User Guide

Solidigm Synergy™ Driver

April 2023

Revision 003

Solidigm Confidential

Ordering Information

Contact your local Solidigm sales representative for ordering information.

Revision History

Revision	Description	Date
001	• Initial Release	May 2022
002	• Revised Content	August 2022
003	• Added performance features • Added Uninstall Solidigm Synergy™ Driver • Added Solidigm™ P41 Plus Solid State Drive • DMA Remapping • Windows NVMe features supported • Fast Lane replaces HMC • Quick Start steps for installing the Solidigm Synergy™ Software	April 2023

Contents

Tables

Figures

1 Overview

This document supports customers in enabling, configuring, and evaluating Solidigm Synergy™ software.

1.1 Intended Audience

This document is targeted for parties that have entered into a non-disclosure agreement ("NDA") with Solidigm.

1.2 Solidigm Synergy™ Software Quick Start Steps

This section will define the steps to install Solidigm Synergy software, including Solidigm Synergy™ Driver and Solidigm Synergy™ Tool.

1.2.1 What is Solidigm Synergy™?

Solidigm Synergy is a free software suite for PCs running Microsoft Windows that unlocks innovative new features on Solidigm client SSDs. By closing the gap between the storage device and the rest of the system, Solidigm Synergy enables a more optimized user experience than hardware alone can provide.

There are two components of Solidigm Synergy, both optional but highly recommended:

The **Solidigm Synergy™ Driver** automatically boosts performance by making "under-the-hood" improvements to the connection between SSD and host system.

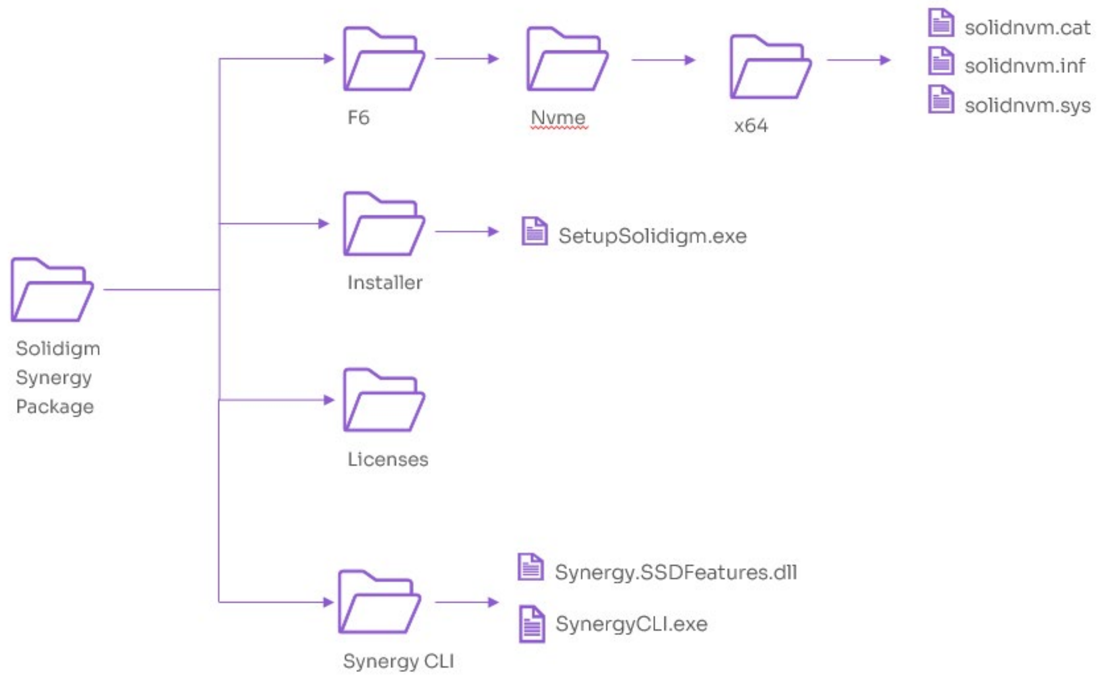
The **Solidigm Synergy™ Toolkit** offers a modern user interface for drive health and information reporting, plus the ability to manually trigger useful functions such as Diagnostic Scan and Secure Erase.

1.2.2 Solidigm Synergy Package

The Solidigm Synergy package structure has the following hierarchy:

- F6
  - NVMe
  - X64
    - Solidnvm.cat
    - Solidnvm.inf
    - Solidnvm.sys
- Installer
  - SetupSolidigm.exe
- License
- Synergy CLI
  - Synergy CLI
  - Synergy SSD feature DLL

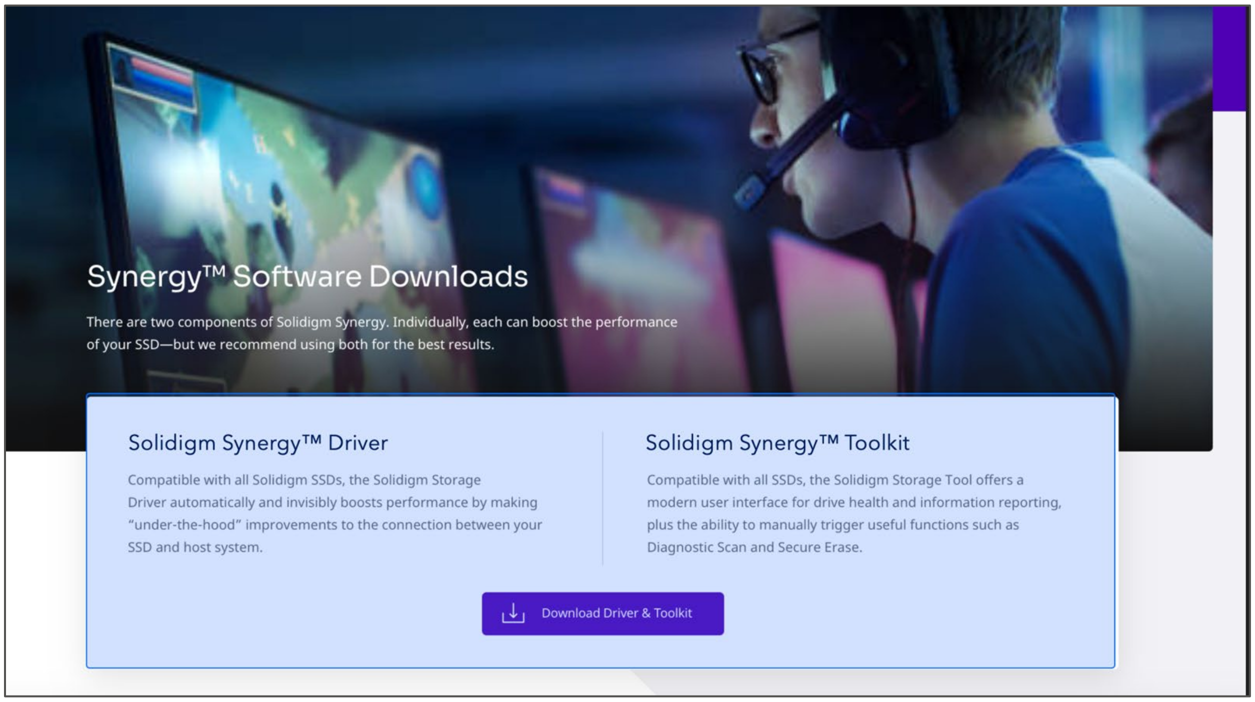
Figure 1: Solidigm Synergy Package Structure



1.2.3 How do I install Solidigm Synergy™?

The Solidigm Synergy Software is available to download at https://www.solidigm.com/synergy. You may choose to install either the Solidigm Synergy Driver or the Solidigm Synergy Toolkit independently, or both.

Figure 2: Free software at https://www.solidigm.com/synergy



Synergy™ Software Downloads

There are two components of Solidigm Synergy. Individually, each can boost the performance of your SSD—but we recommend using both for the best results.

Solidigm Synergy™ Driver

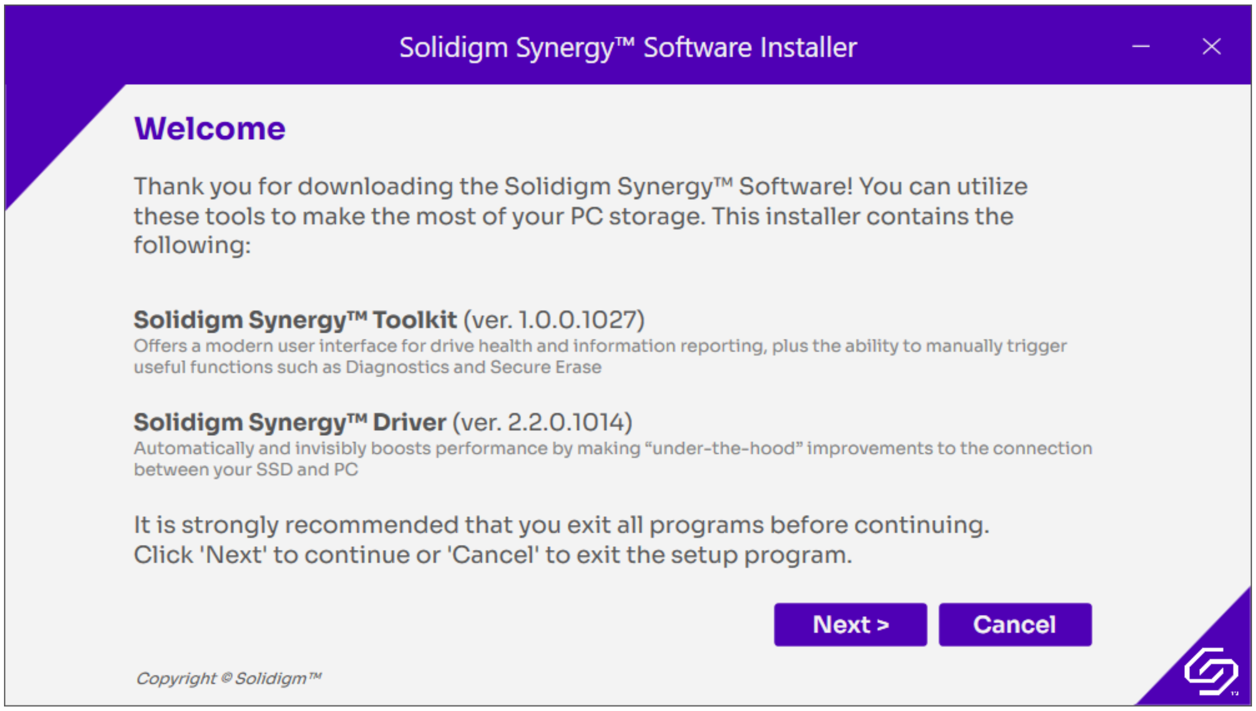
Compatible with all Solidigm SSDs, the Solidigm Storage Driver automatically and invisibly boosts performance by making "under-the-hood" improvements to the connection between your SSD and host system.

