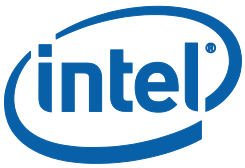


Intel® Optane™ DC Persistent Memory

Quick Start Guide

June 2020

Revision 1.1

Contents

1 Getting Started

Intel® Optane™ DC persistent memory represents a new class of memory and storage technology architected specifically for data center usage. Platforms based on the 2nd Gen Intel® Xeon® Scalable processors may be populated with a combination of DRAM and Intel® Optane™ DC persistent memory. DRAM has the lowest memory latency. Intel® Optane™ DC persistent memory has slightly higher latency but offers affordable capacity and data persistence.

This document will use "PMM" in reference to the Intel® Optane™ DC persistent memory module.

1.1 Basics

Intel® Optane™ DC memory modules support two modes: Memory Mode, which is volatile, and App Direct mode, which is byte addressable persistent memory. The modes determine which capabilities of the Intel persistent memory module are active and available to software. The two operating modes may be configured to run concurrently.

1.1.1 Operating Modes

In **Memory Mode**, the DRAM acts as a cache for the most frequently accessed data, while the Intel® Optane™ DC persistent memory provides large memory capacity. Cache management operations are handled by the Intel® Xeon® Scalable processor's integrated memory controller.

When data is requested from memory, the memory controller first checks the DRAM cache, and if the data is present, the response latency is identical to DRAM. If the data is not in the DRAM cache, it is read from the Intel® Optane™ DC persistent memory with slightly longer latency.

The applications with consistent data retrieval patterns that the memory controller can predict will have a higher cache hit-rate, and should see its performance close to all-DRAM configurations, while workloads with highly-random data access over a wide address range may see some performance difference versus DRAM alone.

Also, data is volatile in Memory Mode; it will not be saved in the event of power loss. Persistence is enabled in the second mode, called App Direct.

In **App Direct Mode**, applications and the Operating System are explicitly aware there are two types of direct load/store memory in the platform and can direct which type of data read or write is suitable for DRAM or Intel® Optane™ DC persistent memory.

Operations that require the lowest latency and don't need permanent data storage can be executed on DRAM, such as database "scratch pads". Data that needs to be made persistent or structures that are very large can be routed to

the Intel® Optane™ DC persistent memory. To make data persistent in memory, you must use App Direct Mode.

App Direct mode requires an operating system or virtualization environment enabled with a persistent memory-aware file system. Contact your OS distributor for support details.

