Release Notes

SmartHBA 2100 and SmartRAID 3100 Software/Firmware

Released

October 2020

Revision History

Revision	Revision Date	Details of Change
25	October 2020	SR 2.5.4 Production Release
24	August 2020	SR 2.5.2.2 Production Release with Firmware 3.00
23	March 2020	SR 2.5.2 Production Release with Firmware 2.93
22	March 2020	SR 2.5 Production Release with Firmware 2.66
21	February 2020	SR 2.5.2 Production Release
20	October 2019	SR 2.5 Production Release
19	September 2019	Updated for SR 2.4.8.1 (fw v2.31 Build 0)
18	August 2019	Updated for SR 2.4.8
17	January 2019	SR2.4 Production Release
16	June 2018	SR2.3 Production Release
15	June 2018	Updated for RC Release
14	October 2017	Update supported OSs
13	October 13, 2017	First Production Release
1-12	June 2016-July 2017	Pre-Production Releases.

Contents

1 About This Release

The development release described in this document includes firmware, OS drivers, tools, and host management software for the SmartHBA 2100/SmartRAID 3100 controller solutions from Microsemi.

1.1 Release Identification

The firmware, software, and driver versions for this release are shown in the following table.

Table 1 • Release Summary

Solutions Release	2.5.4
Package Release Date	October 15, 2020
Firmware Version	3.21 B0[1,2] (basecode 06.05.008.000)
UEFI Version	1.3.11.1
Legacy BIOS	1.3.11.3
Driver Versions	Windows SmartPQI: • Windows 2012/2016/2019: 106.190.4.1062 Linux SmartPQI: • RHEL 6/7/8: 1.2.16-040 • SLES 12/15: 1.2.16-040 • Ubuntu 16/18/20: 1.2.16-040 • CentOS 6/7/8: 1.2.16-040 • Debian 9/10: 1.2.16-040 • Oracle Linux 7/8: 1.2.16-040 • Citrix XenServer 7/8: 1.2.16-040 VMware SmartPQI: • VMWare ESXi 6.5/6.7/7.0: 4030.0.101 FreeBSD/Solaris SmartPQI: • FreeBSD 11/12: 4030.0.101 • Solaris 11: 4030.0.101
arcconf/Maxview™	B23821

Note:

1. Downgrading to 1.04 B0 or older builds from this release or prior 1.29 releases may cause the board to not boot or have supercap errors due to an incompatibility in SEEPROMs between this release and prior releases. Refer to the section " *Updating the Controller Firmware* " to downgrade an existing board.
2. If the firmware running on the board is older than 0.01 B594, existing data in the logical volumes must be backed up if it needs to be used after the upgrade. After the upgrade from firmware prior to 0.01 B594, the logical volumes will need to be recreated.
3. Only run the driver on firmware 0.01 build 500 or later.

1.2 Components and Documents Included in this Release

Download the firmware, drivers, host management software, and supporting documentation for your SmartHBA 2100/SmartRAID 3100 controller SmartHBA 2100/SmartRAID 3100 controller and SmartRAID 3100 and SmartRAID 3100 controller solutions from the Microsemi Web site at *https://storage.microsemi.com/en-us/support/start/*

1.3 Files Included in this Release

This release consists of the files listed in the following tables:

Firmware Files

Table 2 • Firmware Files

Component	Description	Pre-Assembly Use	Post-Assembly Use
SmartFWx100.bin	Programmable NOR Flash File Use to program NOR Flash for boards that are already running firmware.		X

Table 3 • Firmware Programming Tools

Tool	Description	Executable
Arcconf romupdate	The command allows to upgrade/downgrade the firmware and BIOS image to the controller.	Refer to *Table 7 • Host Management Utilities* on page 4
maxView firmware upgrade wizard	The firmware upgrade wizard allows to upgrade/downgrade the firmware and BIOS image to one or more controller(s) of same model in the system.	Refer to *Table 7 • Host Management Utilities* on page 4

Driver Files

Table 4 • Windows Storport Miniport SmartPQI Drivers

Package	Drivers	Binary	Version
2012	Server 2019 Server 2016 and Windows 10 Server 2012 SP1, R2 SP1 and Windows 8.1, 8	SmartPqi.sys	x64
		SmartPqi.inf	x64
		Smartpqi.cat	x64

Table 5 • Linux SmartPQI Drivers

Drivers	Version
Red Hat Enterprise Linux/CentOS 8.2, 8.1, 8.0, 7.8, 7.7, 7.6, 7.5[1]	x64
Red Hat Enterprise Linux/CentOS 6.10, 6.9[1]	x64
CentOS 7.4	x64
SuSE Linux Enterprise Server 12[1], SP5, SP4, SP3, SP2	x64
SuSE Linux Enterprise Server 15 SP2, SP1[1]	x64
Oracle Linux 7.6 with UEK5u2 (4.14.35)	x64
Oracle Linux 7.7 with UEK5u2 (4.14.35)	x64
Oracle Linux 7.8 UEK 5u2	x64

Drivers	Version
Oracle Linux 8.0	x64
Oracle Linux 8.1 UEK6	x86
Oracle Linux 8.2 UEK6 U1	x86
Ubuntu 20.04	x86
Ubuntu 18.04.5, 18.04.4, 18.04.3, 18.04.1	x64
Ubuntu 16.04.5, 16.04.4	x64
Debian 10.04	x64
Debian 9.12	x64
Citrix xenServer 8.1, 8.0, 7.6	x64
Fedora 30 (inbox only)	x64

Note: 1. To mitigate against the Spectre Variant 2 vulnerability, the RHEL 6u9/RHEL7u4/RHEL7u5 and SLES11 SP3 and higher drivers have been compiled to avoid the usage of indirect jumps. This method is known as "Retpoline".

Table 6 • FreeBSD, Solaris, and VMware SmartPQI Drivers

Drivers	Version
FreeBSD 12.1, 11.4	x64
Solaris 11.4, 11.3	x64
VMware 6.7 U3/U2/U1, 6.5 U3/U2/U1	x64
VMware 7.0 U1	x64

Host Management Software

Table 7 • Host Management Utilities

Description	OS	Executable
ARCCONF Command Line Utility	Windows x64 Linux x64 VMware 6.5 and above XenServer FreeBSD x64 Solaris x86	See the Arcconf download package for the OS-applicable installation executable.
ARCCONF for UEFI		Included as part of the firmware downloadable image.
maxView Storage Manager	Windows x64 Linux x64 VMware EXSi 6.5 and above	See the maxView Storage Manager download package for the OS-applicable installation executable.

Description	OS	Executable
	XenServer	
maxView vSphere Plugin	VMware 6.5 and above	See the VMware maxView Storage Manager download package for the OS-applicable installation executable.
Boot USB (offline or pre-boot) for ARCCO-NF and maxView Storage Manager	Linux x64	See the maxView BootUSB download package for the .iso file.

2 What is New?

2.1 Features

The following table lists features supported for this release.