Solidigm Synergy™ Toolkit

Compatible with all SSDs, the Solidigm Storage Tool offers a modern user interface for drive health and information reporting, plus the ability to manually trigger useful functions such as Diagnostic Scan and Secure Erase.

Once you have downloaded the installer(s), follow the prompts to install the chosen component(s) on your system.

Figure 3: Welcome to the software installer page, click Next to install



Solidigm Synergy™ Software Installer

Welcome

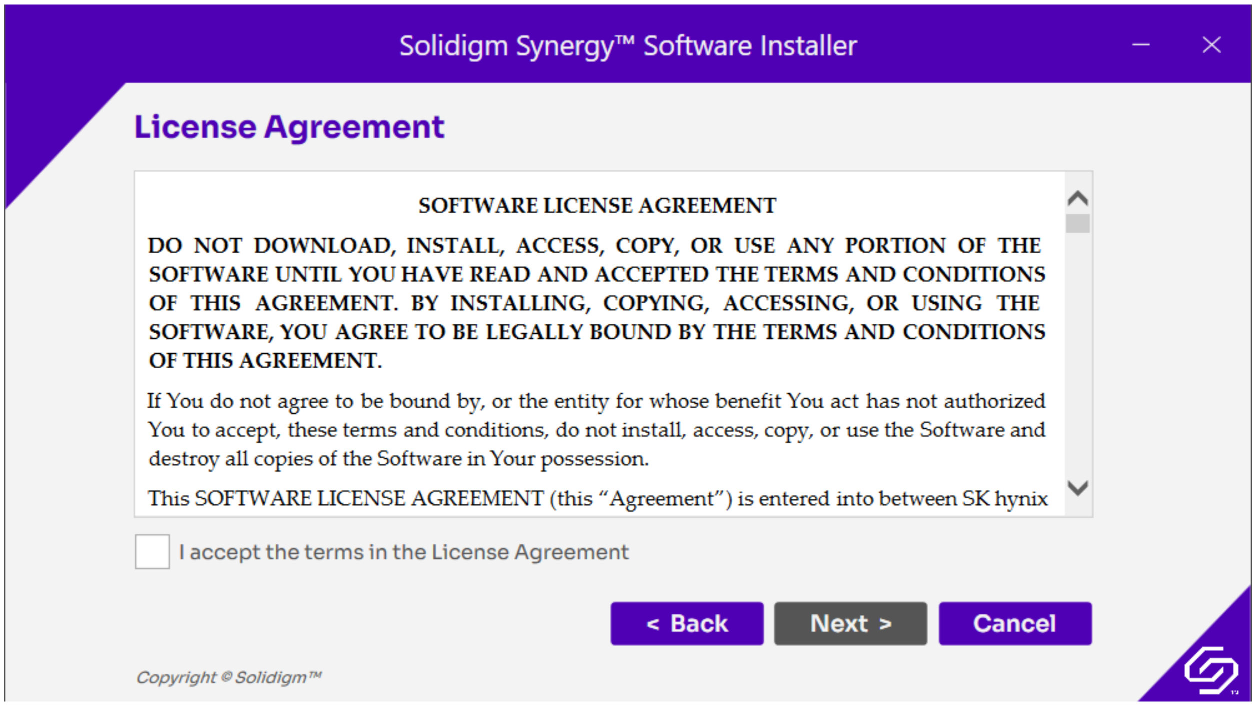
Thank you for downloading the Solidigm Synergy™ Software! You can utilize these tools to make the most of your PC storage. This installer contains the following:

Solidigm Synergy™ Toolkit (ver. 1.0.0.1027)
Offers a modern user interface for drive health and information reporting, plus the ability to manually trigger useful functions such as Diagnostics and Secure Erase

Solidigm Synergy™ Driver (ver. 2.2.0.1014)
Automatically and invisibly boosts performance by making "under-the-hood" improvements to the connection between your SSD and PC

It is strongly recommended that you exit all programs before continuing. Click 'Next' to continue or 'Cancel' to exit the setup program.

Figure 4: License page, click Next to install



Solidigm Synergy™ Software Installer

License Agreement

SOFTWARE LICENSE AGREEMENT

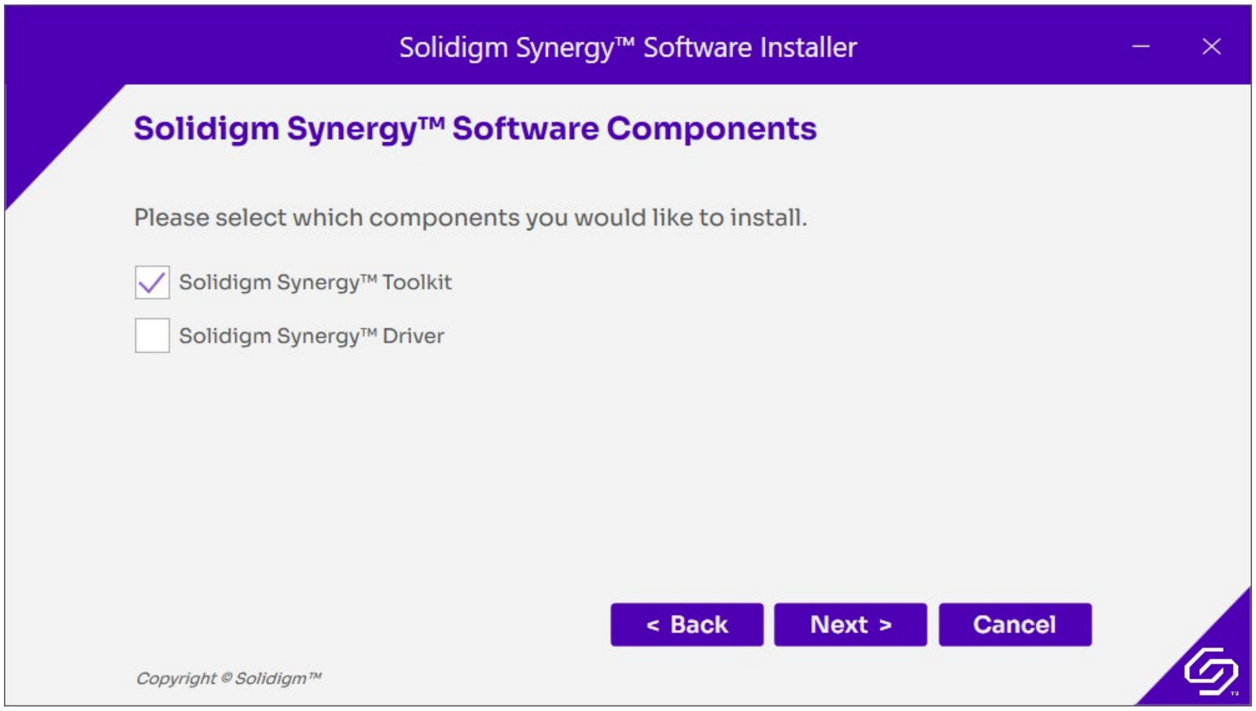
DO NOT DOWNLOAD, INSTALL, ACCESS, COPY, OR USE ANY PORTION OF THE SOFTWARE UNTIL YOU HAVE READ AND ACCEPTED THE TERMS AND CONDITIONS OF THIS AGREEMENT. BY INSTALLING, COPYING, ACCESSING, OR USING THE SOFTWARE, YOU AGREE TO BE LEGALLY BOUND BY THE TERMS AND CONDITIONS OF THIS AGREEMENT.

If You do not agree to be bound by, or the entity for whose benefit You act has not authorized You to accept, these terms and conditions, do not install, access, copy, or use the Software and destroy all copies of the Software in Your possession.

