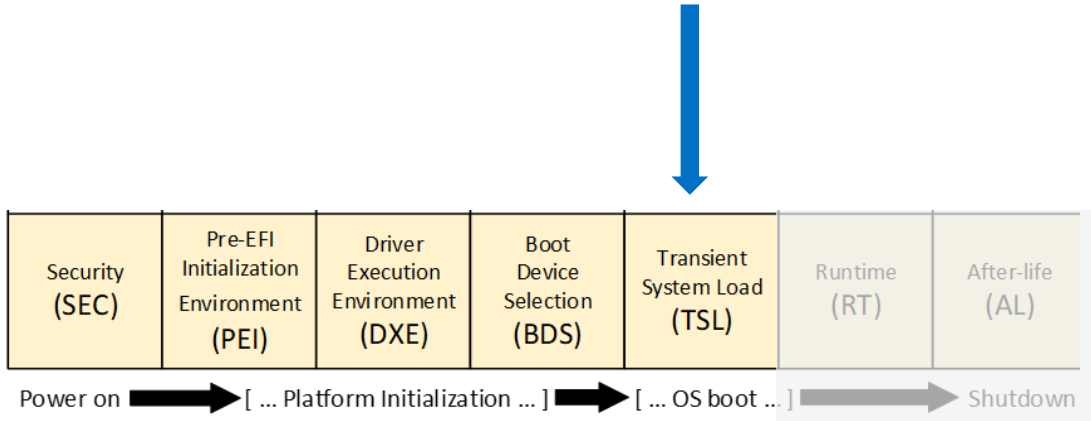


Configuring NVMe-oF™ Boot (UEFI-based example): Pre-Operating System Boot

Existing functionality

Transient System Load phase:

- OS image loaded from boot device
- UEFI hands over execution to OS specific boot loader
- OS Boot Loader continues the OS boot

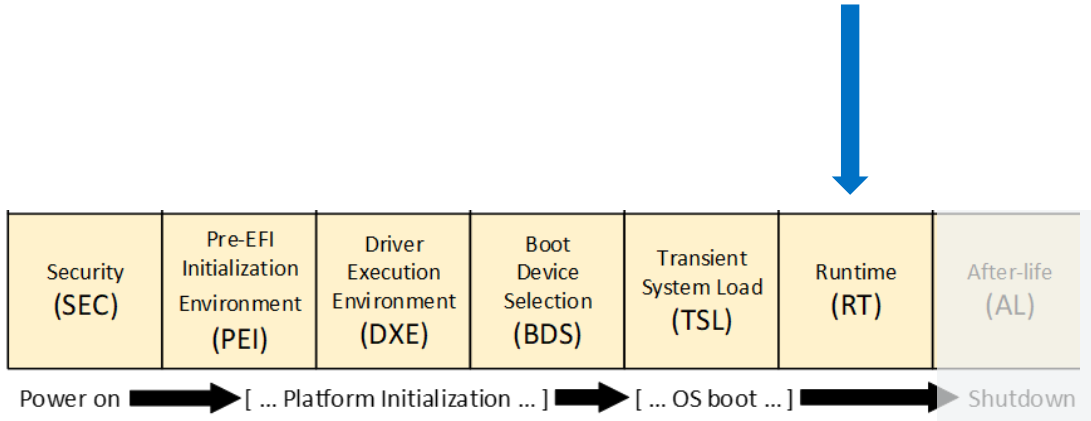


At this point, the NBFT has been generated, stored in main memory, and can be accessed by the OS as an ACPI table

Configuring NVMe-oF™ Boot (UEFI-based example): OS Transition to Runtime

Runtime phase:

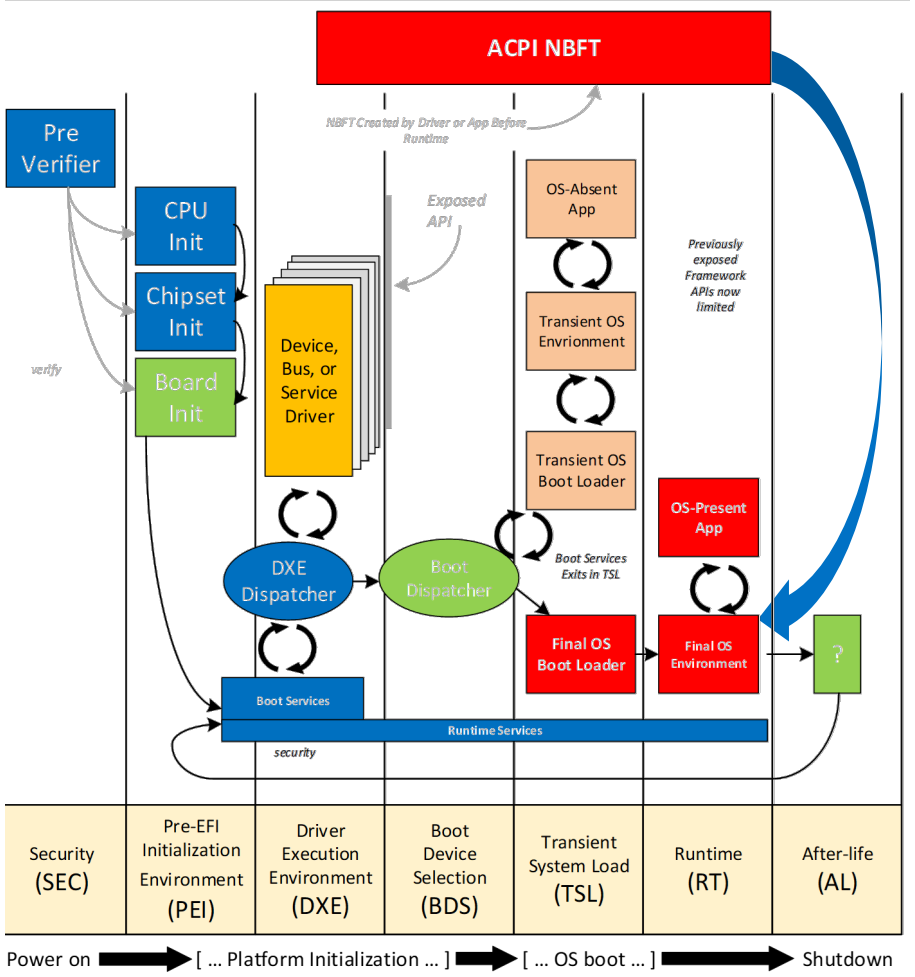
- read the configuration from the NBFT
- set up the network (interfaces, routing, …)
- (optionally) retrieve authentication credentials
- (optionally) perform discovery and authentication
- connect to NVMe® subsystems
- provide namespaces to other parts of the OS



Configuring NVMe-oF™ Boot (UEFI-based example): Typical OS Handover and Initialization

- Normal operating system boot:
  - To persist info to restore NVMe-oF transport connections, OS may either:
    - continue using the NBFT
    - Use OS specific mechanism

- Operating system installation:
  - A user may either:
    a) use the NBFT provided host NQN as its own host NQN
    b) set a separate host NQN (if NVMe-oF subsystem supports multiple host NQNs)



Public Reference Implementation Based on UEFI

Open-source Reference code* for booting over NVMe-oF™ transport is based

- on the NVM Express® Boot Spec 1.0
- on open-source frameworks

Developed by a subset of NVM Express member companies including:

To be released* under BSD-3-Clause (or other open-source license as required by components)

*https://github.com/timberland-sig

Reference Implementation of Booting over NVMe®/TCP transport

Pre-OS time of boot:

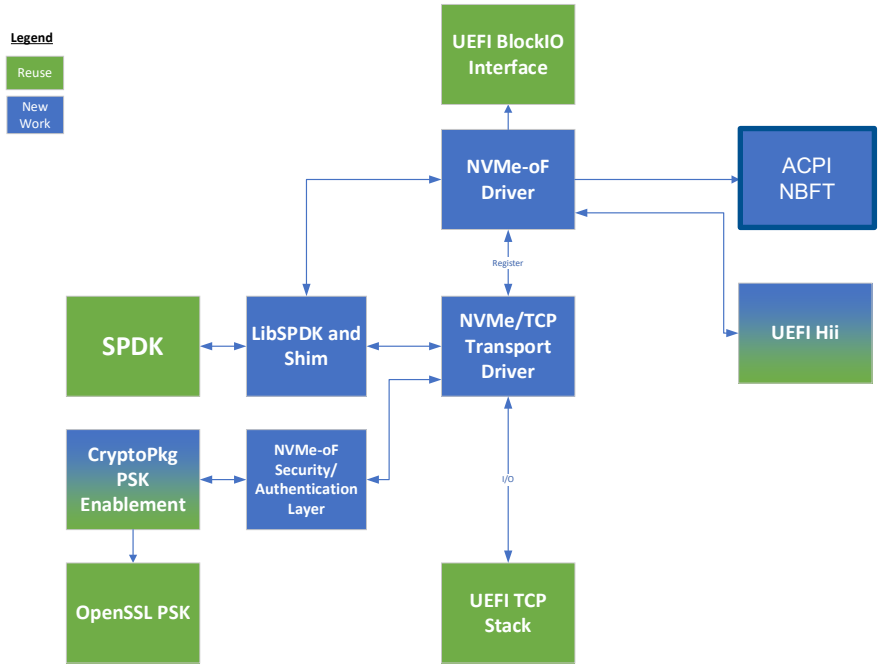
- EDK2 NVMe-oF™ UEFI Driver for the NVMe/TCP transport
  - ACPI NBFT will be produced by this UEFI implementation prior to OS boot

OS Boot and Runtime:

- Linux® reference implementation that:
  - Exposes the NBFT to the user-space
  - Consumes the NBFT contents to connect to configured namespaces
- Enables common tools (e.g., dracut, nvme-cli) to use the NBFT

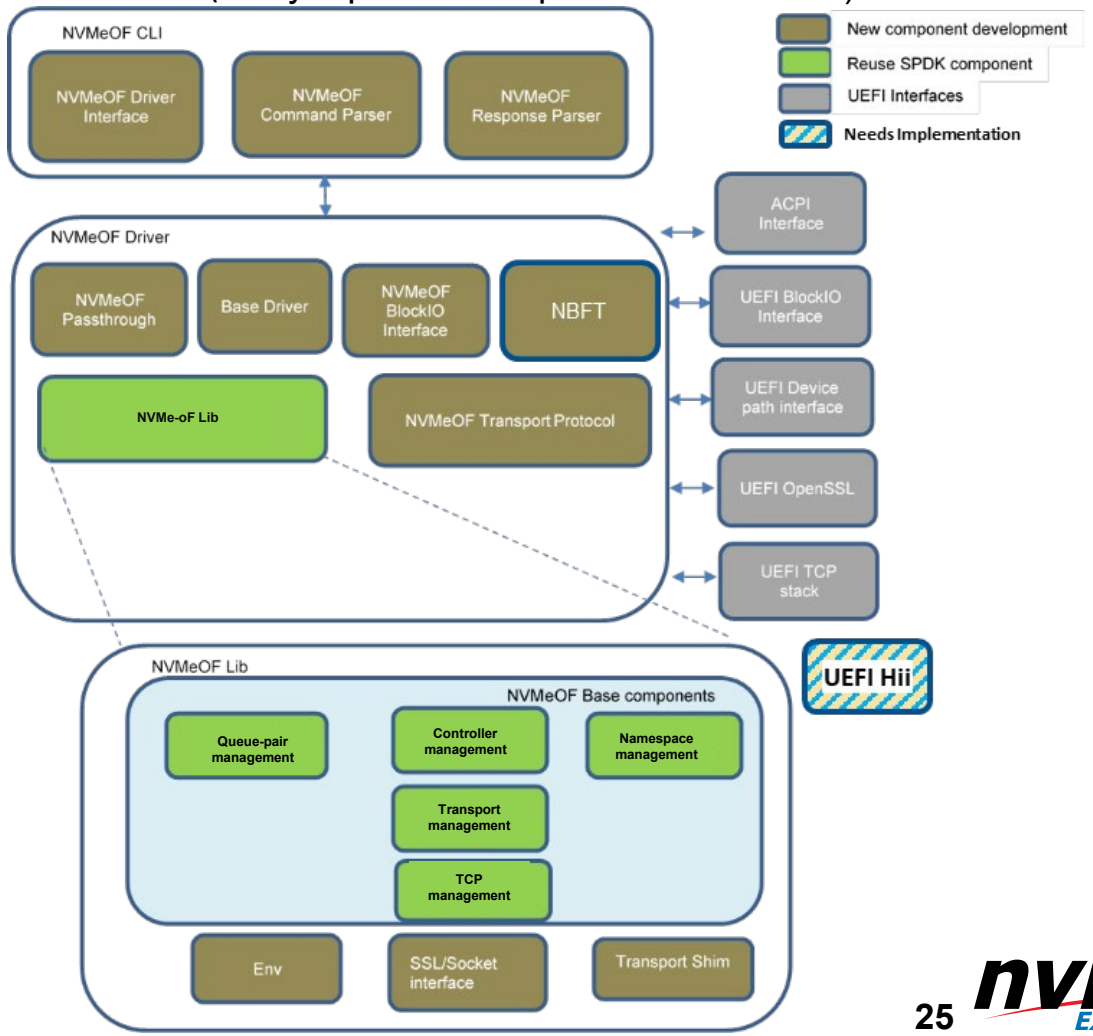
Configuring NVMe-oF™ Boot (UEFI-based example): Pre-Operating System Boot