Table 8 • Feature Summary

Feature		Supported in this Release	Future Release
UEFI Driver, Boot Support		X	
Legacy Boot Support		X	
Dynamic Power Management		X	
SMR Drive Support	Enumeration, Unrestricted Command Flow-T-hrough	X	
	SATL Translation for HA/HM SMR Management	X	
	Identify All Drive Types	X	
Driver Support	Windows	X	
	Linux	X	
	VMware	X	
	FreeBSD	X	
	Solaris	X	
	OS certification	X	
Out of Band interface selection support of MCTP or PBSI		X	
Flash Support		X	
MCTP BMC Management		X	
Configurable Big Block Cache Bypass		X	
Green Backup Support for SmartRAID		X	
4Kn Support in RAID		X	

2.2 Fixes

2.2.1 Firmware Fixes

2.2.1.1 Fixes and Enhancements for Firmware Release 3.21 B0

This release includes the following fixes and enhancements:

- Enhanced the performance of sequential READ I/Os greater than 256 KiB for cached logical drives.
- Added support for user configurable expander scan duration parameters.
- Improved background consistency check to scan a logical drive more efficiently.
- Added support for Port Discovery Protocol host tool option to support either SGPIO or UBM backplanes.
- Performance drop is observed on certain queue depth for the 4 KB sequential write workload on RAID logical volumes with IOBypass and DDR caching disabled.
  - Root Cause: The coalescing logic was applying a uniform time window in which to wait for available coalescing opportunities. Under low queue depths in which a successful fully coalesced request could not be achieved, this resulted in a uniform maximum performance based on this time window.
  - Fix: Modified the coalescing logic to use the incoming host I/O counts as a metric in which to adjust the coalescing time window to be more or less aggressive in abandoning coalescing opportunities for each RAID volume.
  - Risk: Medium
- Fixed a controller lockup when a cable is pulled from an enclosure while command timeout error handling is also occurring.
  - Root Cause: A race condition between an enclosure cable pull operation and command timeout error handling logic which results in the physical drive being failed can lead to a firmware timer expiring that triggers a controller lockup.
  - Fix: During this race condition, firmware checks the drive state and if the drive is failed the timer is stopped so the lockup does not occur.
  - Risk: Low
- Fixed a controller 0x13B0 lockup problem when clear controller configuration is done on encryption enabled controllers.
  - Root Cause: When deleting encryption enabled volumes during clear configuration, firmware performs encryption self tests but prior to that it suspends all internally generated I/Os. While checking if the corresponding thread is paused or not, firmware does not check the correct status and triggers a lockup thinking the thread is not suspended yet, even though it is.
  - Fix: Modify the code to check the correct status to tell if the corresponding thread is suspended or not.
  - Risk: Low
- Fixed an issue where PHY ID is always 0 in ADU report or lsscsi output when unconfigured drives are directly attached.
  - Root Cause: The correct PHY map for the controller was not being used while populating the response from firmware.
  - Fix: Populate with the correct PHY mapping for the drives connected to the controller.
  - Risk: Medium
- Fixed an issue where the filesystem or application may read incorrect data when a transformation is in progress on the logical drives.
  - Root Cause: If a Shrink Array, RAID Level Migration, or Extend Logical Drive transformation occur at the same LBAs as a pending read or write operation, in the case of a read the data may not be returned correctly or in the case of a write the subsequent read may return incorrect data.

- Fix: Prioritize the host read or write operation to complete first then allow the transformation to occur.
- Risk: Low

- Fixed a PBSI problem where incorrect details are provided to BMC from firmware.
  - Root Cause:
    1. PBSI TWI is transmitting wrong data to the master when internal Data Set Table (DST) page is updated. Firmware was checking for internal table updates on every byte put into the PBSI transmit buffer. Thus, any change to the internal table was immediately updated during the transfer causing checksum failures.
    2. If the controller does not receive entire 3 bytes Data Set Address Pointer (DSAP) write, firmware ignores entire write data.
  - Fix:
    1. Instead of checking for internal table updates on every byte transferred, only check when DSAP is written or page boundary is crossed.
    2. Accumulate the entire 3 bytes until STOP bit interrupt from TWI layer and write the new DSAP into DST.
  - Risk: Low
- Fixed an issue where rekey/encryption operation gets paused after firmware upgrade and server is warm rebooted.
  - Root Cause: If a system is warm rebooted to complete a firmware update while a rekey/encryption operation on a MaxCache-enabled volume is in progress, firmware incorrectly leaves the green backup subsystem enabled even though the relevant contents are flushed to the logical drive. The enabled green backup subsystem causes the rekeying/encryption operation to be paused.
  - Fix: Firmware will correctly disable the green backup subsystem so the rekeying/encryption operation will not be paused.
  - Risk: Low
- Fixed an issue where SMART READ DATA command via Out-of-Band (OOB) returns all ZEROs in response.
  - Root Cause: When the command returns with CHECK CONDITION status with RECOVERED ERROR sense key, OOB layer was not transferring the data back to the host, instead it returned error response.
  - Fix: Data will be returned to host when sense data is RECOVERED ERROR.
  - Risk: Low
- Fixed a problem where supercap status was reported as failed once in every three to four days through PBSI.
  - Root Cause: Firmware was interpreting a non-fatal status as failure status. Supercap state machine runs a health check every four days where an imbalance can be detected, which is not an error condition. This was reported as error status.
  - Fix: Do not consider an imbalance condition as failure status in the PBSI layer.
  - Risk:
- Fixed incorrect Maximum Connection Number (MCN) calculation for indirectly attached wide-port end-target.
  - Root Cause: In calculating the maximum number of connections for a wide-port end-target, the narrowest connection between the controller and the wide-port end-target was getting tracked. For indirectly attached devices, an incorrect device index was used to calculate the MCN value.
  - Fix: Use the correct device index to calculate the MCN value.
  - Risk: Low
- Fixed an issue to adjust Completion Timeout (CTO) range in the Dev Control 2 register and in the PCIe sub-system to handle longer CTOs.

- Root Cause: In certain servers, the CPU takes an unusually long time to send a completion to a MemRd and this period exceeds the timeout bounds set via the capabilities register space and the settings in the controller itself. This results in a CTO, which in turn triggers a fatal error.
- Fix: Modify the Dev Control 2 register's Completion Timeout value to use the 64 ms to 210 ms range. Modify the PCIe completion timeout limits from 67 ms to 1 s.
- Risk: Low

- Fixed a firmware exception during Task Management (TM) Abort handling for non-NCQ SATA command.
  - Root Cause: As designed, the check power mode (CPM) command was sent as part of TMF handling to recover the I/O. The drive completed the CPM (but higher level firmware was expecting a SATA error) and low level firmware cleared the active request for the outstanding non-NCQ command. Later, when the non-NCQ command was completed by the drive, SATA firmware had already cleared its internal references and took an exception while de-referencing the stale I/O.
  - Fix: Add checks to determine the type of command that is outstanding before processing the task management function and reset the drive for non-NCQ cases (as defined in the SATA specification). For NCQ cases, the behavior remains the same.
  - Risk: Low

- Fixed firmware timeout due to missing SMP request.
  - Root Cause: When an SMP request encounters an OPEN REJECT and the end device is an expander, then controller hardware does not allow discovery to complete resulting in a firmware timeout. Discovery can not complete because the controller hardware drain bit (DR) remains set thereby blocking all further SMP requests from the firmware.
  - Fix: Modify firmware to clear the DR bit in the hardware if the device is an expander before sending a new SMP request for discovery.
  - Risk: None