This SOFTWARE LICENSE AGREEMENT (this "Agreement") is entered into between SK hynix

☐ I accept the terms in the License Agreement

Figure 5: After the License page, select the checkboxes to install driver.exe or tool.exe or both



Solidigm Synergy™ Software Components

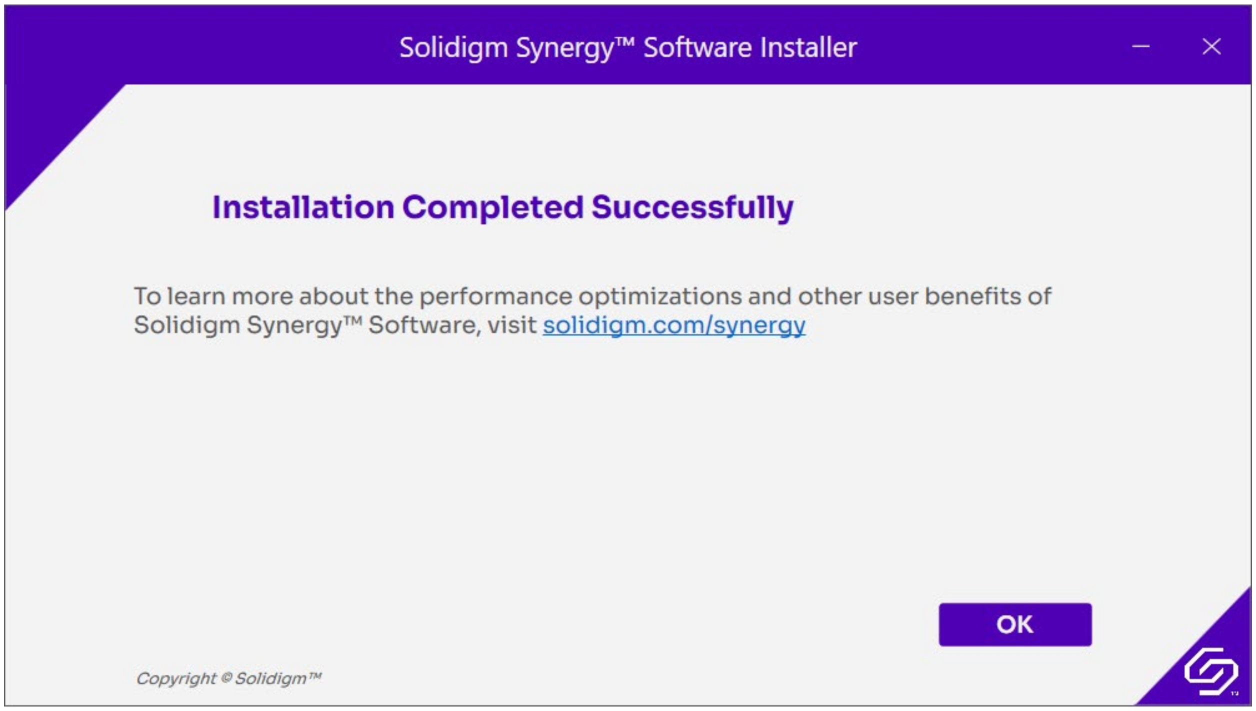
Please select which components you would like to install.

[✓] Solidigm Synergy™ Toolkit

[ ] Solidigm Synergy™ Driver

Copyright © Solidigm™

Figure 6: Installation completion will be displayed from the tool



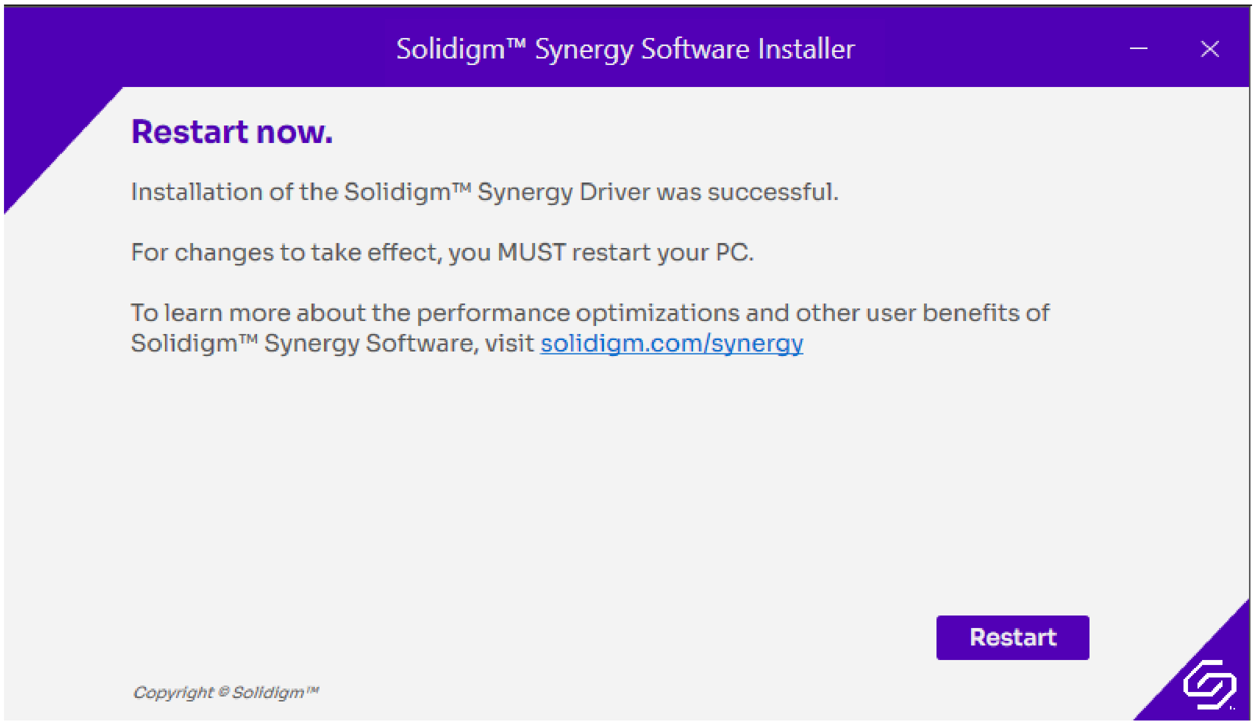
Installation Completed Successfully

To learn more about the performance optimizations and other user benefits of Solidigm Synergy™ Software, visit solidigm.com/synergy

Copyright © Solidigm™

The driver installation process will prompt you to restart your PC.

Figure 7: Restart your PC to complete the driver installation



Solidigm™ Synergy Software Installer

Restart now.

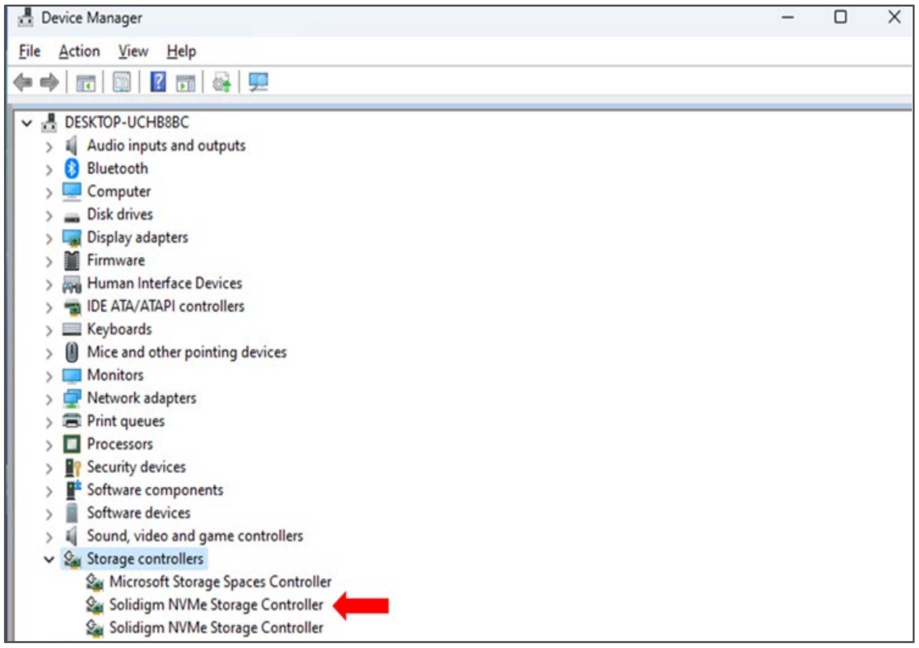
Installation of the Solidigm™ Synergy Driver was successful.

For changes to take effect, you MUST restart your PC.

To learn more about the performance optimizations and other user benefits of Solidigm™ Synergy Software, visit solidigm.com/synergy

Once installation is complete, you can verify the driver is present on your system by opening the Windows Device Manager and checking under the Storage controllers drop-down list for an entry titled Solidigm NVMe Storage Controller. If it's there, you're good to go.

Figure 8: Confirm the driver correctly installed in the Windows Device Manager



```
DESKTOP-UCHB8BC
  Audio inputs and outputs
  Bluetooth
  Computer
  Disk drives
  Display adapters
  Firmware
  Human Interface Devices
  IDE ATA/ATAPI controllers
  Keyboards
  Mice and other pointing devices
  Monitors
  Network adapters
  Print queues
  Processors
  Security devices
  Software components
  Software devices
  Sound, video and game controllers
  Storage controllers
    Microsoft Storage Spaces Controller
    Solidigm NVMe Storage Controller
    Solidigm NVMe Storage Controller
```

1.3 Installer

The installer is a software component used for installation and maintenance of both the Solidigm Synergy™ Driver and Solidigm Synergy™ Toolkit on Windows operating systems. The executable filename is SetupSolidigm.exe.

The following command line parameters are available:

Table 1: Installer Command Line Parameters

Parameters	Description
[-help][-?]	Display this help for command-line options.
[-report][-r] <path>	Changes the default log directory path. The **-accepteula** flag must be provided with the silent flag (**-s**).
[-report] <directory>	Path to directory where log files will be saved.
[-reportfile] <filename>	Path and log filename where installer information will be written to.
[-accepteula]	You must accept the End User License Agreement (EULA) and provide this flag when installing in silent mode.
[-onlyapp][-a]	Install only Solidigm Synergy™ Tool application.
[-onlydriver][-o]	Install only Solidigm Synergy™ Driver.

1.4 Command Line Interface

The Solidigm Command-Line Interface is used for interacting with Solidigm Synergy™ Driver on Windows operating systems. Executable file name: *StorageCli.exe*. The following command line parameters are available:

Table 2: Global Switches

Long-form	Short-form	Switch Value	Value Format	Description
help	h	None		Display this help for command-line options.

Table 3: Mode Switches

Long-form	Short-form	Switch Value	Value Format	Description
information	I	None		Displays help documentation for command-line utility modes, options, usage, examples, and return codes. When used with mode switch (information, Fast Lane), instructions for that mode display. For

Long-form	Short-form	Switch Value	Value Format	Description
				example –Fast Lane –help displays Fast Lane option help.
Rescan	R	None		Forces the system to rescan for hardware changes.
Version	V	None		Displays version information.
Fast Lane		None		Allows to configure Fast Lane feature

Table 4: Mode Switch: Information

Long-form	Short-form	Switch Value	Value Format	Description
comma		None		Lists information in a comma-delimited format view.
Controller	C	Optional	<controller name>	Lists information about the controllers in the storage system.
Disk	D	Optional	<host>-<bus>-<target>-<lun>	Lists information about the disks in the storage system.