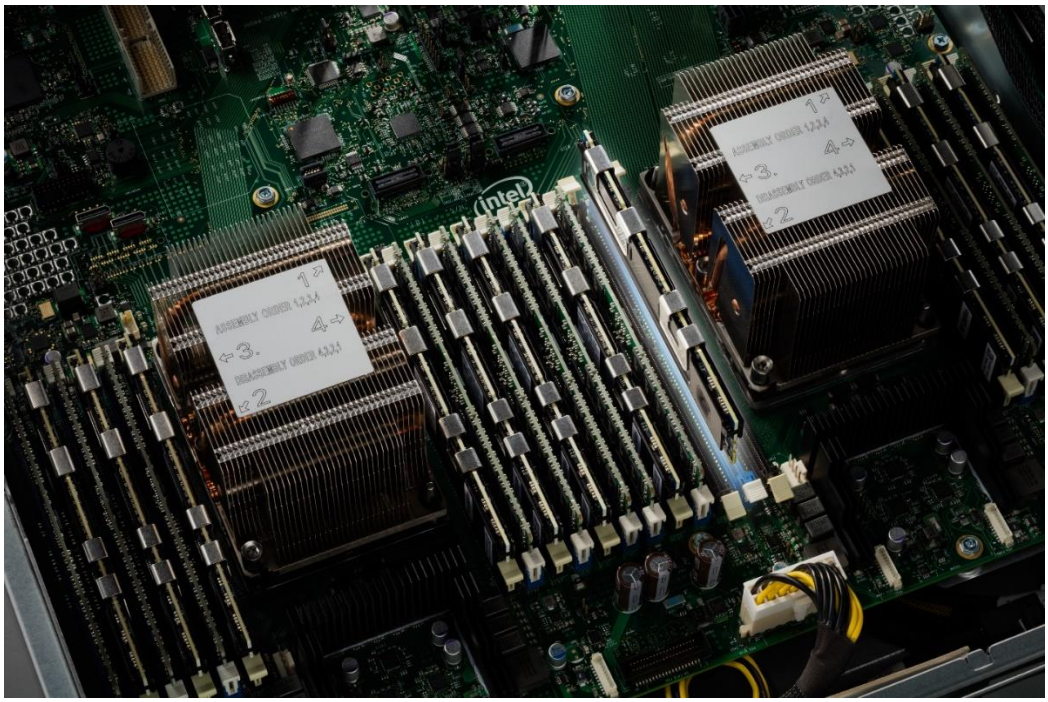
1.1.2 Using Persistent Memory

App Direct Mode may be used as Storage over App Direct, in which case, the driver surfaces a traditional block storage interface transparent to applications, so they do not need to be modified. Storage over App Direct is implemented with copy on write optimization via a block translation table to provide power-fail write atomicity.

Otherwise, applications can be modified to access App Direct capacity with direct load/store mechanisms using a persistent memory aware file system. This completely bypasses the kernel and provides the shortest code path to the persistent memory. To learn more about using and programming for persistent memory, refer to http://pmem.io/.

Warning: A backup of the persistent memory data sets to secondary storage is needed so that the backup may be used to restore the original data sets after a data loss event.

1.2 Installation



1.2.1 System Requirements

Intel® Optane™ DC persistent memory is designed for use with the 2nd Gen Intel® Xeon® Processor Scalable Family and is enabled by a select sub-set of processors, primarily enabled by the Gold and Platinum tier models.

1.2.2 Software and Firmware Requirements

Platform software and Intel® Optane™ DC persistent memory firmware alignment is strongly recommended. The PMM active (operating) firmware version for all installed modules must be the same prior to normal/runtime operations.

The benefits of updating the software and firmware is to keep the platform software current and compatible with other system modules. Updates may include changes such as fixed issues, improved stability and security updates.

1.2.3 Handling, Installation and Removal

Industry standard DIMM practices and procedures must be followed while handling, installing or removing PMM. While taking proper ESD (Electrostatic Discharge) measures, care must be taken not to stack modules together and to handle only by the module edges, never with enough force to flex or bend the module.

1.2.4 DIMM Populations

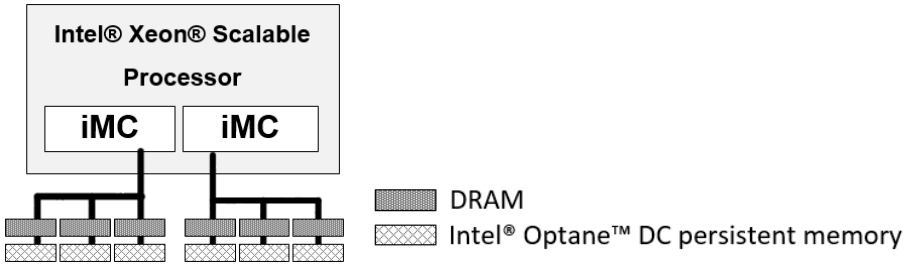
1.2.4.1 Recommended Topologies

The following topologies are recommended per CPU socket. For multiple socket systems, each socket should be populated identically, with the exception of the single PMM per socket topology. The part number must be the same for all installed PMM. Populating both slots on a single channel with PMM is not supported.

Note: DDR channel and DIMM slot nomenclatures may vary depending on platform implementation. Guidance for PMM installation in first position (slot 0) followed by the DIMM (slot 1). When one DIMM is used, it must be populated furthest away from the CPU (slot 0) of the channel.

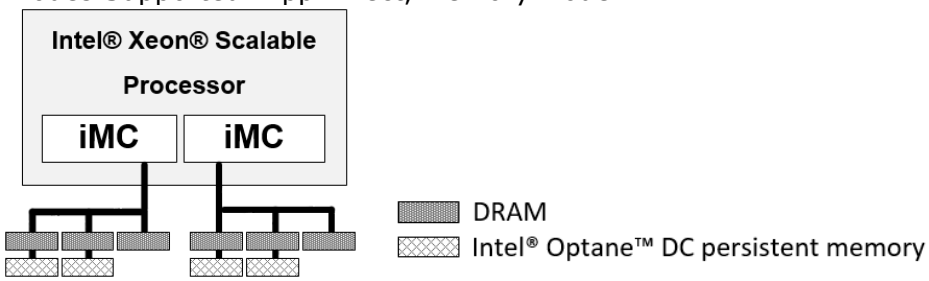
2-2-2 (Six DDR4 DIMMs, Six PMMs)