- Fixed premature Scan Completion on Re-scan.
  - Root Cause: On a topology re-scan, discovery on ports with no expander may complete before the per-port discovery status is cleared on other ports. This leads to the Scan Device being declared completed, before all devices are re-reported on the re-scan.
  - Fix: Clear the per-port Scan Device status on all ports before doing re-scan on any port.
  - Risk: Low

- Fixed I^2C read failures with SEEPROM.
  - Root Cause: According to the SEEPROM data sheet, the device requires a 5 ms wait time to process a write command and the firmware is required to allow sufficient time before issuing a subsequent write command. It was found that in a corner case scenario, multiple SEEPROM transactions coming from different threads were not adhering to this wait time causing I^2C reads to fail with NACKs. This results in the higher level APIs to skip saving data to the SEEPROM. This in turn causes the sections to be corrupted and controller reverting to factory default states on next reboot.
  - Fix: Implement ACK polling, where firmware sends a dummy write command after finishing the previous data transfer to check if the SEEPROM device is ready. Once the device sends an ACK for the dummy write, low level firmware sends a completion for the previous transaction. This ensures that the next request coming from higher level firmware will not encounter a NACK.
  - Risk: None

- Added support for better handling of reserved SMBUS addresses.
  - Root Cause: When higher level firmware calls into lower level API to initialize SMBUS channel with a reserved address, low level firmware ASSERTS instead of gracefully handling the scenario.
  - Fix: Return an error code when reserved addresses are used instead of ASSERT.
  - Risk: None

- Fixed an issue where the controller reported a lockup code 0x1F00E.
  - Root Cause: The PHYs continuously going through link up/down were causing the SAS wide port between the controller and expander to indicate there are no PHYs in the wide port. Firmware

will assert after trying to add a device while the wide port has no PHYs listed. A firmware fix is required to avoid adding a device while the port does not have any PHYs listed.
    - Fix: Add PHYs into port map under-going Loss of Sync (LOS) due to link up/down conditions to ensure not to to process a device add after LOS recovery with an empty port map.
    - Risk: Low
- Fixed an issue where encryption self test fails on a hardware block that is not used by the controller.
    - Root Cause: A hardware block that is not required for controller functionality was causing the encryption self tests to report a false failure.
    - Fix: Do not run the encryption self tests on the hardware block that isn't used for controller functionality.
    - Risk: None
- Fixed the lockup in SATA firmware when a Report Zones command gets clobbered by link error during a device reset.
    - Root Cause: When a Report Zones command gets a link error, there is a missing check in SATA firmware to terminate the command and processing continues despite the error. This causes controller SATA firmware to lose track of this I/O and hence remains active after flushing all the requests for a particular device.
    - Fix: Modify SATA firmware to consider link error along with drive errors to terminate the command and return appropriate status to the host.
    - Risk: None
- Fixed an issue where I/O timeouts due to Open frames that results in Open Reject Bad Destination (ORBD).
    - Root Cause: An invalid OPEN frame caused an Open Reject Bad Destination (ORBD) response and a host I/O timeout resulting in a server node being evicted from a cluster.
    - Fix: To avoid the server node eviction, added a firmware recovery mechanism when ORBD response message is detected. Firmware will now issue an internal reset that will cause the outstanding I/Os to be flushed from the drive and those I/Os to be retried internally.
    - Risk: Low/Medium
- Fixed an issue where an attached JBOD with a redundant fan module removed results in overall status of the JBOD reported as failed.
    - Root Cause: The logic parsing the SES status from the JBOD was incorrectly promoting this state into failed instead of degraded.
    - Fix: Modified the logic to appropriately identify degraded and failed states so that the proper status is reported up to other software layers.
    - Risk: Low
- Fixed an issue where a drive reporting hardware failure causes the controller to appear as hung for 15 minutes.
    - Root Cause: A particular drive behavior was observed in which media access requests to a failing drive always result in non-response while management requests for device status immediately return with fatal status indicating device failure. Because the I/O path error recovery immediately attempts several retries, the controller error recovery would become stalled in a loop repeating retry, timeout, and I/O recovery for each request until it had exhausted the retry counts and worked it's way through the entire backlog of I/O for that device. For a sufficiently busy system, this was observed to take as long as 15 minutes. It was also found that several internal firmware processes were not evoking appropriate error handling for this status even for non I/O requests.
    - Fix: The I/O error recovery path was modified to send a SCSI TestUnitReady request prior to attempting the first retry of an IO to the device. If the device responds with status indicating it has a hardware fault, then no further recovery is attempted, and the drive is immediately marked failed. This same error handling was also added to several non I/O contexts so that the device can be taken offline promptly at the first sighting of this status.
    - Risk: Low

- Fixed an issue where a MaxCache write bypass due to the line already being loaded results in that line being stuck on a pending queue.
  - Root Cause: A MaxCache write bypass operation may result in a cache line being stuck on a pending queue if the write to the primary logical drive gets retried. The cache line remained on the pending queue because an I/O counter was incorrectly incremented twice when the retry was executed.
  - Fix: Corrected the firmware so the I/O counter will not be incremented twice.
  - Risk: Low
- Fixed an issue resulting in the controller becoming non-responsive, if repeatedly forced between SIS (Inbox) and PQI I/O submission modes where an I/O timeout is also occurring.
  - Root Cause: A hardware queue is used for request submission in all I/O modes, but the hardware queue "not-empty" interrupt was not always being armed on transition from PQI to SIS mode of operation.
  - Fix: Explicitly enable the non-empty interrupt when starting each host transport so that it's always armed prior to the driver submitting the first request in the new mode of operation.
  - Risk: Low
- Fixed an issue in the write coalescing logic where a partially coalesced opportunity would be abandoned under very high queue depth workloads.
  - Root Cause: The write coalescing logic was abandoning the current coalescing opportunity, if all of the logical request resources had been consumed.
  - Fix: Adjusted the request submission path to force I/O's into a wait queue when there are no resources available. Also if there are no further host I/O's to attempt to coalesce, wait for a tick in hopes that recent or imminent I/O completions will result in additional I/O submissions that could be coalesced.
  - Risk: Medium
- Fixed a race condition in which the lockup code reported by the OS driver does not match the actual code given by firmware.
  - Root Cause: The interrupt to the driver was being set prior to the lockup code being populated in PCIe config space. This created a race condition where the driver could react to the lockup and read out the code prior to the controller populating that field.
  - Fix: Don't notify the driver of the lockup until after the lockup code field has been populated.
  - Risk: Medium
- Fixed an issue where SATA drives are unnecessarily reset during a power-on self-test.
  - Root Cause: Under balanced power mode, various self-tests are run to determine what SAS PHY rates should be used in the current or in a future boot. These tests were not accounting for restricting SATA drives to 6 Gbps which is their max rate, which would result in a drive reset each time the PHY was reconfigured to attempt faster rates.
  - Fix: Don't attempt to reconfigure PHY rates when a SATA device is direct-attached.
  - Risk: Low
- Fixed an issue where a non-hotplug attached drive's bay number is reported as 255/0xFF after the device inventory re-query API is invoked.
  - Root Cause: The requery API was invoking a discovery routine that first invalidated the drive box/bay information but then did not handle the direct-attach non-hotplug case when restoring this inventory information.
  - Fix: Modified the re-query API to handle this configuration appropriately.
  - Risk: Low
- Fixed a controller hang after attempting 'Clear Configuration' on a degraded volume undergoing Rapid Parity Initialization (RPI).
  - Root Cause: A field in use by the RPI process's drive state information was not being initialized appropriately at boot resulting in the RPI process being unable to be halted when the user directed the controller to clear all RAID configuration data.