EDK2 Concept Architecture



Development

EDK2 Reference Architecture as implemented
(Not yet published upstream for review)



Pre-Boot Environment Configuration Tool

nvmeofcli for EFI:

Command Line tool to facilitate basic diagnostics and interoperability with pre-OS reference driver

nvmeofcli list command

```
FS0:\> NvmeOfCli.efi list
-----------
Node       : nvme1n1
NID        : b25579bd-77c1-4507-b7e9-4166612e50b9
SN         : 855b090558d284bd
Model      : Linux
NSID       : 2
Usage      : 6 GiB
Format     : 512
FW Rev     : 5.8.0-48
```

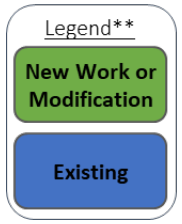
nvmeofcli connect command

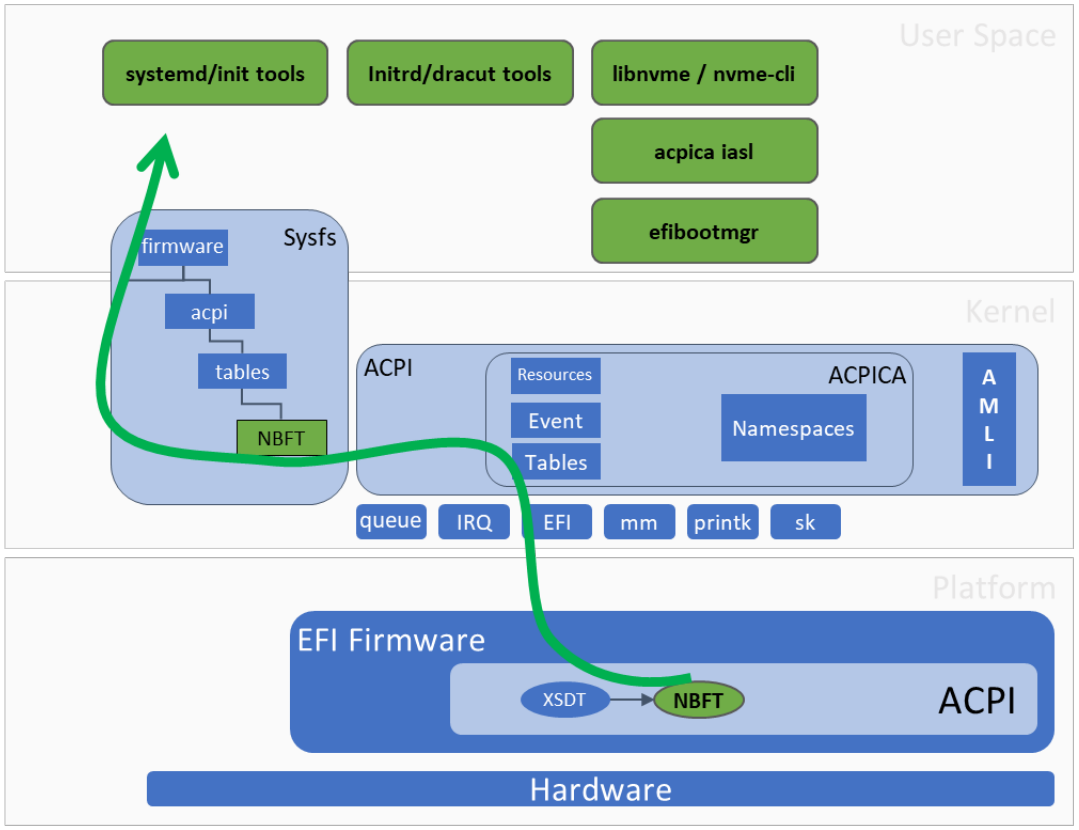
```
FS0:\> NvmeOfCli.efi connect -n nvmet-test-40-3 -t tcp -a 10.118.242.40 -s 4422
--mac 52:54:00:12:34:56 --ipmode 0 --localip 192.168.122.76 --subnetmask 255.255
.255.0 --gateway 192.168.122.1
Connected Successfully
-----------
Node       : nvme1n1
NID        : b25579bd-77c1-4507-b7e9-4166612e50b9
SN         : 855b090558d284bd
Model      : Linux
NSID       : 2
Usage      : 6 GiB
Format     : 512
FW Rev     : 5.8.0-48
-----------
```

OS Handoff Enablement in Reference Design

OS Handoff Enablement in Reference Design

- Linux Kernel support for ACPI “NBFT” Table
- User-Space Device Connection and Configuration tools consuming Linux sysfs
- initrd/dracut changes to support NVMe®/TCP transport:
  - Detects NBFT presence
  - connects pertinent networking
  - uses nvme-cli to connect to NVMe Subsystems/Namespaces