Modes Supported: App Direct, Memory Mode



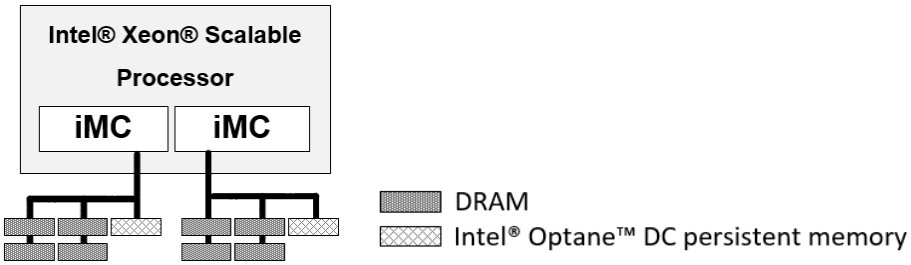
2-2-1 (Six DDR4 DIMMs, Four PMMs)

Modes Supported: App Direct, Memory Mode



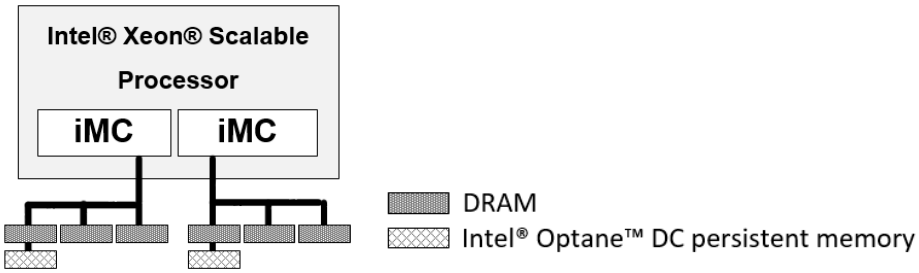
2-2-1 (Eight DDR4 DIMMs, Two PMMs)

Modes Supported: App Direct, Memory Mode

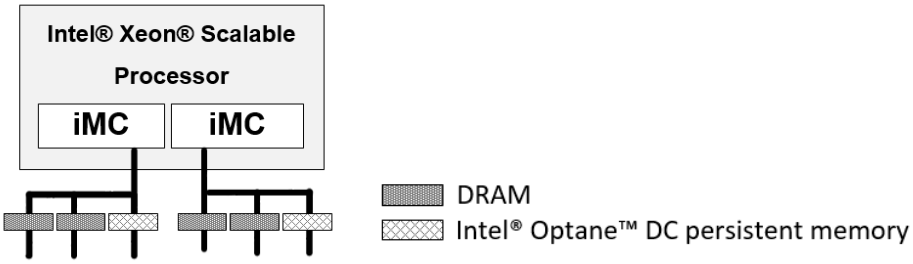


2-1-1 (Six DDR4 DIMMs, Two PMMs)

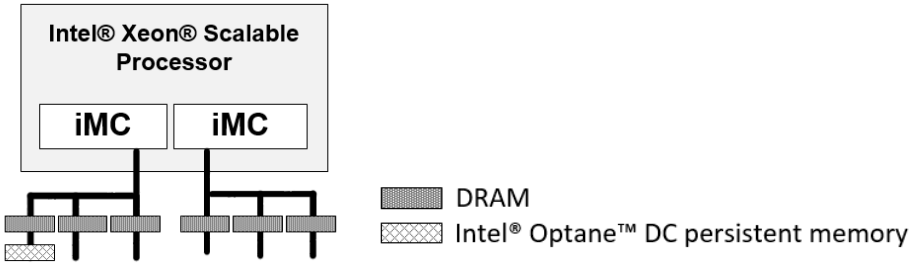
Modes Supported: App Direct, Memory Mode



1-1-1 (Four DDR4 DIMMs, Two PMMs)
Modes Supported: App Direct, Memory Mode



2-1-1 (Six DDR4 DIMMs, One PMM)
Modes Supported: App Direct Only



1.2.5 Documentation and Software Resources

Quick Start Guide: Provision Intel® Optane™ DC Persistent Memory

Non-Volatile Device Control (ndctl) is recommended for creating namespaces for the Linux operating system. It is available for download from GitHub* at https://github.com/pmem/ndctl.

To learn more about persistent memory programming and the Persistent Memory Development Kit (PMDK), visit: http://pmem.io/.

Ipmctl is an open source utility for configuring and managing PMM. This utility was created and is maintained by Intel. It is available for download from GitHub*. at https://github.com/intel/ipmctl.