- Fix: Properly initialize the RPI state information such that incorrect state information is not applied to failed or missing drives.
- Risk: Low

- Fixed an issue where I/O coalescing is not enabled when DDR cache is not present or enabled.
  - Root Cause: The logic used to determine if coalescing should be enabled for a volume was incorrectly looking at information regarding DDR cache configuration. For controllers that do not support DDR caching, this was caused by the associated fields not being initialized appropriately.
  - Fix: When creating a new volume, explicitly disable the volume cache properties even when DDR cache is completely disabled. Also, attempting to update the coalescing state information will also trigger an update of the cache status first so that the information used is up to date.
  - Risk: Low
- Fixed a condition in which a drive is incorrectly marked failed (reason = 0x3) if spun down and then immediately used to create a new volume with rapid parity initialization.
  - Root Cause: When the drive is spun down, attempts to update the RAID metadata in the RPI process result in check conditions and failure that are not handled appropriately.
  - Fix: Added error handling logic to issue a START_UNIT request to the drive prior to retry if it returns a NOT_READY check condition.
  - Risk: Low
- Fixed a race condition in which MaxCache metadata is lost after creating a new volume while running I/O traffic.
  - Root Cause: The sequencing of events during volume configuration results in DDR cache being flushed prior to the new volume being created. After the DDR cache has flushed, the backup power subsystem is momentarily disabled but a logic error results in the backup subsystem remaining disabled even though the configuration procedure continues and invalidates the metadata snapshot and restores the metadata to cache. If power is subsequently lost, the current metadata can be lost.
  - Fix: Do not disable the backup subsystem during a configuration event since the resultant flush and restore of the metadata snapshot or via backup power loss process would be the same.
  - Risk: Low
- Fixed an issue where 'abnormal volume state' is observed with multiple MaxCache volumes configured using 512e drives.
  - Root Cause: Logic restoring the SSD cache metadata does so based on fixed offset logic which assumed volume alignment based on the sizes of the volumes. However, volumes configured on 512e drives are aligned to their native block size, for example 4 KiB.
  - Fix: Properly account for the alignment requirements when iterating through cache luns at different offsets.
  - Risk: Low
- Fixed a potential controller hang during volume transformation if the backup power source suddenly goes offline, is spuriously hot-removed, or indicates a cabling error.
  - Root Cause: The transformation state machine is halted on backup power loss. If a host I/O happens to span the boundary between the non-transformed region of the volume and the newly transformed region, that I/O would cause a deadlock until the backup power source became healthy.
  - Fix: Since the backup power source may never become healthy in this condition, logic was added to split the host I/O into two requests: one each for the non-transformed and transformed regions of the volume.
  - Risk: Low
- Fixed a LUN reset issue during heavy write I/O workload on cache enabled volumes.
  - Root Cause: On cache enabled volumes under heavy I/O workload with controller cache full condition, on rare occasions host writes could wait longer than expected causing a LUN reset condition.

- Fix: Frequency of Host I/Os waiting for cache writes check is increased to complete the host writes without having to wait longer and causing a LUN reset.
- Risk: Medium

- Fixed an incorrect spare SSD physical drive state during rebuild.
  - Root Cause: During rebuild on spare SSD drives, physical drive state can be out of sync with logical drive state as some of the physical drive spare rebuild fields are not cleared correctly.
  - Fix: Spare SSD physical drive rebuild fields are now handled correctly and is in sync with logical drive state.
  - Risk: Low
- Fixed a I/O latency issue during periodic disk write metadata operations.
  - Root Cause: Periodic disk metadata write operations occurs from a CPU core which also processes I/O, resulting in small latency increase on host I/Os.
  - Fix: Periodic disk metadata write operations are moved to a different CPU core, leaving I/O processing cores to process I/Os without periodic latency issues.
  - Risk: Medium
- Fixed an issue where drive failure reason code can occasionally get updated with generic drive failure reason code.
  - Root Cause: During media error handling cases a drive can get marked offline if the data is not recoverable and in some conditions the failure reason code can get updated with generic drive failure reason code.
  - Fix: Checks are added to not update the failure reason code for a failed drive if it already has a specific drive failure reason code.
  - Risk: Low
- Fixed an issue where migrating an expanding MaxCache/volume pair after a dirty shutdown to another controller fails to get configured.
  - Root Cause: During migration of expanding MaxCache/volume pair after a dirty shutdown to another controller, MaxCache pair incorrectly gets configured to default cache ratio, causing MaxCache pair to not get added to the configuration.
  - Fix: MaxCache configuration logic now sets the correct cache ratio settings to get configured into migrating controller.
  - Risk: Low
- Fixed a controller lock up issue when an expanding volume is migrated from a controller that supports DDR cache to a controller that does not support DDR cache.
  - Root Cause: Expansion uses controller memory (or) disk storage for its operation based on board type and when its migrated to a controller that doesn't support DDR cache, it resumes expansion expecting the same memory and leads to a controller lock up.
  - Fix: Expansion memory information is now stored in disk metadata, so when migrated to a different board type, expansion can switch to memory based or disk based operation and continue without a lockup.
  - Risk: Low
- Fixed an issue where cache ratio is reset to default values after backup cable error condition on split mirror configuration.
  - Root Cause: In split mirror configurations on backup cable error condition, cache reconfigures to read only cache. But during further host tools management command processing, cache ratio gets reinitialized to default values.
  - Fix: Host tools management command path now checks for cable error conditions to retain read only cache settings than reinitializing to default cache ratio setting values.
  - Risk: Low
- Fixed an issue where incorrect I^2C address for PBSI configuration gets displayed in host tools after I^2C address is modified.

- Root Cause: When I^2C address for PBSI configuration gets modified, the values are stored in NVRAM to get activated after controller reboot. However before reboot when current I^2C address is queried it returns the yet to be activated I^2C address incorrectly rather than returning the current I^2C address.
- Fix: Current PBSI I^2C address is now exported from active I^2C configuration data.
- Risk: Low