Table 5: Mode Switch: Fast Lane

Long-form	Short-form	Switch Value	Value Format	Description
diskId		None	<host>-<bus>-<target>-<lun>	Selects disk.
Enable		None		Enables Fast Lane feature.
Disable		None		Disables Fast Lane feature.

2 Supported Features

Here we will provide an overview of the key features of the Solidigm Synergy™ Driver.

2.1 Fast Lane

Fast Lane, previously known as Host-Managed Caching, is supported on the **Solidigm P41 Plus.**

The purpose of Fast Lane is to improve performance consistency over lifetime of an SSD by keeping the user's most frequently used data on the fastest area of the drive (SLC cache), while the rest of the data resides in slower bulk storage (QLC). It significantly improves utilization of SLC cache within the drive and speeds up access to the user's most frequently used data.

This feature is available in both the Solidigm UI and CLI.

Fast Lane is enabled on supported drives by default, but it can be disabled using the CLI or UI or by adding a registry key.

2.1.1 Via Solidigm Synergy™ Tool CLI

```
set [-help|h] [-output|o (text|nvmxml|json)] -ssd
(Index|SerialNumber|PhysicalPath) FastlaneEnabled = ('true'|'false')
```

FastlaneEnabled = (true|fase)	(Required) Specify whether to enable or disable

true: Enable Fast Lane

false: Disable Fast Lane

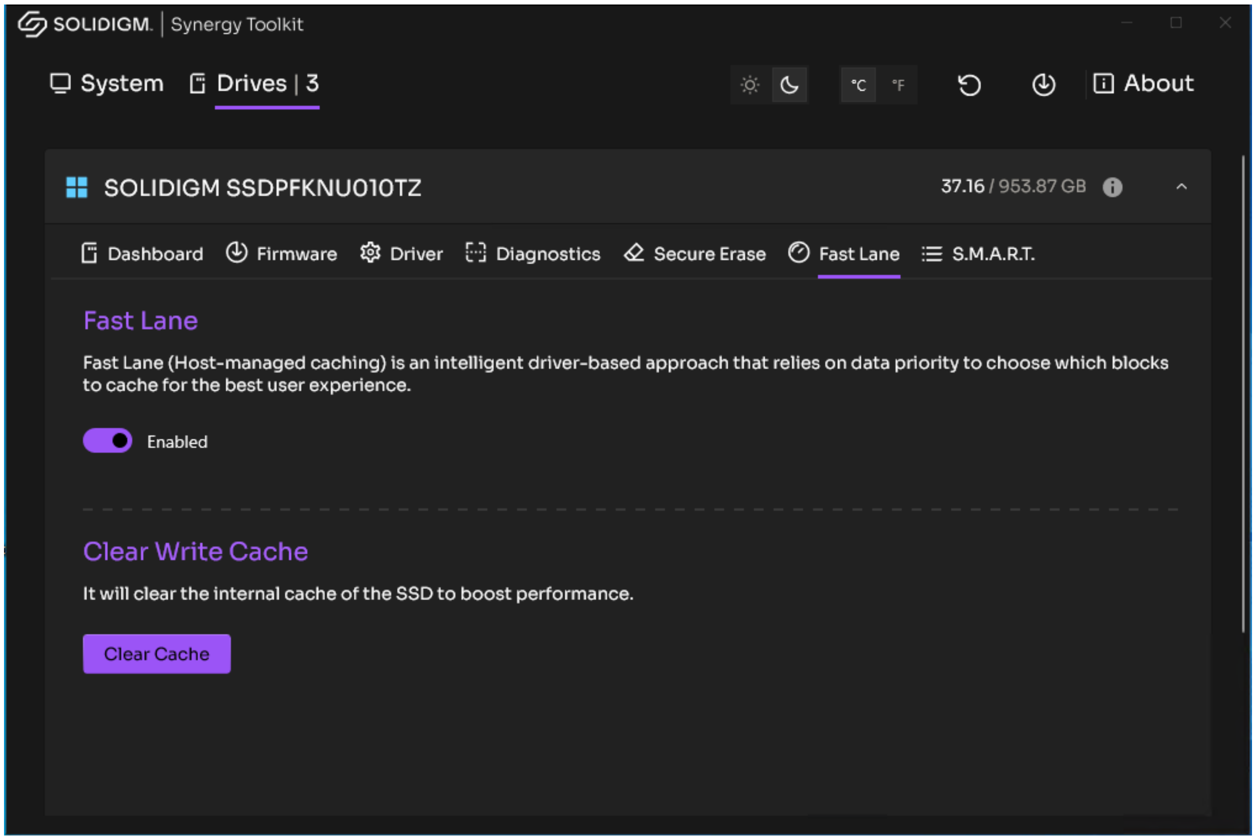
Limitations: To successfully execute this command, the driver and SSD must support this feature. After sending Fast Lane enable/disable, a reboot is necessary to perform operation successfully.

Currently, the Fast Lane feature is enabled by default. In the CLI tool user can also check the status of Fast Lane if enabled or disabled.

2.1.2 Via Solidigm Synergy™ Toolkit GUI

User can enable/disable Fast Lane by using Solidigm Synergy™ Tool by clicking enable/disable button via GUI. It will become visible to user if the drive supports Fast Lane.

Figure 9: Fast Lane options to enable or disable in the GUI



Fast Lane

Fast Lane (Host-managed caching) is an intelligent driver-based approach that relies on data priority to choose which blocks to cache for the best user experience.

Enabled

Clear Write Cache

It will clear the internal cache of the SSD to boost performance.

Clear Cache

2.2 Dynamic Queue Assignment

Dynamic Queue Assignment is a performance feature whose goal is to redirect request's completion to another CPU than originally arrived to improve performance in situation where CPU is a bottleneck.

For some specific traffic like Q32T1 4K disk may be able to process more requests than OS and drivers are able to provide, by redirecting completion driver is gaining more time for processing requests

2.3 Smart Prefetch

Smart Prefetch is a feature introduced to improve sequential read I/O traffic with only one outstanding command (Smart Prefetch) - It takes advantage of the fact that for a Smart Prefetch traffic there is a short period of idle time between each I/O for disk controller. The driver tries to predict next I/O and schedules this operation before actual I/O request is received.

2.4 Host Memory Buffer (HMB)

The Solidigm driver does handle the Host Memory Buffer (HMB) NVMe feature, see NVMe specification. Feature enablement and the maximum number of DRAM pages that can be set should be configurable by OS registry key per individual controller. By default, the feature should be enabled and the number of DRAM pages that can be set shouldn't be limited.

2.5 APST Configuration

The Solidigm driver does allows for setting NVMe APST table values at initialization time based on OS registry configuration for drives that support the APST feature. Registry configuration shall allow for specifying APST table ITPT and ITPS values for individual power states per individual controller.

2.6 Microsoft DirectStorage Optimized

Microsoft DirectStorage for PC was added in Windows 11. The Solidigm Synergy Driver is compliant with the DirectStorage bypass IO path, which should reduce read latency and CPU load when using supported applications.

2.7 DMA Remapping

Windows has a Memory Access Protection feature (Kernel DMA Protection, or DMAr) that enables the OS to protect the system against malicious DMA attacks by DMA capable devices, during the boot process or via devices connected to easily accessible internal/external DMA-capable ports, such as M.2 PCIe slots and Thunderbolt 3, during OS runtime. The Solidigm Synergy Driver is compatible with this feature.

2.8 Windows NVMe Features Supported

2.8.1 NVMe Command Error Logging

The Solidigm Synergy Driver will log following information in the Windows Event Log:

- When any command is completed with non-success status, the following information will be logged:
  - Command set
  - Opcode
  - SCT
  - SC
  - Controller Status (CSTS)

- When there is an asynchronous notification, the following information will be logged:
    - Asynchronous event type
    - Asynchronous event information
    - Log page Identifier
    - Log page binary content

2.8.2 Command Effects Log for Identifying Commands

The Solidigm Synergy Driver supports NVMe passthrough IOCTLs. The IOCTL allows the driver to send NVMe commands directly to the device. Because commands can impact device behaviour, IOs are blocked until the command is completed. Any command is always sent to the device without any validation. This can be improved by getting information from the controller for which commands the IOs must be blocked. Additionally, if the command is not supported by the controller, it can be rejected without sending it to the device.

2.9 Additional Driver Features Supported

The driver supports the following additional features:

- 512B and 4KB LBA support
- RTD3 support
- AER

3 Limitations

Solidigm Synergy™ driver package comes with the following limitations:

3.1 Hardware Limitations

The following NVMe drives are validated on the Solidigm Synergy™ driver:

- Intel SSD 665p
- Intel SSD 670p
- Solidigm™ P41 Plus
- Solidigm™ P44 Pro

3.2 Software Limitations

The following Microsoft Windows versions are supported by the Solidigm Synergy™ driver:

- Windows 11
- Windows 10
- Windows PE

3.3 Configuration Limitations

The Solidigm Synergy™ Driver does not support:

- Volume Management Device (VMD) enabled ports
- RAID configurations

4 Solidigm Synergy™ Driver API

The Solidigm Synergy™ Driver has an API to allow communication with user application.

4.1 Introduction to Storage IOCTL on Windows

This is an introduction for Windows IOCTL and will help user to implement the Solidigm Synergy™ Storage API.

4.1.1 Windows NVMe IOCTLs Support

Solidigm Synergy™ Driver implements following IOCTLs:

- IOCTL_STORAGE_QUERY_PROPERTY
- IOCTL_STORAGE_PROTOCOL_COMMAND
- IOCTL_STORAGE_SET_TEMPERATURE_THRESHOLD
- IOCTL_STORAGE_SET_PROPERTY
- IOCTL_STORAGE_FIRMWARE_GET_INFO
- IOCTL_STORAGE_FIRMWARE_DOWNLOAD
- IOCTL_STORAGE_FIRMWARE_ACTIVATE

4.1.2 DeviceIOControl

The main function is to send control codes to the device driver.