nvme-cli – New subcommands: nvme show-nbft free-text format

```
[root@localhost nvme-cli]# .build/nvme show-nbft --help
Usage: nvme show-nbft <device> [OPTIONS]

Show ACPI NBFT table conects

Options:
  [  --output-format=<FMT>, -o <FMT> ]  --- Output format: normal|json
  [  --subsystem, -s ]                  --- show NBFT subsystems
  [  --hfi, -H ]                        --- show NBFT HFIs
  [  --discovery, -d ]                  --- show NBFT discovery controllers
  [  --nbft-path=<STR>, -P <STR> ]      --- user-defined path for NBFT tables
```

```
NBFT Subsystems:
Index Host-NQN
Transport Address                                   SvcId HFIs
----- ------------------------------------------------------------
1     nqn.1988-11.com.dell:powerstore:00:2a64abf1c5b81F6C4549
tcp       100.71.103.48                             4420  1
2     nqn.1988-11.com.dell:powerstore:00:2a64abf1c5b81F6C4549
tcp       100.71.103.49                             4420  1

NBFT HFIs:

Index Transport PCI Address  MAC Address       DHCP IP Address
Subnet Mask Bits Gateway                                 DNS
----- --------- ------------ ----------------- ---- ------------
1     tcp       0:40:0.0     b0:26:28:e8:7c:0e yes
100.71.245.232                           24
100.71.245.254                           100.64.0.5

NBFT Discovery Controllers:

Index Discovery-URI
Discovery-NQN
----- ------------------------------------------------------------
1     nvme+tcp://100.71.103.50:8009/
nqn.2014-08.org.nvmexpress.discovery
```