- Fixed an controller hang issue after a cold boot during volume rekeying.
  - Root Cause: After initiating a volume rekey operation followed by cold boot, controller gets into a tight loop where it expects outstanding flushes to complete. However, there are situations where this condition can be met early even before the flush thread starts, causing a controller hang.
  - Fix: Added a condition to check if the flush thread has started during volume rekey cold boot case to avoid the tight while loop and controller boots after flushing the required data.
  - Risk: Low
- Fixed an controller lock up issue on HBA SAS drive dual path configuration.
  - Root Cause: In HBA SAS drive dual path configurations, during LUN reset timeout handling, controller handles the error always in active path. However, if LUN reset timeout occurred on an inactive path, error handling logic incorrectly handles it in active path leading up to controller lockup.
  - Fix: LUN reset timeout error handling now can handle the appropriate path.
  - Risk: Low
- Fixed a rare host utility triggered device reset command time out issue in minimum power mode.
  - Root Cause: Host requests submitted to HBA path and RAID path are stored in queues and sequencing to clear the queues cannot be maintained in minimum power mode condition. Requests on the queue with less traffic will get processed quickly as it is prioritized. This causes the device reset request to timeout as the HBA path queue is not cleared.
  - Fix: Added changes to balance the queues more efficiently in minimum power mode so that HBA queues can be cleared and device reset command gets processed on time.
  - Risk: Low
- Fixed an issue where turning ON/OFF the identify all LED command frequently may occasionally not turn on/off all the drive LEDs.
  - Root Cause: When Identify all LED command is toggled frequently, a race condition between ON/OFF host management commands gets created resulting in some LEDs not turning on/off at that moment.
  - Fix: Changes are done to sequence the host management commands to avoid the race condition, so all LEDs can be turned ON/OFF when toggled frequently.
  - Risk: Low
- Fixed an issue where Green Backup subsystem reports backup/restore error persistently to the user during controller migration.
  - Root Cause: During controller migration, if the board had a previous backup disable code stored in NVRAM, it gets propagated to disk metadata and reports backup error.
  - Fix: When disk metadata configuration change is detected, clear the stale NVRAM previous backup error code and do not propagate the backup error to disk metadata.
  - Risk: Low
- Fixed an issue where PBSI shows a failed drive member as part of volume when spare drive is active.
  - Root Cause: Fixed an issue where PBSI shows a failed drive member as part of volume when spare drive is active.
  - Fix: If member drive of a volume is offline, a spare drive check is added in PBSI module to get the volume data.
  - Risk: Low

- Fixed an rare issue where internal Test Unit Ready (TUR) command during drive hot removal/addition causes controller hang.
  - Root Cause: During drive hot addition, internal TUR commands are sent back to back. During this sequence of back to back TUR commands, if a command fails, then a controller hang may occur.
  - Fix: Internal back to back TUR commands are now sent with added delay between them.
  - Risk: Low
- Fixed an issue where marking an Unrecoverable Read Error (URE) on a drive can return success though the drive did not plant the URE properly.
  - Root Cause: After marking a URE on a drive using write commands and verifying the return status of the command, controller issues reads to verify if the drive is holding the planted URE. On occasions, if the drive is not holding the URE, subsequent reads can succeed, whereas a check condition is expected. This status is not infused into controllers mark URE logic.
  - Fix: Mark URE logic now includes the read command status check and if read succeeds on a planted URE location, it fails the drive as its not expected.
  - Risk: Low
- Fixed an occasional controller lock up issue when resuming rebuild operation after a reboot.
  - Root Cause: During rebuild when drive errors are encountered, sector level rebuild occurs for that stripe and this rebuild size information gets stored in disk metadata during reboot. On subsequent boot, due to calculation errors the rebuild size ends up as zero and leads to a controller lock up.
  - Fix: Rebuild size calculation logic post reboot is modified to arrive at the same value prior to reboot so rebuild operation can resume post reboot.
  - Risk: Low
- Fixed a surface scan inconsistency check event notification failure in logs on RAID 1 volumes.
  - Root Cause: Surface scan inconsistency failure notification events counter for RAID 1 volumes gets incremented even before the notification feature is enabled, resulting in events to not get logged.
  - Fix: Surface scan inconsistency failure notification event counter for RAID 1 volume gets incremented only after the notification feature is enabled.
  - Risk: Low
- Fixed a rare controller lock up issue when activating spare drive for an encrypted volume during I/Os.
  - Root Cause: Spare drive activation for an encrypted volume is done in a critical section and occasionally internal shared resource used for the operation gets freed up before critical section can terminate, leading to a controller lock up.
  - Fix: Existing dedicated crypto resource is used for encrypted volume spare activation and dedicated resource doesn't require allocating/freeing it up in critical section.
  - Risk: Low
- Fixed an issue where a drive failed due to Read Capacity failure is presented to the host.
  - Root Cause: When read capacity command is executed, internal drive present bit was incorrectly set, causing the drive to be presented to host.
  - Fix: After read capacity command is executed, if return status indicates a failure, do not set drive present bit.
  - Risk: Low
- Fixed an issue where periodic background drive temperature check may occasionally skip every other minute.
  - Root Cause: Periodic drive temperature check occurs every minute and on occasions when time difference calculation error occurs, some drives could skip the periodic check every other minute.
  - Fix: Periodic drive temperature time difference check calculations are modified to account for skews and now checks every minute as expected.
  - Risk: Low

- Fixed a controller lockup issue during expansion after multiple subsequent attempts to abort the expansion earlier.
  - Root Cause: When expansion is aborted disk metadata holds the previous configuration data to inform a failed expansion to host tools. However during multiple expansion aborts, disk metadata to store aborted expansion volumes runs out of allocated space and controller locks up.
  - Fix: Free up the previous stale expansion aborted volume configuration data from disk metadata when a subsequent expansion operation is attempted for the volume.
  - Risk: Low
- Fixed an issue where encryption Host management commands are processed for MaxCache volume.
  - Root Cause: Encryption rekey host management command is not applicable for MaxCache volumes and the command path fails to check the volume type before processing the command.
  - Fix: Encryption host management commands now checks for volume type and allows only for primary logical volumes.
  - Risk: Low
- Fixed an issue where the host may read incorrect data after an array expansion operation.
  - Root Cause: During expansion if an Unrecoverable Read Error (URE) is encountered on a LBA, expansion logic propagates the URE to the new volume. In rare cases, if marking the URE is failed by the drive, the expansion logic did not detect the error condition and completes the expansion. Subsequent reads to the LBA with the failed URE will lead to incorrect data being returned.
  - Fix: Expansion logic detects the failed URE propagation and fails the physical drive.
  - Risk: Low
- Fixed an issue where high severity events are not given priority for event logged when controller is in critical event buffer over flow condition.
  - Root Cause: During event buffer overflow conditions, high severity events are not given increased priority over lower severity events as the logic to prioritize them had incorrect conditional checks.
  - Fix: Conditional checks for event prioritization is modified to make sure high severity events are prioritized in event buffer over flow conditions.
  - Risk: Low
- Fixed an issue where redundant surface scan pass complete events can get posted for the same surface scan iteration across controller reboot.
  - Root Cause: Surface scan completion event gets posted when it completes an iteration and during reboot the logic reinitializes to a previous values leading to redundant surface scan completion event for the previous iteration.
  - Fix: Surface scan event logging reinitialization logic on reboot is modified to log event once per surface scan iteration completion.
  - Risk: Low
- Fixed an issue where during expander firmware upgrade (reduced functionality mode), controllers last management device index is kept open for processing I/Os that can lead to I/O timeout scenarios.
  - Root Cause: During expander firmware upgrade, controller firmware pauses/resumes I/Os for all management devices (Expanders/SEPs), except for the last device index, which can still receive I/Os from controller during upgrade time and can lead to I/O time out scenarios.
  - Fix: Extended the pause/resume I/Os boundary checks during reduced functionality mode expander firmware to include all management devices.
  - Risk: Low