Table 6: DeviceIOControl Parameters

Parameters	Description
hDevice	A handle to the device on which the operation is to be performed.
dwIoControlCode	For storage device, this field will always be IOCTL_SCSI_MINIPORT.
lpInBuffer	A pointer to the input buffer that contains the data required to perform the operation.
nInBufferSize	The size of the input buffer, in bytes.
lpOutBuffer	The same value as the input buffer.
nOutBufferSize	The size of the output buffer, in bytes.
lpBytesReturned	A pointer to the variable that receives the size of the data stored in the output buffer, in bytes.
lpOverlapped	A pointer to the OVERLAPPED structure (optional).

The function will return a value of 1 if the control code is sent successfully; otherwise, it will return a value of 0. If so, then the Windows* user space function *GetLastError()* can be used to obtain the error code.

More details can be found on MSDN: https://docs.microsoft.com/en-us/windows/win32/api/ioapiset/nf-ioapiset-deviceiocontrol

4.1.3 SRB_IO_Control

Data sent to the device via DeviceIoControl must start with specific header – structure defined by Microsoft – SRB_IO_CONTROL. The following data will be IOCTL implementation specific.

Table 7: SRB_IO_CONTROL Structures

Parameters	Description
hDevice	A handle to the device on which the operation is to be performed.
HeaderLength	Size, in bytes, of header (SRB_IO_CONTROL).
Length	Size, in bytes, of all data sent by IOCTL without header length.
ReturnCode	Field utilized to contain data returned by device.
Timeout	Value, in seconds, in which the request should be executed.
ControlCode	Field to distinguish which operation should be performed.
Signature	8-bit ASCII string, identifies the application-dedicated, target HBA for this request.

More details can be found on MSDN: https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/ntddscsi/ns-ntddscsi-_srb_io_control

Figure 10: SRB_IO_CONTROL Example Code

```
struct ExampleIoctl {
    SRB_IO_CONTROL Header;
    ULONG Data[10];
};

std::string deviceSymbolicName = "\\\\.\\Scsi0:";
HANDLE handle = CreateFile(deviceSymbolicName.c_str(),
    GENERIC_READ | GENERIC_WRITE,
    0,
    NULL,
    OPEN_EXISTING,
    0,
    NULL);

ExampleIoctl ioctl = {};
ioctl.Header.HeaderLength = sizeof(SRB_IO_CONTROL);
ioctl.Header.Length = sizeof(ExampleIoctl) - sizeof(SRB_IO_CONTROL);
ioctl.Header.ReturnCode = 0;
ioctl.Header.Timeout = 10;
ioctl.Header.ControlCode = EXAMPLE_IOCTL_CONTROL_CODE;
memcpy_s(&ioctl.Header.Signature,
    sizeof(ioctl.Header.Signature),
    EXAMPLE_IOCTL_SIGNATURE,
    sizeof(ioctl.Header.Signature));

DWORD bytesReturned = 0;

BOOLEAN success = DeviceIoControl(handle,
    IOCTL_SCSI_MINIPORT,
    (void*)&ioctl,
    sizeof(ioctl),
    (void*)&ioctl,
    sizeof(ioctl),
    &bytesReturned,
    NULL);
```

4.2 NVMe Pass-through

The Solidigm Synergy™ Driver supports NVMe Pass-through channel API to send NVMe commands directly.

Table 8: Supported NVMe Commands

NVMe Command	Opcode
Flush	0x00
Read	0x02

NVMe Command	Opcode
Vendor Specific	0x80-0xFF

Table 9: Supported Admin Commands

Admin Command	Opcode
Get Log Page	0x02
Identify	0x06
Set Features*	0x09
Get Features	0x0A
Device Self-test	0x14
Format NVM (WinPE only)	0x80
Security Send	0x81
Security Receive	0x82
Sanitize (WinPE only)	0x84
Vendor Specific	0xC0-0xFF

Table 10: Supported Set Features Commands

Set Features Command	Feature ID
Power Management	0x02
Temperature Threshold	0x04
APST	0x0C
HTCM	0x10
Non-Operational Power State Config	0x11
Vendor Specific	0xC0-0xFF

Figure 11: NVMe Pass-Through IOCTL Structure

```
struct NVME_PASS_THROUGH_PARAMETERS {
    GENERIC_COMMAND Command;
    BOOLEAN IsIOCommandSet;
    COMPLETION_QUEUE_ENTRY Completion;
    ULONG DataBufferOffset;
    ULONG DataBufferLength;
    ULONG Reserved[10];
};

struct NVME_IOCTL_PASS_THROUGH {
    SRB_IO_CONTROL Header;
    UCHAR Version;
    UCHAR PathID;
    UCHAR TargetID;
    UCHAR Lun;
    NVME_PASS_THROUGH_PARAMETERS Parameters;
};

#define INTELNVM_SIGNATURE          "IntelNvm"

#define IOCTL_NVME_PASS_THROUGH     CTL_CODE(0xF000, 0xA02, METHOD_BUFFERED,
FILE_ANY_ACCESS)

#define NVME_PASS_THROUGH_VERSION   1
```

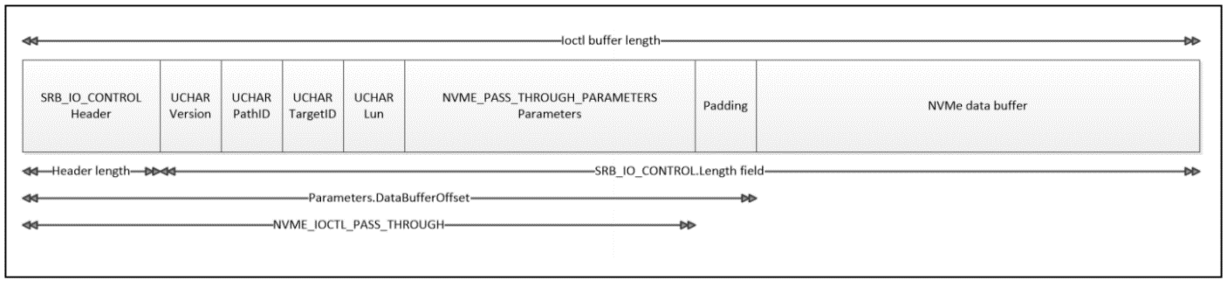


Table 11: NVMe Pass-Through Headers

Header	Description
Header	Filled accordingly to previous chapter
Version	This field should contain the version of the Solidigm Synergy™ Driver Pass-through API. Reserved for marking future changes and should be filled with NVME_PASS_THROUGH_VERSION value
PathID	This field is not used currently. Reserved for future development. Should be set to 0.
TargetID	This field is not used currently. Reserved for development. Should be set to 0.
Lun	This field is not used currently. Reserved for future development. Should be set to 0.

Header	Description
Parameters	This structure contains NVMe Pass-through specific data.
Command	Standard data structure defined by the NVM Express specification. Contains data that will be directly put into the Submission Queue of the NVMe drive. For more information refer to NVMe specification. API heavily relies on those fields: • Command DWORD 0 Opcode (OPC) – This field contains operation code. Refer NVMe specification. Examples of Admin Command Set opcodes are given in NVMe specification and for NVM Command Set. • Namespace identifier (NSID) – This field specifies namespace ID for command, on which it should be executed on. Refer NVMe* specification. Command DWORD 10 – 15 – those fields contain command specific data that will be directly used by the controller when executing command. Refer NVMe specification. For information about standard commands refer NVMe specification or the disk technical document for vendor specific commands.
IsIOCommandSet	This field differentiates between admin command set and IO command set (NVM command set). It determines type of queue that the command will be submitted to. If field is set to TRUE (1) command will be sent to IO Submission Queue. If FALSE (0) will be sent to Admin Submission Queue.
Completion	This field is used for generic output data from command. It is 16 bytes in size, the entry of Completion Queue. Refer NVMe specification for information. Take under consideration that some NVMe commands return the output in DWORD 0 of the Completion Queue Entry field. For more information about standard commands completions refer NVMe specification or disk technical document for vendor specific commands.
DataBufferOffset	This field should contain offset, in bytes, from the beginning of SRB_IO_CONTROL structure to data buffer that will be used for input/output for the command (if needed). According to the NVMe specification data buffer must be DWORD aligned, so offset might not be equal size of NVME_IOCTL_PASS_THROUGH structure. Padding might be 1, 2 or 3 bytes to achieve required alignment.
DataBufferLength	This field should contain size of data buffer, in bytes, which will be used for input/output of command (if needed).

Figure 12: NVMe Pass-Through Example Code

```
// allocate memory
ULONG offset = ((sizeof(NVME_IOCTL_PASS_THROUGH) - 1) / sizeof(DWORD) + 1) *
sizeof(DWORD);
ULONG ioctlBufferSize = offset + sizeof(ADMIN_IDENTIFY_CONTROLLER_DATA);
BYTE* ioctlBuffer = new BYTE[ioctlBufferSize];
memset(ioctlBuffer, 0, ioctlBufferSize);

// prepare header
auto & srbHeader = *reinterpret_cast<SRB_IO_CONTROL*>(ioctlBuffer);
srbHeader.HeaderLength = sizeof(SRB_IO_CONTROL);
srbHeader.ControlCode = IOCTL_NVME_PASS_THROUGH;
srbHeader.Length = ioctlBufferSize - sizeof(SRB_IO_CONTROL);
srbHeader.Timeout = 10;     // example timeout value
memcpy_s(&srbHeader.Signature, sizeof(srbHeader.Signature), INTELNVM_SIGNATURE,
sizeof(srbHeader.Signature));

// Fill version
auto& ioctl = *reinterpret_cast<NVME_IOCTL_PASS_THROUGH*>(ioctlBuffer);
ioctl.Version = NVME_PASS_THROUGH_VERSION;

// Fill paramteters
auto& parameters = *reinterpret_cast<NVME_PASS_THROUGH_PARAMETERS*>(&ioctl.Parameters);
parameters.Command.CDW0.OPC.Raw = ADMIN_COMMAND_IDENTIFY;
parameters.Command.NSID = 0;
parameters.Command.CDW10 = ADMIN_IDENTIFY_CNS_CONTROLLER;
parameters.DataBufferLength = sizeof(ADMIN_IDENTIFY_CONTROLLER_DATA);
parameters.DataBufferOffset = offset;

// open device handle
std::string deviceSymbolicName = "\\\\.\\Scsi0:";
HANDLE driverHandle = CreateFile(deviceSymbolicName.c_str(), GENERIC_READ |
GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);

// send the IOCTL
DWORD bytes = 0;
auto status = DeviceIoControl(driverHandle, IOCTL_SCSI_MINIPORT, ioctlBuffer,
ioctlBufferSize, ioctlBuffer, ioctlBufferSize, &bytes, NULL);

// get result
auto identifyData = *reinterpret_cast<ADMIN_IDENTIFY_CONTROLLER_DATA*>(ioctlBuffer +
parameters.DataBufferOffset);

// cleanup
delete[] ioctlBuffer;
```

4.3 AER

The Solidigm Synergy™ Driver provides an API to allow using Asynchronous Event Notifications reported by NVMe drivers in user mode applications.