2 Managing PMM

Intel Ipmctl is a platform utility for configuration and management capabilities including discovery, provisioning, maintenance, and monitoring of the PMMs. It supports the following functionality.

- Discovery
- Configuration
- Firmware management
- Security functionality management
- Health monitoring
- Performance tracking
- Debug and troubleshooting

For more complete information, refer to the ipmctl GitHub* page.

Note: This document provides examples of essential PMM management functionality from the perspective of the OS ipmctl command line utility.

```
ipmctl help
```

Example commands are formatted where [brackets] indicate optional items and (parenthesis) indicate that a value is required. The binary executable name "ipmctl" is omitted for simplicity.

2.1 Discovery

The *Show Topology* command displays both the PMM and DDR4 DRAM DIMMs.

```
show -topology
```

The *Show Device* command displays the PMM discovered in the system.

```
show -dimm
```

To view detailed information about the PMM, add the all option.

```
show -a -dimm
```

The *Show Memory Resources* command displays how the PMM capacity is provisioned at the system level.

```
show -memoryresources
```

2.2 Provisioning

Provisioning Intel® Optane™ DC memory modules is a two-step process. During this process, a goal is specified and stored on the persistent memory modules for the BIOS to read on the next reboot. A goal configures the PMM in Memory Mode, App Direct mode, or both.

The installed DIMM population topology also impacts which modes are supported; refer to the recommended topology figures previously shown.

Warning: Changing the memory configuration is a destructive operation which may result in the loss of data stored in the persistent memory region of the PMM. Therefore, existing data should be backed up to other storage prior to provisioning if it needs to be preserved.

Note: Persistent data is not explicitly cleared during provisioning. Metadata such as file system and SW RAID should be deleted prior to provisioning.

2.2.1 Create Memory Allocation Goal

Use the *Create Memory Allocation Goal* command to create a new configuration request.

Memory Mode 100%

Any percentage of the PMM capacity across sockets can be provisioned in Memory Mode, as described next. In this example, **100** represents the percentage of capacity to be provisioned in Memory Mode.

```
create -goal MemoryMode=100
```

App Direct Mode

PMM can be provisioned in App Direct mode with the interleaving enabled or disabled for the persistent memory.

App Direct (Interleaved)

Interleaving increases the throughput of reads and writes to persistent memory. The default *Create Goal* command creates an interleaved region configured for App Direct mode. The following two commands are equivalent:

```
create -goal

create -goal PersistentMemoryType=AppDirect
```

App Direct (Not Interleaved)

To disable interleave, keep the App Direct capacity contained to each PMM, specify PersistentMemoryType=AppDirectNotInterleaved.

```
create -goal PersistentMemoryType=AppDirectNotInterleaved
```

Concurrent Operating Modes (Mixed Mode)

The following command assigns 60 percent of the available persistent memory capacity to Memory Mode. The remainder is configured as an interleaved set for App Direct mode.

```
create -goal MemoryMode=60
```

The *Show Memory Allocation Goal* command displays a pending goal request prior to reboot. In the case that a goal is not applied correctly, this command returns the configuration request and status. "No result" is returned if the goal was processed successfully during reboot.

```
show -goal
```

The *Delete Memory Allocation Goal* command enables the removal of a pending goal.

```
delete -goal
```

2.2.1.1 Advanced Preferences

User preferences allow for provisioning PMM for more advanced use cases. To view the current preferences, use the *Show Preferences* command.

```
show -preferences
```

To change preferences, use the *Change Preferences* command.

```
set -preferences (Name=Value)
```

The following preferences allow for adjusting the default settings when creating a memory allocation goal.

Preference Name	Description	Options
APPDIRECT_GRANULARITY	The minimum App Direct granularity per PMM.	RECOMMENDED: Use the default recommended App Direct granularity of 32 GiB. 1: Allow 1 GiB App Direct granularity.

2.2.2 Linux* Persistent Memory and Namespace Management

Each OS vendor (OSV) provides native tools for persistent memory and namespace management. The following sections provide Linux examples and references for detailed explanation.

Linux References:

Intel's Quick Start Guide for Linux
https://software.intel.com/en-us/articles/quick-start-guide-configure-intel-optane-dc-persistent-memory-on-linux

The official project documentation for ndctl.
http://pmem.io/ndctl/

The official project location for ndctl.
https://github.com/pmem/ndctl

Basic guide to getting started with persistent memory in Linux.
https://nvdimm.wiki.kernel.org/

2.2.2.1 Linux Command Line (ndctl) Overview

The ndctl is a utility library for managing the libnvdimm (non-volatile memory device) sub-system in the Linux kernel. This library does not provide functionality for memory allocation provisioning, the ipmctl create -goal equivalent functionality.