2.2.2 UEFI Fixes

Note: Microsoft signed and secure boot is supported**.**

2.2.2.1 Fixes and Enhancements for UEFI Driver 1.3.11.1/Legacy BIOS v1.3.11.3

This release includes the following UEFI fixes and enhancements:

- Added a new menu to configure Port Discovery Protocol. The sub menu **Configure Port Discovery Protocol** is added to HII under **Controller Configuration** menu. The options added are to view current/pending protocol settings, Set Port Discovery Protocol(Auto/UBM/SGPIO), and to Reset to Default settings.
- Added a new menu to configure Port Expander Minimum Scan Duration Implementation. The **Modify Expander Minimum Scan Duration** option is added to HII form under the **Controller Configuration** menu. The option will modify the **Expander Minimum Scan Duration** in seconds.
- Fixed an issue where the HII help text does not match the options provided for physical drive write cache.
  - Root Cause: Incorrect HII help text for physical drive write cache options.
  - Fix: HII help text corrected for physical drive write cache options.
  - Risk: Low
- Fixed an issue where the drive rebuild progress percentage is incorrect.
  - Root Cause: The calculation for recovery progress percentage was using the size in blocks of the entire logical drive as a number for how many blocks needed to be handled. This value was being used in determining how many blocks of data still needed to be recovered in the rebuild operation. For other volume operations like expansion and rekeying this would be valid because all blocks will need to be handled. In the case of volume recovery after a failed physical drive event, the number of blocks being handled is equal to only the number of blocks on one physical drive so the progress percentage math became incorrect. The progress percentage for all of these operations is currently calculated using the same internal function that automatically used the logical drive size in blocks for determining progress.
  - Fix: The internal function that calculates volume operation progress percentage now dynamically determines how many total blocks need to be handled depending on the operation taking place. If the volume is rebuilding then the number to be used for total blocks to be handled is equal to the number of blocks on one physical drive. For all other operations the number of total blocks in the logical drive will be used.
  - Risk: Low
- Fixed an issue where the deferred drive flashing fails for drives with longer drive revision.
  - Root Cause: Deferred Drive flashing does not support drives which has 8-byte drive revision.
  - Fix: Allow SCSI direct flash for all drives.
  - Risk: Low
- Fixed an issue where software failed to create split mirror from Raid 1 Array having the spare drives.
  - Root Cause: Software includes the number of spare drives in the array while sending the command to the controller firmware to create the split mirror that results in an error because only data drives in the array should be included.
  - Fix: Fixed the logic where number of spare drives was getting included to only include the correct number of data drives.
  - Risk: Low
- Fixed an issue where the erase drive option is not available under Disk Utilities in HII when a controller is in HBA mode.
  - Root Cause: Legacy erase support did not include erase on HBA drives, and when the firmware added support for HBA erase, software was not updated to match.
  - Fix: Instead of blocking erase on all drives that are HBA mode drives, use the Sense Feature page (if available) to determine erase support.
  - Risk: Low
- Fixed an issue where the max parallel surface scan count returns invalid value 0.

- Root Cause: The max parallel surface scan count is read from a field that it does not initialize. If this field was previously initialized by another API call, this function will return a valid value. If it was not initialized this function will erroneously return 0.
- Fix: Ensure that the max parallel surface scan count field is initialized before attempting to read and return the information.
- Risk: Low

- Fixed an issue where the driver allows rekey operation when backup power source status is not OK.
  - Root Cause: Checks were not being made before rekey to determine if the backup power source is present and charged.
  - Fix: Add checks for backup power source and powerless transformations where there is no expansion memory.
  - Risk: Low
- Fixed to modify No Battery Write Cache when IOBypass is enabled for the volume
  - Root Cause: The driver did not check if all logical drives have accelerator set to IOBypass.
  - Fix: The driver will now return a new error code when all logical drives have accelerator set to IOBypass.
  - Risk: Low
- Fixed an issue where the bootable OS volumes still exist and OS is booting even after clear configuration from UEFI HII when OS installed on Logical drive RAID 1.
  - Root Cause: The driver was not saving the command status from the helper function which actually executes the I/O request.
  - Fix: The driver will now save the command status from the helper function that performs the I/O request.
  - Risk: Low
- Fixed an issue where the Disk Utilities menu does not report the reason if the drive configuration is not supported.
  - Root Cause: The unsupported drive status and reason decode is not considered for drive information under Disk Utilities menu.
  - Fix: A field is added to report unsupported reason if the drive configuration is unsupported.
  - Risk: Low
- Fixed an issue where the stripe size option is not enabled when creating RAID5 array with more than six drives.
  - Root Cause: Out of range value is used to check the eligibility for stripe size configuration.
  - Fix: The stripe size option is enabled based on appropriate range check.
  - Risk: Low
- Fixed an issue where an error code in driver health message is not displayed when an unsupported device is connected.
  - Root Cause: The unsupported drive status flags are not considered to set error status using driver health message.
  - Fix: Consider the unsupported device status flags to enable error status in driver health message.
  - Risk: Low
- Fixed an issue where no warning message is displayed when Self Encrypting drives are used for configuration.
  - Root Cause: HII forms does not provide any warning to the user when using Self Encrypting drives for logical drive configuration.
  - Fix: Warning message is added to the HII forms when Self Encrypting drives are used for configuration.
  - Risk: Low
- Fixed where the I^2C slave address can be configured to reserved values in Out of Band Management menu.

- Root Cause: The software was limiting the slave address between 0xD0-0xFF.
- Fix: Remove the limitation of the I^2C slave address and allow the firmware to validate the I^2C address.
- Risk: Low

- Fixed an issue where enabling the SMBus physical channel under Out-of-Band (OOB) message settings in HII menu performs disable operation.
  - Root Cause: Incorrect values used for enabling and disabling the SMBus physical channel for OOB messaging configuration.
  - Fix: Corrected values used for enabling and disabling the SMBus physical channel for OOB messaging configuration.
  - Risk: Low

This release includes the following Legacy BIOS fixes and enhancements:

- Fixed an issue where the system cannot boot when hard drive is first on IPL in legacy mode.
  - Root Cause: If there is a Unit Attention (UA) pending on a drive and an I/O is sent to that drive, the command returns success which is replayed to the System BIOS. But when the System BIOS tried to read the buffer, there will be no valid content to process. So it skips the current target and proceeds to the next boot target in the IPL.
  - Fix: Check the SCSI status in addition to the overall status of the command. If the SCSI status is returned as CHECK CONDITION, return error for the command so that the command can be retried.
  - Risk: Medium
- Fixed an issue where the mirror group information of the drives were displayed incorrectly.
  - Root Cause: The mirror drive pairs were incorrectly grouped.
  - Fix: Corrected the mirror drive pairs grouping.
  - Risk: Low
- Fixed an issue where the logical drives were created with lesser number of blocks compared to the other utilities.
  - Root Cause: The incorrect number of blocks were used for the size calculation.
  - Fix: Corrected the number of blocks used for the size calculation.
  - Risk: Low
- Fixed an issue where memory corruption was observed during POST with larger drive configuration.
  - Root Cause: A conditional check to see if the number of drives exceeded the maximum supported drives was missing which led to unauthorized memory access.
  - Fix: Added the missing conditional check to prevent unauthorized memory access.
  - Risk: Low
- Fixed an issue where the logical drive was created with maximum capacity even if the specified size is 16 MiB on a 4Kn drive.
  - Root Cause: Incorrect blocksize was used for the capacity calculation.
  - Fix: Used the correct blocksize for the capacity calculation.
  - Risk: Low

2.2.3 Driver Fixes

2.2.3.1 Fixes and Enhancements for Linux Driver Build 1.2.16-040

This release includes the following enhancements:

- Fixed an issue where the system hangs when resuming from hibernation.
  - Root Cause: A recent driver change to the system state transitions (suspend/hibernate/shutdown) introduced an issue where the system would successfully suspend/hibernate the first time but subsequent attempts to suspend/hibernate would hang the system.