4.3.1 Capabilities

- Allows receiving asynchronous event notifications instantly when they are reported by drive.
- Registration for specific drives and events from specific log pages.
- Data about event's purpose are stored for each registered Event Object.

4.3.2 Limitations

- Up to 10 events objects per drive can be registered at the same time.

4.3.3 NVME_IOCTL_REGISTER_AER

This IOCTL is used to register or unregister for AENs from specific drive. To unregister an event, use this IOCTL to overwrite old AEN slot with *eventMask* equal to 0. This will be interpreted by driver as request to remove the event slot with matching *eventName*.

Figure 1313: NVME_IOCTL_REGISTER_AER Structure

```
#define INTELNVM_SIGNATURE "IntelNvm"
#define IOCTL_NVME_REGISTER_AER         CTL_CODE(0xF000u, 0xC00, METHOD_BUFFERED,
FILE_ANY_ACCESS)

#define AEN_MAX_EVENT_NAME_LENGTH       32

typedef struct _NVME_REGISTER_SHARED_EVENT {
    U8  eventName[AEN_MAX_EVENT_NAME_LENGTH];
    U8  reserved;
    U64 eventMask;
} NVME_REGISTER_SHARED_EVENT;

struct NVME_IOCTL_REGISTER_AER {
    SRB_IO_CONTROL Header;
    NVME_REGISTER_SHARED_EVENT EventData;
};
```

Table 12: NVME_IOCTL_REGISTER_AER Headers

Header	Description
Header	Filled accordingly to previous chapter.
EventData	This structure contains event specific data.
eventName	32-byte array of ASCII characters (only values 0-127 will be processed correctly). Name will be used by driver to obtain Event Object handle. Keep in mind: It's not equal to the array used in CreateEvent function.
reserved	Field reserved for NULL termination of eventName.

Header	Description
eventMask	64-bit mask used to mark what kind of events from the drive should be signaled by Event Object. Currently only the first 2 bits have assigned meaning: • Bit 0 – Set event when one of the errors from the Log Page Error Information will be signaled by drive. • Bit 1 – Set event when one of the warnings from the Log Page SMART / Health Information will be signaled by drive.

4.3.4 NVME_IOCTL_GET_AER_DATA

Data from AER Completion DW0 (containing information about cause of event's occurrence) is stored in a separate queue for each event object registered in driver. This IOCTL allows to pop this information to evaluate, what action should be performed by application. Current implementation of the driver can pop no more than one Completion DW0 per IOCTL, but user mode application should be ready for receiving more data. The application should try to pop Completion DW0 until IOCTL returns 0 in *CompletionsCount* field. Completions are stored in circular buffer with ten slots and the oldest entry is overwritten by new if buffer is full. The oldest entry is always returned first.

Figure 14: NVME_IOCTL_GET_AER_DATA Structure

```
#define INTELNVM_SIGNATURE "IntelNvm"
#define IOCTL_NVME_GET_AER_DATA             CTL_CODE(0xF000u, 0xE00, METHOD_BUFFERED,
FILE_ANY_ACCESS)
#define AEN_MAX_EVENT_NAME_LENGTH           32
#define NVME_GET_AER_DATA_MAX_COMPLETIONS   10

struct ADMIN_ASYNCHRONOUS_EVENT_REQUEST_COMPLETION_DW0 {
    union {
        struct {
            U32   AsynchronousEventType : 3;
            U32   Reserved1 : 5;
            U32   AsynchronousEventInformation : 8;
            U32   AssociatedLogPage : 8;
            U32   Reserved2 : 8;
        };
        U32 Raw;
    };
};

struct NVME_AER_DATA {
    U8   eventName[AEN_MAX_EVENT_NAME_LENGTH];
    U8   reserved;
    ADMIN_ASYNCHRONOUS_EVENT_REQUEST_COMPLETION_DW0
Completions[NVME_GET_AER_DATA_MAX_COMPLETIONS];
    UINT32 CompletionsCount;
};

struct NVME_IOCTL_GET_AER_DATA {
    SRB_IO_CONTROL Header;
    NVME_AER_DATA Data;
};
```

Table 13: NVME_IOCTL_GET_AER_DATA Headers

Header	Description
Header	Filled accordingly to previous chapter.
Data	This structure contains event specific data.
eventName	32-byte array of ASCII characters. It should match event name given in IOCTL_NVME_REGISTER_AER to be able to pop correct data
reserved	Field reserved for NULL termination of eventName.
Completions	Array of Completion DW0 structs. It corresponds to value of DW0 from completion of Asynchronous Event Request and it should be interpreted based on NVMe specification.
CompletionsCount	Number of elements returned in Completions field. In current implementation it can only take values 0 or 1.

Example:

Proposed algorithm of registration that should be applied for every drive from which we want to receive notification:

1. Choose name for event. We suggest obtaining it from GUID generated by Windows; that will make the name unique and will exclude possible errors while trying to create new event with name already in use.
2. Create Event Object using WINAPI *CreateEvent* function. In parameter *lpName* enter event's name with added prefix "Global\" – to make Event Object available for driver. Keep returned event handle.
3. Use IOCTL_NVME_REGISTER_AER to pass name of created event (without "Global\" prefix).

```
HANDLE registerForNotifications(HANDLE scsiPortHandle, PTL ptl, string * eventNameOutPtr)
{
    string eventName = getUniqueName();
    string eventNameWithPrefix = string("Global\\") + eventName;
    HANDLE eventHandle = CreateEvent(NULL, TRUE, FALSE, eventNameWithPrefix.c_str());

    if (eventHandle == NULL) {
        return NULL;
    }

    NVME_IOCTL_REGISTER_AER ioctl = {};
    ioctl.Header.HeaderLength = sizeof(SRB_IO_CONTROL);
    ioctl.Header.Length = sizeof(NVME_IOCTL_REGISTER_AER) - sizeof(SRB_IO_CONTROL);
    ioctl.Header.ControlCode = IOCTL_NVME_REGISTER_AER;
    ioctl.Header.Timeout = 10;
    RtlMoveMemory(ioctl.Header.Signature, INTELNVM_SIGNATURE, 8);

    ioctl.EventData.eventMask = static_cast<U64>((1 << AE_TYPE_ERROR_STATUS) | (1 <<
AE_TYPE_SMART_HEALTH_STATUS));

    memcpy_s((char*)&ioctl.EventData.eventName, sizeof(ioctl.EventData.eventName),
eventName.c_str(), eventName.length());

    DWORD bytesReturned = 0;
    if (DeviceIoControl(scsiPortHandle, IOCTL_SCSI_MINIPORT, &ioctl, sizeof(ioctl),
&ioctl, sizeof(ioctl), &bytesReturned, FALSE) == FALSE) {
        CloseHandle(eventHandle);
        return NULL;
    }
    *eventNameOutPtr = eventName;
    return eventHandle;
}


void registerEvents(std::vector<Disk>& nvmeDisks) {
    for (auto& disk : nvmeDisks) {

        HANDLE scsiPortHandle = openScsiPortHandle(disk.mScsiControllerIndex);

        if (scsiPortHandle == INVALID_HANDLE_VALUE) {
            continue;
        }

        string eventName{};
        HANDLE eventHandle = registerForNotifications(scsiPortHandle, disk.mPtl,
&eventName);
        if (eventHandle == NULL) {
            printf("Event registration process have failed for NVMe disk with % s!\n",
disk.mPtl.toString().c_str());
        }
        else {
            EventInfo info{ eventHandle, eventName, disk, disk.mScsiControllerIndex };
            pushToEventTab(info);
        }
        CloseHandle(scsiPortHandle);
    }
}
```

4.3.5 Event Handling

Events reported by a NVMe drive are either persistent or non-persistent. Each category should be handled in a different way (and therefore handling in user space application must differ).

4.3.6 Non-Persistent Events

All events from Error Log Page and most from SMART / Warning Log Page belongs here. The event happens at a specific point in time and when driver receives AER Completion, it checks corresponding Log Page – and that automatically clears corresponding bit. User app won't be able to read that bit before driver, but information about event's purpose is saved in Completion DW0, which can be popped using IOCTL_NVME_GET_AER_DATA.

4.3.7 Persistent Events

It is not clearly specified in NVMe specification, which events should be implemented as persistent. This kind of events persists through time period – until the cause of the event is resolved. The Solidigm Synergy™ Driver masks persistent events for a specific time period – if not, AER Completions would be reported continuously by the drive. After event will be unmasked back, if warning persists in drive, new AER Completion will be received. Therefore, when persistent warning condition takes place on drive, AENs are signaled through API every dozen seconds (until condition will disappear). User mode should poll SMART / Health Log Page (using NVME_IOCTL_PASS_THROUGH) to monitor if bit corresponding to persistent event is still set. Driver considers two warnings to be persistent: "temperature threshold" (bit 1) and "spare capacity below threshold" (bit 0) in SMART / Health Log Page.