nvme-cli – New subcommands: nvme show-nbft JSON format

```
[root@localhost nvme-cli]# .build/nvme show-nbft -o json -H -d -
s -P /home/nbft_0.65_7jul
[
  {
    "filename":"/home/nbft_0.65_7jul/NBFT",
    "host":{
      "nqn":"nqn.1988-11.com.dell:PowerEdge.R760.1234567",
      "id":"44454c4c-3400-1036-8038-b2c04f313233",
      "host_id_configured":0,
      "host_nqn_configured":0,
      "primary_admin_host_flag":"not indicated"
    },
    "subsystem":[
      {
        "index":1,
        "num_hfis":1,
        "hfis":[
          1
        ],
        "transport":"tcp",
        "transport_address":"100.71.103.48",
        "transport_svcid":"4420",
"subsys_nqn":"nqn.1988-11.com.dell:powerstore:00:2a64abf1c5b81F6
C4549",
        "controller_id":0,
        "asqsz":0,
        "pdu_header_digest_required":0,
        "data_digest_required":0
      },
      {
        "index":2,
        "num_hfis":1,
        "hfis":[
          1
        ],
        "transport":"tcp",
        "transport_address":"100.71.103.49",
        "transport_svcid":"4420",
        "subsys_port_id":0,
        "nsid":148,
        "nid_type":"nguid",
        "nid":"c82404ed9c15f53b8ccf0968002e0fca",
        "subsys_nqn":"nqn.1988-11.com.dell:powerstore:00:2a64abf1c5b81F6C4549",
        "controller_id":0,
        "asqsz":0,
        "pdu_header_digest_required":0,
        "data_digest_required":0
      }
    ],
    "hfi":[
      {
        "index":1,
        "transport":"tcp",
        "pcidev":"0:40:0.0",
        "mac_addr":"b0:26:28:e8:7c:0e",
        "vlan":0,
        "ip_origin":82,
        "ipaddr":"100.71.245.232",
        "subnet_mask_prefix":24,
        "gateway_ipaddr":"100.71.245.254",
        "route_metric":500,
        "primary_dns_ipaddr":"100.64.0.5",
        "secondary_dns_ipaddr":"100.64.0.6",
        "dhcp_server_ipaddr":"100.71.245.254",
        "this_hfi_is_default_route":1,
        "dhcp_override":1
      }
    ],
    "discovery":[
      {
        "index":1,
        "hfi":1,
        "uri":"nvme+tcp://100.71.103.50:8009/",
        "nqn":"nqn.2014-08.org.nvmexpress.discovery"
      }
    ]
  }
]
```

Reference Implementation of Booting over NVMe®/TCP transport

Proof-of-Concept for NVM Express® Boot

- The initial PoC is currently based on an openSUSE distribution

Prerequisites for openSUSE

- An openSUSE image
- A Linux host with working qemu/KVM setup
- A virtual network on the host
- An NVMe-oF™ transport server (nvmet) accessible over this virtual network, with at least one free namespace (10GB or more)

Ongoing work

- It will be ported to work on other distributions as well, such as RedHat
- This will be useful because the details of early OS bring-up differ strongly between distributions

Future Enhancements: Open Source and Ecosystem

- UEFI HII for EDK2 driver
- Support for Authentication/TLS
- Support for DMTF Redfish Secrets
- Additional OS and installer support

Future Enhancements: NVM Express® Boot Specification

- Big Namespace Qty Management in Large Fleets
- Multi-Path Topology Examples
- Support Device Tree
- Setting NVMe-oF™ Boot Entries in OS
- Investigate Booting over Additional Transports

Contributions are welcome!
Join NVM Express and the NVM Express Boot Task Group
https://nvmexpress.org/join-nvme/

Adding new Transport Support to NVM Express® Boot Specification

Header for new HFI Transport Info Descriptor in NBFT

Bytes 00 – 05: Mandatory to describe the Header structure for a new Transport Info Descriptor type

Bytes	Description
00	Structure ID
01	Version
02	HFI Transport Type.
03	Transport Info Version
05:04	HFI Descriptor Index