- Fix: The driver now correctly restores all driver state after a system resume.
- Risk: Low

- Fixed an issue where the enclosure identifier field corresponding to the physical devices became empty in lsscsi/sysfs during device rescan.
  - Root Cause: During initial device enumeration, the devtype attribute of the device (in current case enclosure device) is filled during `slave_configure`. But whenever a rescan occurs, the firmware would return zero for this field, and the valid devtype is overwritten by zero. The devtype field should not be updated using the value returned by the controller firmware.
  - Fix: Update this device attribute only during slave_configure.
  - Risk:
- Fixed an issue to display topology using PHY numbers.
  - Root Cause: New feature to include PHY numbers in physical device attributes on multipath configuration.
  - Fix: As part of the device discovery, the active path index of the target would be changed and accordingly PHY number is updated to SAS transport layer.
- Fixed an issue where the driver did not export valid SAS `initiator_port_protocols` and `target_port_protocols` to sysfs.
  - Root Cause: The `port_protocols` values are hard coded and hence lsscsi was not showing correct values.
  - Fix: Export `port_protocols` values depending on the type of target device from controller to end device.
  - Risk: Low
- Fixed an issue where the driver would cause a kernel panic if it got an invalid response from the controller.
  - Root Cause: Code left over from prior development.
  - Fix: Driver now takes the controller offline instead of causing kernel panics.
  - Risk: Low
- Fixed an issue where the LUN reset, system shutdown, system suspend, system hibernate, controller offline, and I/O requests could reach the controller when they are not required.
  - Root Cause: Inadequate synchronization in the driver.
  - Fix: Optimized the driver synchronization code.
  - Risk: Low
- Fixed an issue where after modifying the logical volume size, lsblk command still shows up previous size of logical volume.
  - Root Cause: The size expansion of the logical volume was not detected by driver. After logical volume size expansion, the driver is not notifying the OS scsi-mid-layer to rescan the device.
  - Fix: the driver identifies the volume size expansion and notifies the OS to rescan logical volume. OS will issue SCSI command SERVICE_ACTION_IN_16 (or READ_CAPACITY) to update the size capacity.
  - Risk: Low
- Fixed an issue where during the device discovery, the driver issued an INQUIRY to physical devices, and these requests could get lost due to broken devices, loss of signal issues, etc. This would cause device discovery to get hung.
  - Root Cause: Sending INQUIRY directly to physical devices is not 100% reliable during device discovery.
  - Fix: The driver now uses the information returned by the controller firmware in place of the information formerly returned by the INQUIRY command.
  - Risk: Low

2.2.3.2 Fixes and Enhancements for FreeBSD Driver Build 4030.0.101

This release provides the following enhancements and bug fixes:

- Fixed an issue where the volume size is not reflecting to OS after expansion.
  - Root Cause: The driver is not detecting the expansion status and not doing disk rescan to update the reflected volume size to OS.
  - Fix: Driver detects the volume expansion status and it sets the rescan flag, later it would do re-probe the scsi disk which includes sending SCSI READ CAPACITY command and updating the size and it is visible to operating system tools.
  - Risk: Low

2.2.3.3 Fixes and Enhancements for Solaris Driver Build 4030.0.101

This release provides the following enhancements and bug fixes:

- Observed hang on executing encryption enable/disable and rekey operation.
  - Root Cause: The IOBypass disabled response was received for I/Os belonging to OS. On changing the command status as DEV_GONE for IOBypass disabled created an unexpected behavior for OS.
  - Fix: Retry the I/O packets through RAID path for IOBypass disabled response cases.
  - Risk: Low
- Fixed an issue where the device removal may fail at certain occasions even though the device might be free.
  - Root Cause: Some device handles are not removed in the kernel.
  - Fix: Added retries to device removal routine if first attempt to remove the device from the OS fails.
  - Risk: Low
- Fixed an issue where references to newly discovered device at the same target/lun are getting lost.
  - Root Cause: The target/lun location was set as null in two places:
    - Before creating remove list in device discovery routine.
    - Device memory is released when OS calls target free.
  - Fix: Changed the logic to add the device with the same target/lun only when the previous device with same target/lun is removed when OS calls driver target free.
  - Risk: Low

2.2.3.4 Fixes and Enhancements for Windows Build 106.190.4.1062

This release provides the following enhancements and bug fixes:

- Fixed an issue where the early completion of the SRB_FUNCTION_SHUTDOWN SRB is in dump mode.
  - Root Cause: During a blue screen crash dump or hibernation operation, the driver receives the SRB_FUNCTION_SHUTDOWN SRB in both `buildio` and `startio` routines, even through the driver is processing the request from the `buildio` routine. This is leading to the early completion of the SRB and subsequent removal of power from the adapter while processing a controller flush cache command.
  - Fix: SmartPQI driver should not complete the shutdown SRB from `startio`, if the blue screen crash dump or hibernation operation is occurring. The Shutdown SRB will be completed as part of normal cleanup path.
  - Risk: Low
- Fixed host tracking of SATA hot plug/hot add drive. The driver was not setting the feature bit to inform the controller to return a unique WWN ID per SATA drive via Inquiry VPD 0x83 rather then returning WWN ID per SAT drive port.

- Root Cause: Driver was not setting SATA WWN ID feature bit in configuration table so that Inquiry VPD 0x83 return unique WWID per drive rather than per port.
- Fix: Driver now sets configuration table feature bit 12. The controller will return a unique WWN ID per SATA drive via Inquiry VPD 0x83. This allows the host to uniquely identify a SATA drive for tracking location.
- Risk: Low