The ndctl command line provides functions used for persistent memory and namespace management, device list, update firmware and more.

2.2.2.2 Listing Regions

After the memory provisioning (goal processing) reboot, the newly created DIMM-interleave-sets are represented as persistent memory "regions" of App Direct capacity. Regions can be listed with the *List Regions* command.

```
# ndctl list --regions --human
```

Sample output:

```
[
  {
    "dev":"region1",
    "size":"756.00 GiB (811.75 GB)",
    "available_size":"756.00 GiB (811.75 GB)",
    "max_available_extent":"756.00 GiB (811.75 GB)",
    "type":"pmem",
    "iset_id":"0x6f3e7f4888992ccc",
    "persistence_domain":"memory_controller"
  },
  {
    "dev":"region0",
    "size":"756.00 GiB (811.75 GB)",
    "available_size":"756.00 GiB (811.75 GB)",
    "max_available_extent":"756.00 GiB (811.75 GB)",
    "type":"pmem",
    "iset_id":"0x1c9e7f487b952ccc",
    "persistence_domain":"memory_controller"
  }
]
```

To determine which PMMs are part of a particular region, use the *list* command with the *regions* and *dimms* modifiers. Each PMM will be listed within the *mappings* group.

```
# ndctl list --regions --dimms
```

Sample output:


```
{
  "dimms":[
    {
      "dev":"nmem1",
      "id":"8680-a2-1730-000006a5",
      "handle":257,
      "phys_id":45
    },
    {
      "dev":"nmem3",
      "id":"8680-a2-1730-00000632",
      "handle":4353,
      "phys_id":69
    },
    {
      "dev":"nmem0",
      "id":"8680-a2-1730-0000059f",
      "handle":1,
      "phys_id":33
    },
    {
      "dev":"nmem2",
      "id":"8680-a2-1730-000004f3",
      "handle":4097,
      "phys_id":57
    }
  ],
  "regions":[
    {
      "dev":"region5",
      "size":268435456000,
      "available_size":268435456000,
      "max_available_extent":268435456000,

      "type":"pmem",
      "iset_id":8743142107817513552,
      "mappings":[
        {
          "dimm":"nmem3",
          "offset":268435456,
          "length":134217728000,
          "position":1
        },
        {
          "dimm":"nmem2",
          "offset":268435456,
          "length":134217728000,
```

```
          "position":0
        }
      ]
    },
    {
      "dev":"region4",
      "size":268435456000,
      "available_size":0,
      "max_available_extent":0,
      "type":"pmem",
      "iset_id":8015984336272174284,
      "mappings":[
        {
          "dimm":"nmem1",
          "offset":268435456,
          "length":134217728000,
          "position":1
        },
        {
          "dimm":"nmem0",
          "offset":268435456,
          "length":134217728000,
          "position":0
        }
      ]
      "persistence_domain":"memory_controller"
    }
  ]
}
```

2.2.2.3 Creating Namespaces

Regions can then be divided into one or more namespaces in order for the capacity to be surfaced to the Linux operating system and used by applications. Just as an SSD can be carved into namespaces, persistent memory namespaces represent the unit of storage that appears as a device that can be used for I/O. To create a namespace, use the *Create Namespace* command.

```
# ndctl create-namespace
```

The default syntax will create namespace with DAX support, mode=*fsdax.* A DAX aware filesystem may subsequently be created on this namespace to provide optimal application performance. The following example is equivalent syntax to explicitly set the mode and also provides the optional syntax to specify a target region.

```
# ndctl create-namespace --mode fsdax [--region (value)]
```

To create a Storage over App Direct namespace for use as a traditional block storage device with power-fail write atomicity, use sector mode.

```
# ndctl create-namespace --mode sector [--region (value)]
```

Multiple namespaces may be created from the same region. Each namespace is a minimum of 1 GiB (in bytes) and aligns to interleave-width and alignment;

for example, '*--size=*' must align to *interleave-width: 6* and *alignment: 2097152*. Sector namespaces may then be further subdivided by a partition table.

To create multiple namespaces of specific capacity within a single region, use the *Create Namespace* command with the *Size* option.