Algorithm:

1. Use *WaitForMultipleObjects* function from WINAPI with event handles to create Event Objects.
2. If this function's return value indicates that one of the events was signalized, calculate corresponding event.
3. If that event was registered in the Solidigm Synergy™ Driver to signalize errors/warnings from drive, pop AER_DATA.
4. If Completion DW0 indicates event's purpose, that is not persistent, handle that purpose and skip the rest of algorithm.
5. If persistent event was signalized:
    a. If you do not intend to handle that warning, ignore handling and skip.
    b. If that warning from that drive is not already handled in other thread, start thread with handling. That thread should periodically poll associated Log Page to know when warning condition disappears.

6. If you intend to wait for another event, return to first step.

Figure 1515: Persistent Events Example

```
void waitForEvent() {
    vector<HANDLE > eventHandles = getEventHandleArray();
    DWORD dwWaitResult = WaitForMultipleObjects(static_cast<DWORD>(eventHandles.size()),
eventHandles.data(), FALSE, INFINITE);

    if (dwWaitResult == WAIT_TIMEOUT) {
        printf("One infinity of seconds later...\n");
    }

    else if (dwWaitResult < WAIT_OBJECT_0 || dwWaitResult >= WAIT_OBJECT_0 +
static_cast<DWORD>(eventHandles.size())) {
        printf("WaitForMultipleObjects failed, dwWaitResult=%d, GetLastError() = % d\n",
dwWaitResult, GetLastError ());
    }

    else {
        DWORD index = dwWaitResult - WAIT_OBJECT_0;
        EventInfo& info = getEventInfo(index);
        ResetEvent(info.eventHandle);

        vector<ADMIN_ASYNCHRONOUS_EVENT_REQUEST_COMPLETION_DW0> aerCompletions =
popCompletions(info.scsiPortHandle, info.eventName, info.disk.mPtl, aerCompletions);

        for (auto& dw0 : aerCompletions) {
            if (doWeWantToHandleThat(dw0) && isHandlingThreadAlreadyWorking(dw0)) {
                startHandlingThread(info);
            }
        }
    }
}
```

5 Power Management Support

This chapter describes platform power management features provided by the Solidigm Synergy™ Driver.

5.1 Modern Standby Support

To meet constantly increasing number of systems capable of S0 low power idle, the Solidigm Synergy™ Driver allows NVMe drives to enter DRIPS phase during Modern Standby. To achieve high percentage value of Hardware DRIPS during Modern Standby the driver takes advantage of RTD3 feature. On Modern Standby session entry, aggressive RTD3 policy is set to enable the drive to achieve low power states easily. Based on platform configuration, NVMe drives will reach D3Cold or D3Hot power state. Aggressive RTD3 policy is revoked on Modern Standby session exit. Minimum RTD3 idle wait times for both Modern Standby and S0 states can be adjusted via registry keys.

5.2 Directed PoFx (DFx) Support

Directed PoFx (DFx) is an optional directed power model provided by the Windows run-time power management framework starting from version 3. With DFx, the operating system directs device stacks to enter their appropriate low-power idle states when the system transitions to idle and thereby enables the system to enter low power more reliably. The objective is to make systems more power-efficient and to reduce energy consumption for Windows devices across form factors. DFx currently supports D-state management only. DFx skips any device subtree with an F-state constraint.

5.3 Dynamic APST Feature

Autonomous Power State Transition (APST) is a mechanism for the Solidigm Synergy™ Driver to configure the NVMe controller to automatically transition between power states on certain conditions without software intervention. Additionally, as the driver has the capability to detect if the system is operating on the battery (DC) or plugged in (AC), based on the status of AC versus DC, the driver sets the APST ITPT setting to performance or power mode. If there is a change in status of AC versus DC, the driver will issue Set Feature APST to change the settings to the drive. The driver disables APST when the system is in Performance mode. For each power state, both operational and non-operational, ITPS is set to the last non-operational state.

5.4 Default APST Settings

Table 14 14: Default APST Settings

ITPT Operational Power State	ITPT Non-Operational Power State
60ms	60ms

5.5 Configuring APST ITPT Settings

There are two ITPT settings that can be configured:

- ITPT set for each operational Power State > timeout to the last non-operational state.
- ITPT set for each non-operational Power State > timeout to the last non-operational state.

Both of ITPT settings can be configured separately for:

- AC and DC mode.
- Individual platform power scheme.

5.5.1 Examples of Configuring ITPT Settings

Using Windows* Powercfg Tool

rem Set APST ITPT for operational power states (for AC and DC mode)

powercfg -setAcValueIndex SCHEME_BALANCED SUB_DISK 849ed91c-aad3-40bb-8312-a9ae012d7371 60

powercfg -setDcValueIndex SCHEME_BALANCED SUB_DISK 849ed91c-aad3-40bb-8312-a9ae012d7371 60

rem Set APST ITPT for non-operational power states(for AC and DC mode)

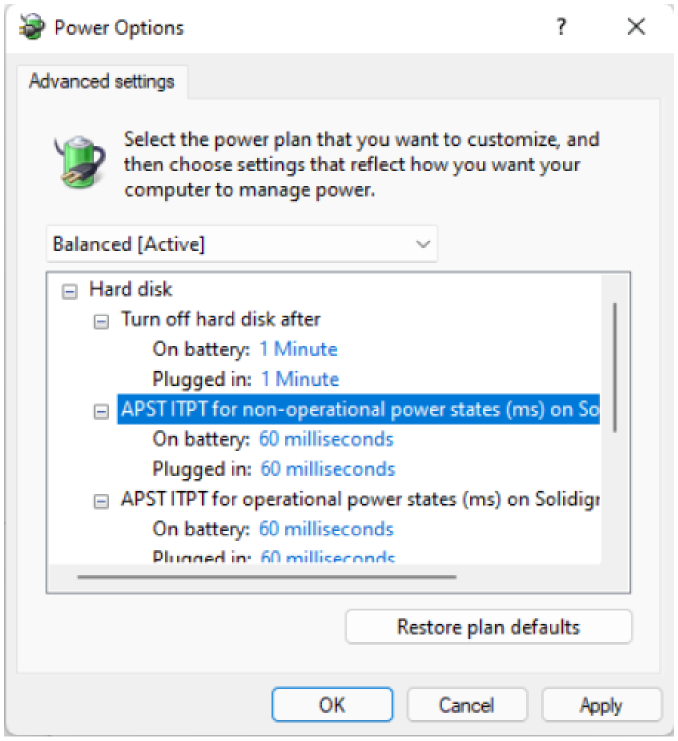
powercfg -setAcValueIndex SCHEME_BALANCED SUB_DISK 6ba22d50-5fd5-4101-a690-96e37e8dd74c 60

powercfg -setDcValueIndex SCHEME_BALANCED SUB_DISK 6ba22d50-5fd5-4101-a690-96e37e8dd74c 60

powercfg -setActive SCHEME_BALANCED

5.5.2 Using Windows Power Options GUI

Figure 10 16: Updated APST Table



Power Options

Advanced settings

Select the power plan that you want to customize, and then choose settings that reflect how you want your computer to manage power.

Balanced [Active]

Hard disk
Turn off hard disk after
On battery: 1 Minute
Plugged in: 1 Minute
APST ITPT for non-operational power states (ms) on So
On battery: 60 milliseconds
Plugged in: 60 milliseconds
APST ITPT for operational power states (ms) on Solidigr
On battery: 60 milliseconds
Plugged in: 60 milliseconds

5.5.3 Using Windows Registry Settings

APST ITPT for operational power states (ms)

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Power\User\PowerSchemes\381b4222-f694-41f0-9685-ff5bb260df2e\0012ee47-9041-4b5d-9b77-535fba8b1442\849ed91c-aad3-40bb-8312-a9ae012d7371]

"ACSettingIndex"=dword:0000003c

"DCSettingIndex"=dword:0000003c

APST ITPT for non-operational power states (ms)

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Power\User\PowerSchemes\381b4222-f694-41f0-9685-ff5bb260df2e\0012ee47-9041-4b5d-9b77-535fba8b1442\6ba22d50-5fd5-4101-a690-96e37e8dd74c]

"ACSettingIndex"=dword:0000003c

"DCSettingIndex"=dword:0000003c

Note: By default, power options may be hidden. To see them you need to change registry key:

```
[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Power\PowerSettings\0012ee47
-9041-4b5d-9b77-535fba8b1442\849ed91c-aad3-40bb-8312-a9ae012d7371]
"Attributes"=dword:00000002
[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Power\PowerSettings\0012ee47
-9041-4b5d-9b77-535fba8b1442\6ba22d50-5fd5-4101-a690-96e37e8dd74c]
"Attributes"=dword:00000002
```

6 Registry

OEM/ODM and end-users can configure several features of the Solidigm Synergy™ Driver using the Windows registry.

Windows registry changes require a system reboot to take an effect.

Root path to all the Solidigm Synergy™ Driver specific registers is:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Solidnvm\Parameters\Devic
e

By default, given registry keys may not be present in registry - in such cases, the key must be created.

6.1 Runtime D3 Settings

Behavior of the Solidigm Synergy™ Driver RTD3 feature can be modified via following registry keys:

Table 1515: Runtime D3 Registry Settings

Registry Value	Description
ModernStandbyAdapterMinIdleTimeoutInMS	DWORD(32) Value specifies the minimum amount of time in milliseconds the power framework must wait to power down device once it is at idle in Modern Standby session. Default: 0x3E8
S0AdapterMinIdleTimeoutInMS	DWORD(32) Value specifies the minimum amount of time in milliseconds the power framework must wait to power down device once it is at idle outside of Modern Standby session (S0) Default: 0xFFFF'FFFF

6.2 APST Settings

Table 1616: APST Registry Settings

Registry Value	Description
NvmeApstEnabled	DWORD(32) Can take values {0,1}. Specifies whether APST feature should be enabled (1) or disabled (0). Default: 0x1

7 Switch or Uninstall Solidigm Synergy™ Driver

7.1 Switch via Windows Device Manager

To switch back to the Microsoft default disk driver after the Solidigm Synergy™ Driver has been installed, complete either of the following options. Options may be grayed out; option 1 is always recommended.