Thereafter Transport-specific descriptor flags as needed following Figure 13 in the NVMe® Boot Spec

```
"hfi":[
  {
    "index":1,
    "transport":"tcp",
    "pcidev":"0:40:0.0",
    "mac_addr":"b0:26:28:e8:7c:0e",
    "vlan":0,
    "ip_origin":82,
    "ipaddr":"100.71.245.232",
    "subnet_mask_prefix":24,
    "gateway_ipaddr":"100.71.245.254",
    "route_metric":500,
    "primary_dns_ipaddr":"100.64.0.5",
    "secondary_dns_ipaddr":"100.64.0.6",
    "dhcp_server_ipaddr":"100.71.245.254",
    "this_hfi_is_default_route":1,
    "dhcp_override":1
  }
],
```

Transports may require a new ECR to the UEFI System Spec if they do not already have a Device Path Messaging Type supporting them

References and Repositories

NVM Express®: https://nvmexpress.org/developers/nvme-specification/

UEFI 2.10:
https://uefi.org/specs/UEFI/2.10/10_Protocols_Device_Path_Protocol.html

ACPI 6.5:
https://uefi.org/specs/ACPI/6.5/05_ACPI_Software_Programming_Model.html

Open-Source Software Repos: https://github.com/timberland-sig

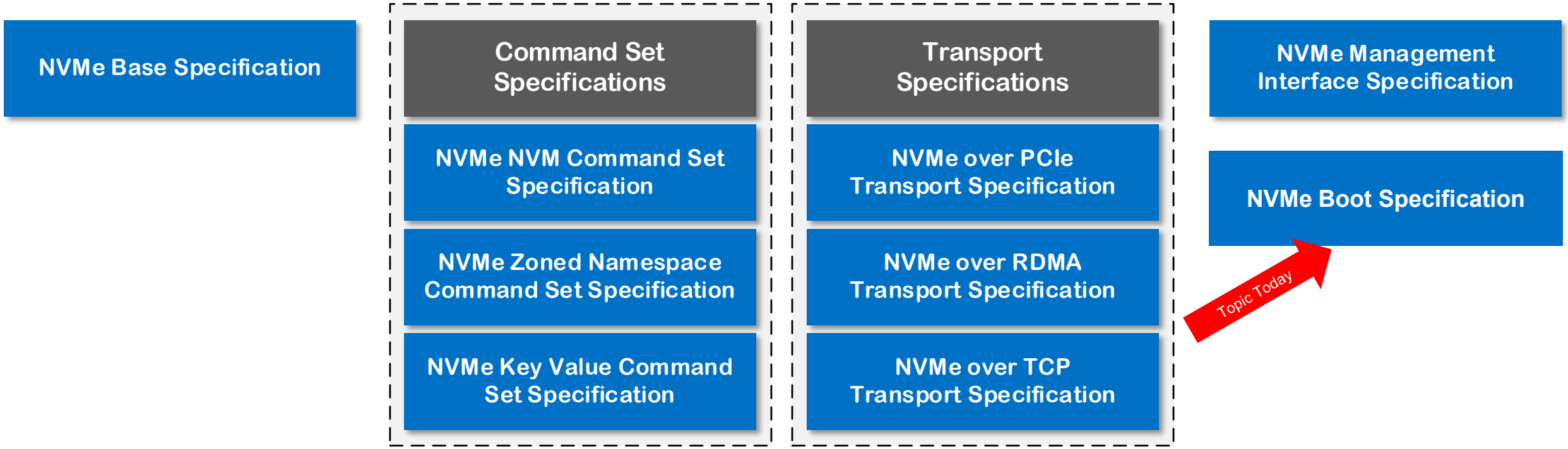
Q&A

Contributed Content

The materials in this presentation represent content from a collaboration between NVM Express and the members of the Timberland SIG, comprised of:

Dell Technologies, Hewlett Packard Enterprise, Intel Corporation, NVIDIA Corporation, Red Hat, Inc., SUSE LLC, VMware, Inc.

NVMe® 2.0 Family of Specifications



*NVMe 2.0 specifications were released on June 3, 2021 - Refer to nvmexpress.org/specifications/