2.2.2.4 Listing Namespaces

To display namespaces, use the *List* command. Namespaces are shown by default without additional arguments.

```
# ndctl list --human
```

Sample output:

```
[
  {
    "dev":"namespace1.0",
    "mode":"fsdax",
    "map":"dev",
    "size":"744.19 GiB (799.06 GB)",
    "uuid":"56ca2d82-36fc-4bf6-b434-b5315957f5ca",
    "blockdev":"pmem1"
  },
  {
    "dev":"namespace0.0",
    "mode":"fsdax",
    "map":"dev",
    "size":"744.19 GiB (799.06 GB)",
    "uuid":"5ec652fa-3642-4e93-ad76-90d70872f3a3",
    "blockdev":"pmem0"
  }
]
```

2.2.2.5 Delete Configuration

The current configuration can be deleted by first disabling and destroying namespaces and then disabling the active regions. When a namespace is destroyed, the capacity is returned to the underlying region and may be used in the creation of new namespaces.

2.3 Maintenance

This section describes some common maintenance operations.

2.3.1 Firmware Update

The *Show Device Firmware* command using ipmctl.

```
show -firmware
```

This command displays up to two firmware versions per PMM considering that updating the firmware requires a power cycle. The active firmware is the firmware that is currently executing. The "staged" firmware is stored on the PMM and staged for execution after the next power cycle.

If for any reason the firmware update fails to load properly after the power cycle, the firmware will fall back to the previously active version.

Note: Only one firmware is allowed to be staged per power cycle. Therefore, if a version is already staged, power cycle before attempting to stage a new version.

The *Update Firmware* command provides an examine option which validates a firmware image file and may be used prior to loading the image.

```
load -source (firmware file) -examine -dimm
```

Load the firmware image on the PMM using the *Update Firmware* command. It is recommended that all PMM in the system have the same firmware version, although updating the firmware on individual PMM is also supported.

```
load -source (firmware file) -dimm
```

In case the PMM firmware is busy processing a long-operation command request, such as Address Range Scrub (ARS), it will not be possible to update the firmware and an error will be returned.

Check that the new firmware was staged properly using the *Show Device Firmware* command and then power cycle the system. Before using the PMM, verify that the new firmware is now successfully executing on all PMM using the *Show Device Firmware* command.

Linux Workarounds for ARS Operation

To wait for completion of an active Linux ARS long operation(s) and then proceed with the *Update Firmware* command, use the following command:

```
ndctl wait-scrub; ipmctl load -source (firmware file) -dimm
```

Note: ARS may take hours to days depending on the persistent memory configuration and capacity. In such case, it may be preferable to disable the ARS at boot via the kernel boot command line parameter.

The *disable ARS process* command will require adding a kernel boot parameter, rebooting the OS and then the firmware update. After verifying the firmware updated successfully, ensure to remove the added kernel boot command entry to restore the original (default) ARS state setting and execution of ARS operations.

1. Add *Kernel Boot Command Line Entry*:

   ```
   nfit.no_init_ars
   ```

2. Reboot Linux.
3. Verify that an ARS long operation(s) is not currently active.

```
show -d ARSStatus -dimm
```

4. Stage the firmware, using the *Update Firmware* command.

```
load -source (firmware file) -dimm
```

5. Check that the new firmware was staged properly using the *Show Device Firmware* command and then power cycle the system.

```
show -dimm -firmware
```

6. Verify a successful firmware update using the *Show Device Firmware* command.
7. Remove the "nfit.no_init_ars" kernel boot command line entry and reboot the OS in order to restore the default Linux background ARS operation setting.

2.3.2 Adding New PMM

Adding new PMMs to a system that is already configured provides for a few choices depending on the current configuration, the platform settings, and the desired usage of the new PMM.

New PMMs will be used in Memory Mode if the platform supports it. The newly added PMMs can be configured for use in a different mode leaving the existing PMM as they were. Or the new and existing PMM can be configured for use in a different mode together.

Refer to the provisioning section for detailed steps.

2.3.3 Moving PMM

The information that describes how the PMM are provisioned and interleaved together along with the namespace information is physically stored on the PMM. Therefore, it is possible to move PMMs from one system to another and retain the configuration and any data stored in the persistent memory.

However, if the PMM are configured with interleaved App Direct Mode capacity, the PMM must be put into the new system so they can be interleaved in the same way as they were in the previous system. Therefore, it is recommended to install them in the same position relative to the CPU, memory controller, and memory channel.