1. Update Driver Option
   In Device Manager, select Solidigm NVMe Storage Controller. Right-click and go to Properties. Select the "Update Driver" option and proceed to "Browse my computer for drivers." Select "Let me pick from a list of available drivers on my computer," then "Standard NVM Express Controller."
2. Rollback Driver
   In Device Manager, select Solidigm Storage Controller. Right-click and go to Properties. Select the "Rollback Driver" option and proceed.

Note: Both operations will only switch the driver on a single-disk controller.

7.2 Uninstalling the Solidigm Synergy™ Driver

In Windows Device Manager, change View to "Devices by driver." Find the Solidigm Synergy™ Driver .inf entry (solidnvm.inf). Right-click on it, select Remove Driver, and hit Remove.

Warning! Do not check the box that says "Attempt to remove the driver for this device" to uninstall or switch the driver for the selected Solidigm NVMe Storage Controller. If the selected storage controller is active on a boot device, it will cause a BSOD loop during the next reboot and successful booting will not be possible.

8 Troubleshooting and Debugging

This chapter explains the most common problem that may occur while using Solidigm products.

8.1 BSOD Issues

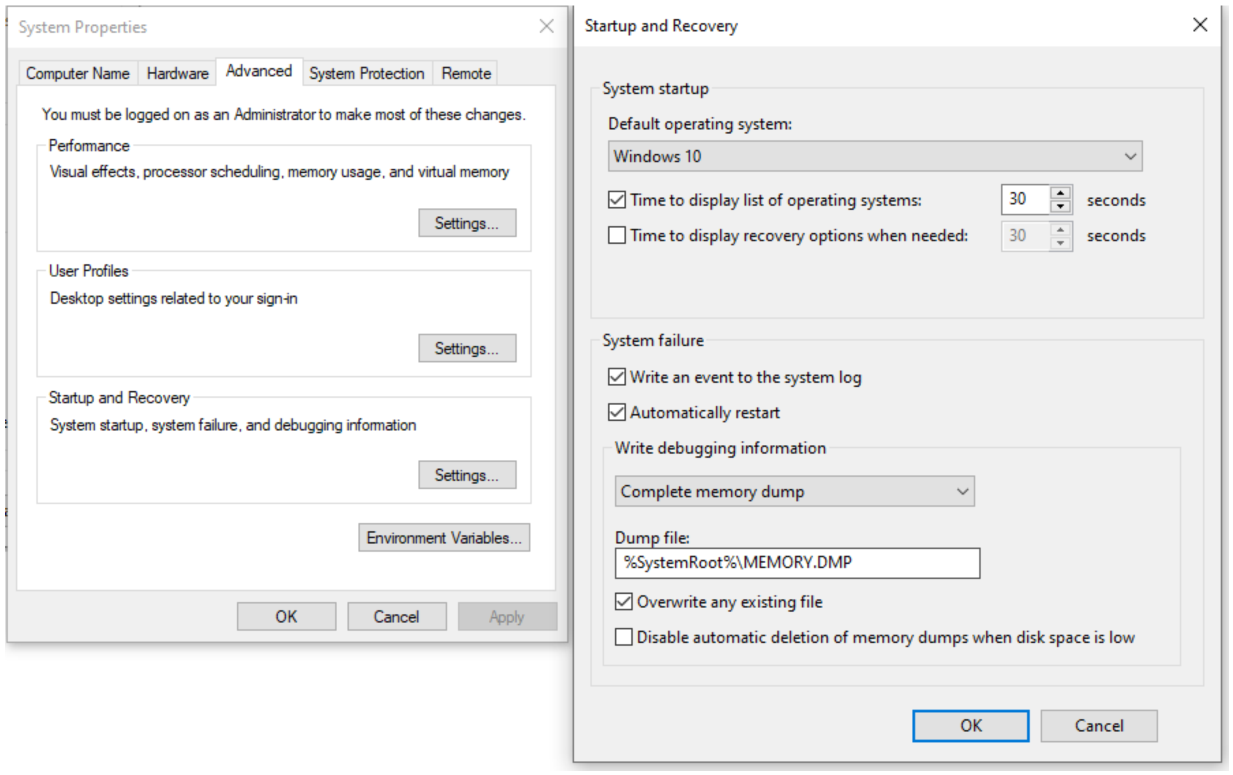
Sometimes a BSOD might occur accidently with different bug check numbers in different scenarios after Solidigm Synergy™ Driver is installed. To identify the problem, a Memory Dump is needed.

There are several ways to save a Memory Dump file.

8.1.1 Automatically Save Memory Dump

If configured properly, the Windows system will save a Memory Dump automatically whenever a BSOD event happens. To get the Memory Dump file, boot the OS after the BSOD event and search in an appropriate path for MEMORY.DMP file.

Figure 1117: System Properties settings for automatic memory dump generation



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

User Profiles
Desktop settings related to your sign-in
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

OK | Cancel | Apply

Startup and Recovery

System startup

Default operating system:
Windows 10

[✓] Time to display list of operating systems: 30 seconds
[ ] Time to display recovery options when needed: 30 seconds

System failure

[✓] Write an event to the system log
[✓] Automatically restart

Write debugging information
Complete memory dump

Dump file:
%SystemRoot%\MEMORY.DMP

[✓] Overwrite any existing file
[ ] Disable automatic deletion of memory dumps when disk space is low

OK | Cancel

8.1.2 Manually Save Memory Dump via WinDBG

There are cases when a BSOD event happens and no MEMORY.DMP file is created, or is it impossible to successfully boot to Windows OS. In such situations, it is recommended to attach a WinDBG application to the faulty system and save a dump file directly from WinDBG.

MSDN articles on how to configure host and target platforms:

- **Via Network**
  https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/setting-up-a-network-debugging-connection
- **Via USB**
  https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/setting-up-a-usb-3-0-debug-cable-connection

If configuration for WinPE is needed, please refer to https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/winpe-debug-apps?view=windows-11

Once the WinDBG application is correctly attached and a debug connection is established, on the next BSOD event WinDBG will break in and a command prompt will be available. Type in the following command to save the BSOD event to a Memory Dump file:

```
kd> .dump /f <path>\Memory.dmp
```

8.1.3 Analyzing Dump Files

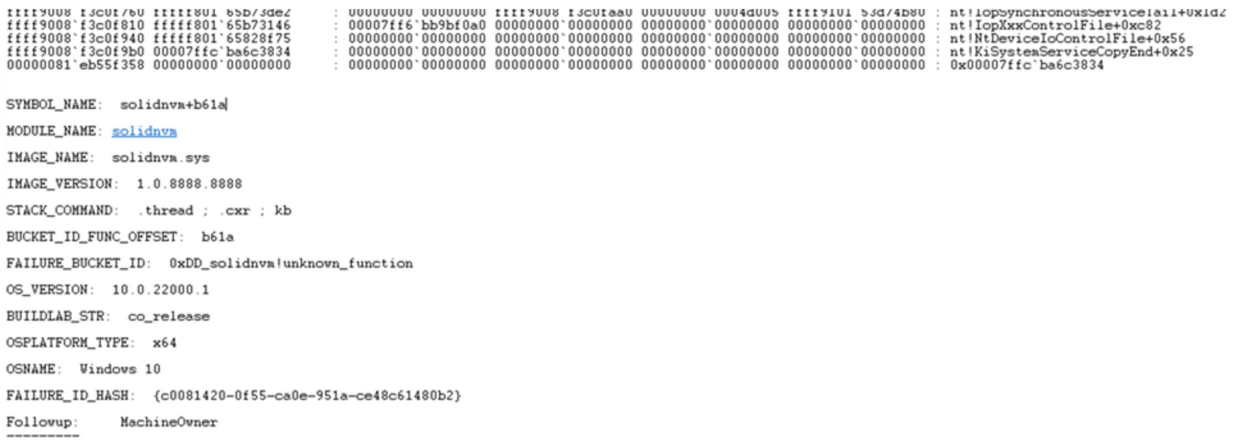
When a BSOD event happens, the Solidigm Synergy™ Driver might or might not be the culprit. To determine the faulty component, open the Memory Dump file with the WinDBG application and run following command:

```
kd>  !analyze -v
```

Based on the command output and investigating MOUDLE_Name field, a final decision can be made whether solidnvm was the root cause of the BSOD event. In such a case, upload the Memory Dump to the Solidigm Support page and seek a Solidigm CE representative for further support.

Figure 1218: Application that pinpoints a failure to the Solidigm Synergy™ Driver

Example Output from WinDB



```
ffff9008`f3c0f760 fffff801`65b73de2     : 00000000`00000000 ffff9008`f3c0faa0 00000000`0004d005 ffff9101`53d74b80 : nt!IopSynchronousServiceTail+0x1d2
ffff9008`f3c0f810 fffff801`65b73146     : 00007ff6`bb9bf0a0 00000000`00000000 00000000`00000000 00000000`00000000 : nt!IopXxxControlFile+0xc82
ffff9008`f3c0f940 fffff801`65828f75     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!NtDeviceIoControlFile+0x56
ffff9008`f3c0f9b0 00007ffc`ba6c3834     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!KiSystemServiceCopyEnd+0x25
00000081`eb55f358 00000000`00000000     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : 0x00007ffc`ba6c3834

SYMBOL_NAME:  solidnvm+b61a

MODULE_NAME: solidnvm

IMAGE_NAME:  solidnvm.sys

IMAGE_VERSION:  1.0.8888.8888

STACK_COMMAND:  .thread ; .cxr ; kb

BUCKET_ID_FUNC_OFFSET:  b61a

FAILURE_BUCKET_ID:  0xDD_solidnvm!unknown_function

OS_VERSION:  10.0.22000.1

BUILDLAB_STR:  co_release

OSPLATFORM_TYPE:  x64

OSNAME:  Windows 10

FAILURE_ID_HASH:  {c0081420-0f55-ca0e-951a-ce48c61480b2}

Followup:     MachineOwner
---------
```