PMMs that are configured for use in only Memory Mode or App Direct Mode that is not interleaved can be installed in any order in the new system and the configuration and data stored on the PMM will be applied.

2.3.4 Replacing PMM

Any App Direct capacity that is interleaved requires that all of the PMMs in the interleave set (generally all the PMM installed on a specific CPU) are present and functional in order to access the persistent data. Therefore, if it becomes

necessary to replace a PMM, further actions are required to maintain the persistent data before replacing the PMM.

Before removing the PMM, the data must be backed up to other storage and all existing namespaces must be deleted. If the same memory configuration is desired after replacing the PMM, dump the current configuration to a file using the *Dump Memory Allocation Settings* command.

```
dump -destination (file) -system -config
```

Then replace the PMM and load the configuration as a new memory allocation goal using the *Load Memory Allocation Settings* command.

```
load -source (file) -goal
```

Because this creates a new memory allocation goal, a reboot is necessary to map the capacity into the system physical address space. After the reboot, proceed with creating namespaces as desired and restoring the existing data to the new namespaces.

2.4 Debug and Troubleshooting

This section provides information about some common operations that may be required to debug or troubleshoot an issue with a PMM.

2.4.1 Checking PMM Health

The *Show Device* command return data includes the "HealthState" attribute.

```
show -a -dimm
```

For additional health statistics, use the *Show Sensor* command.

```
show -a -sensor
```

Sensors are reported for the following health data.

Sensor Name	Description
Health	The current PMM health as reported in the SMART log
MediaTemperature	The current PMM media temperature in Celsius
ControllerTemperature	The current PMM controller temperature in Celsius
PercentageRemaining	Percentage of spare capacity remaining
LatchedDirtyShutdownCount	The number of shutdowns without notification over the lifetime of the PMM
UnlatchedDirtyShutdownCount	The number of shutdowns without notification over the lifetime of the PMM. This counter is the same as LatchedDirtyShutdownCount, except it will always be incremented on a dirty shutdown even if Latch System Shutdown Status was not enabled
PowerOnTime	The total power-on time over the lifetime of the PMM

Sensor Name	Description
UpTime	The total power-on time since the last power cycle of the PMM
PowerCycles	The number of power cycles over the lifetime of the PMM
FwErrorCount	The total number of firmware error log entries

PMM support alarm thresholds for the percentage value of factory expected life span remaining, the temperature of the Intel® Optane™ Media, and the temperature of the controller. These are customizable using the *Change Sensor Settings* command.

```
set -sensor (sensor) NonCriticalThreshold=(value)
EnabledState=(0|1)
```

If enabled, the PMM will send an alert when the reading exceeds the threshold and the sensor status will change to a *non-critical* state.

2.4.2 Diagnostics

Beyond checking the health information, diagnostics are provided to perform more detailed checks of the PMM using the *Run Diagnostic* command.

```
start -diagnostic
```

2.4.3 Recovering from Issues

As PMM near their lifespan or encounter issues, recovery actions may be required to bring the PMM and the overall system back to a functional state. Refer to the following sections to determine how best to handle different types of issues based on the PMM health as reported by the ipmctl commands.

Issue	How to Determine	What to Do
At Risk	SMART: Health Status = Noncritical show -dimm: HealthState = Minor Failure show -sensor: CurrentState = Noncritical	Backup persistent data and replace the impacted PMM.
Degraded	SMART: Health Status = Critical show -dimm: HealthState = Critical Failure show -sensor: CurrentState = Critical	Backup persistent data and replace the impacted PMM.
Busy	show -dimm: ARSStatus = In progress	Wait until the long running operation completes before trying to use the PMM.
Setup issue	show -dimm: ConfigurationStatus = Failed	Run the PCD diagnostic to determine the cause or re-provision.
Failed	SMART: Health Status = Fatal show -dimm: HealthState = Non-functional	Reset or Power Cycle the platform to recover from a firmware error.

Issue	How to Determine	What to Do
		If "Non-functional" state persists, triage further to determine the correct recovery action.
Failed	SMART: Health Status = Fatal (Persisted status across Power Cycle) show -dimm: HealthState = Non-recoverable error show -sensor: CurrentState =Fatal	Replace the impacted PMM.
Unmanageable	show -dimm: HealthState = Unmanageable	Run the quick health diagnostic to determine the cause.
Missing namespace	show -namespace	Triage further to determine the correct recovery action.

2.4.3.1 At Risk

There are two indications that a PMM may be at risk of failure.

1. The SMART health status changes to *noncritical.* When this happens, an alert is triggered (if enabled) and the health state is reported as *Minor Failure*.
2. An alarm is tripped and the sensor state changes to *Noncritical*.

A *noncritical* SMART health status is indicative that the PMM is reaching its lifespan and should be replaced.

When an alarm threshold is exceeded, the sensor changes to a *Noncritical* status and an alert is triggered. In the case of a temperature alarm, the PMM should be cooled to avoid further remediation such as throttling or shutdown. If the percentage of spare capacity remaining alarm is tripped, the PMM may be reaching its lifespan and should be replaced.

2.4.3.2 Degraded

When a PMM is degraded, performance may be limited and the PMM is at risk of failure. This is indicated in one of the following ways.

1. The SMART health status changes to critical. When this happens, an alert is triggered (if enabled) and the health state is reported as Critical Failure.
2. A temperature sensor changes to *Critical* state.

When the SMART health status changes to *Critical*, the PMM has reached its lifespan or has suffered an internal hardware failure. In either case, the PMM should be immediately replaced to avoid potential loss of persistent data.

If a temperature sensor reports a *Critical* state, the PMM has begun throttling performance to prevent damage to the PMM. In this situation, the PMM should be cooled down immediately to avoid shutdown.

2.4.3.3 Failed

When a PMM fails, any persistent data stored on the PMM is inaccessible and the PMM must be replaced or removed in order to restore overall system functionality. Depending on when or how a PMM fails and whether or not it is still responding, the failed state will be reported in the following way.

1. The SMART health status changes to fatal. When this happens, an alert is triggered (if enabled) and the health state is reported as non-recoverable error.

2.4.3.4 Busy

When a long operation such as an address range scrub is running, some PMM functionality may be limited. Refer to the Discovery section for information on how to retrieve PMM status with the *Show Device* command. Once the long operation is complete, the PMM functionality will be restored.

2.4.3.5 Unmanageable

Manageability is the ability for the host software to manage a PMM. A PMM will report an *Unmanageable* health state when the software is incompatible with the DIMM or the PMM is not responding.

Manageability is determined based on the DIMM vendor and device identifiers and the interface version of the firmware installed on the DIMM. Running the quick health diagnostic will report the reason for the incompatibility.

If the vendor or device information is incorrect or the firmware API version is unavailable, and the DIMM is actually a PMM, it is likely the PMM is not responding. This could occur at runtime, for example, if the PMM is shut down due to a thermal issue. Cooling down the PMM and restarting may resolve the *unmanageable* state.

If the PMM is not responding due to a hardware failure, the PMM will need to be replaced or removed in order to restore overall system functionality.

If the PMM firmware is incompatible with the software, refer to the Installation section to install a compatible version of the firmware and software.

2.4.3.6 Setup Issue

Provisioning is an interaction between the software, the platform firmware, and metadata stored on the PMM during a system reboot. While unlikely, it is possible that an incompatibility exists or a hardware failure occurs that results in a configuration status issue as reported in the Show Device command.

The platform configuration diagnostic may provide more details about the cause of the failure. If the issue is the result of old metadata stored on the PMM, re-provisioning the PMM may fix the issue. If the issue is the result of a PMM failure, refer to the preceding sections to determine the correct actions.

2.4.3.7 Missing Namespace

One or more missing namespaces could be caused by the following:

1. The software is waiting for an address range scrub long operation to complete before surfacing the namespaces to the OS. In this case, wait for the long operation to complete before attempting to use the namespace.
2. One or more PMM contributing capacity to the namespace are security locked with a passphrase. The namespaces will not be surfaced to the OS until the underlying PMM are unlocked or security is disabled by removing the passphrase.
3. One or more PMM contributing capacity to the namespace has encountered an issue and the persistent memory capacity is not available. Refer to the preceding sections to determine the correct action.

2.4.4 Dump Debug Log

Occasionally, PMM encounter an error that requires debug information to be gathered for offline analysis. This can be done using the *Dump Debug Log* command. This debug log is also referred to as the *Firmware Debug Log*.

```
dump -destination (file) -debug -dimm (DimmID)
```

The *Firmware Debug Log* is in binary form and it is not encrypted or signed.

2.4.5 Dump System Support Data

The *Dump Support Data* command provides functionality to create and save platform level support data to a file for off-line analysis by support personnel.

Support data includes system log(s), error log(s), sensor information and diagnostic results.

```
dump -destination (file) -support
```