- Fixed BSOD on Windows 8 HCK CHAOS test. While executing HCK CHAOS test, SmartPQI driver crashes with bugcheck 0x7A (KERNEL_DATA_INPAGE_ERROR).
  - Root Cause: While returning from hibernation, the storport workitem is not able to queue the worker thread. This causes the initialization of the SmartPQI controller to fail. Resuming from hibernate will cause the kernel to crash with bugcheck 0x7A.
  - Fix: Replaced storport workitem with a direct function call for initializing and configuring the controller after returning from Hibernate/Sleep.
  - Risk: Low
- Fixed BSOD on reboot path. While rebooting, the SmartPQI driver crashes with bugcheck 0xD1.
  - Root Cause: If any of the OutboundQueue DPC objects takes more time to process, the DPC objects will be saved for debug purpose. As part of reboot operation, SmartPQI driver free all the operational outbound queues, this involves freeing DPC objects that were part of OutboundQueues. If the flush cache completion routine accesses invalid DPC objects, system will crash with bugcheck code 0xD1.
  - Fix: During cleanup path, if the longest DPC object points to OutboundQueue DPC objects, SmartPQI driver reset the longest DPC object to NULL.
  - Risk: Low
- Fixed an issue where Sleep test causes the SmartPQI driver to trigger a bugcheck 0xD1.
  - Root Cause: During a Sleep operation the DPC object for the outbound queue was freed so that when the OS came out of the Sleep operation the SmartPQI driver accessed an invalid pointer causing the bugcheck 0xD1 crash.
  - Fix: When coming back from a Sleep operation the SmartPQI driver will reinitialize the pointer to the DPC object.
  - Risk: Low
- Added an enhancement to fix the stale Drive firmware version returned by a Power Shell command.
  - Root Cause: An IOCTL to get the drive firmware version may complete before a rescan occurs. The driver now bypasses rescan and requests the information from the controller firmware to respond to the IOCTL.
  - Fix: Request drive firmware information from the controller.
  - Risk: High
- Fixed an issue while executing driver disable/reinstall, the SmartPQI driver crashes with bugcheck 0xD1.
  - Root Cause: Recently driver code path changed the handling of command completion from DPC objects to direct function call. As a result, Flush Cache completion is called as a direct function call and it makes other operational queue DPC objects to delay its processing. Flush cache completion will free all the resources and DPC scheduled for different operational outbound queue will access NULL pointer and subsequently the system will go for bugcheck 0xD1.
  - Fix: Changed the processing of Flush cache completion from direct function call to DPC queuing.
  - Risk: Low
- Fixed an issue where the application hangs due to lost command. When issuing "Get-Disk" from Powershell, the command would eventually hang indefinitely.
  - Root Cause: A DPC was being invoked with command context but if the DPC was issued while the DPC was already queued for another command, it could clobber the command info context. The result is only one command would end up completing even though multiple commands were queued for completion.
  - Fix: Remove DPC from the completion logic and invoke completion handler directly. It is not needed in order to handle this completion because it doesn't require chaining.

- Risk: Low

- Fixed the registry lookup for determining where I/O is completed was using the string "`SubmitViaStartIo`" instead of "`CompleteIoInDpc`".
  - Root Cause: Copy and paste error caused the lookup to use the wrong string.
  - Fix: Replace string for complete I/O location with "`CompleteIoInDpc`".
  - Risk: Low
- Fixed an issue where while executing sleep, the windows SmartPQI driver crashes with bugcheck 0x7E.
  - Root Cause: Driver is not checking `pConfigInfo` is a NULL pointer or not when returning from sleep.
  - Fix: Do a NULL check before accessing `pConfigInfo`.
  - Risk: Low
- Fixed an issue where the system would BSOD when doing unnecessary initialization of the multi-tag table after declaring controller lockup.
  - Root Cause: Unnecessary initialization of the multi-tag table.
  - Fix: Remove the unnecessary initialization of the multi-tag table.
  - Risk: Low
- Fixed an issue that causes a DRIVER_POWER_STATE_FAILURE BSOD.
  - Root Cause: Blocking commands in the miniport layer with an SRB_STATUS_BUSY causes a DRIVER_POWER_STATE_FAILURE BSOD.
  - Fix: The driver now handles I/O correctly to prevent the BSOD.
  - Risk: Low
- Fixed an issue where assert did not take into account hibernation/sleep mode where the driver limits the max I/O setting.
  - Root Cause: Driver limits the maximum I/O allowed to 32 commands during hibernation/sleep mode.
  - Fix: Chang Assert to handle the maximum I/O allowed correctly.
  - Risk: Low

2.2.3.5 Fixes and Enhancements for VMware Driver Build 4030.0.101

This release provides the following enhancements and bug fixes:

- Fixed an issue when the capacity of a logical volume is increased using management application, the updated size was not getting reflected in ESXi.
  - Root cause: The driver was not taking care of the expansion status and not notifying the OS to update the size.
  - Fix: The driver detects the volume expansion status and it sets the flag. After that, first command to that volume will be completed with sense data 06:2A:09 Unit Attention: Capacity Data has Changed. With that OS will trigger READ_CAPACITY and update the volume size
  - Risk: Low
- Added module parameters to enable the feature SATA_WWN_FOR_DEV_UNIQUE_ID.
  - Details: In the current driver, WWN ID of the attached SATA drive is PHY/slot dependent. This makes SATA drive reinserted in another slot to discover as new device and resulting the existing datastore to become unavailable.
  - Fix: Enabling the feature SATA_WWN_FOR_DEV_UNIQUE_ID using module parameters, which will assign WWN ID reported by the SATA drive. By default, this feature is disabled. If this module parameter is enabled, the current datastores might become unavailable. So, this setting is recommend for fresh server installations.
- Fixed an issue while testing for device removal failures, it is observed that the references to newly discovered device at the same target/lun getting lost.
  - Root Cause: The target/lun location became NULL in two places.

1. Before creating remove list in device discovery routine.
2. Device memory is released when OS calls target free.

- Fix: Changed the logic to add the device with the same target/lun only when the previous device with same target/lun is removed once OS calls driver target free.
- Risk: Medium

2.2.4 Management Software Fixes

2.2.4.1 Fixes and Enhancements for Arcconf/maxView Build B23821

This release includes the following fixes and enhancements for arcconf/maxView:

- Add the support to configure Backplane Discovery Protocol setting.
- Add the support to configure Expander Scan Duration.
- Fixed an issue where maxView cannot delete logical device with no valid partition on it.
  - Root Cause: maxView disables deletion of logical device option when a reserved partition is present on the logical device in windows.
  - Fix: Added changes to ignore the reserved partition on the logical device in windows while deleting.
  - Risk: Low
- Fixed an issue where Arcconf displays Spare Activation mode values incorrectly.
  - Root Cause: Arcconf displays the Spare Activation mode setting as Enable/Disable incorrectly.
  - Fix: Added changes for Arcconf displays the Spare Activation mode setting as Failure/Predictive.
  - Risk: Low
- Fixed an issue where Arcconf doesn't support valid stripe sizes for logical device creation.
  - Root Cause: Arcconf disabled certain supported stripe sizes for different RAID levels and sub arrays.
  - Fix: Added changes for Arcconf to calculate and allow the logical device creation for valid stripe sizes.
  - Risk: Low
- Fixed an issue where Arcconf collects limited UART log as part of support archive.
  - Root Cause: Arcconf collects only limited amount of UART log as part of support archive.
  - Fix: Added changes for Arcconf to collect as max limit of 2 MB UART log if available as part of support archive.
  - Risk: Low
- Fixed an issue where maxView standalone mode is not working with silent installation.
  - Root Cause: Silent installation for standalone mode of maxView is missing in Linux.
  - Fix: Added changes for silent installation for standalone mode of maxView is missing in Linux.
  - Risk: Low
- Fixed an issue where Arcconf doesn't display active spare as part of logical device segment information after reboot.
  - Root Cause: The condition for display of the spare association with logical device is not present after reboot.
  - Fix: Added changes for retrieving the spare association to logical device information for display.
  - Risk: Low
- Fixed an issue where the progress task display was improper in maxView.
  - Root Cause: Mapping of hostname of the machine was not proper while updating the progress task information to the UI resulting in invalid progress information on the screen.
  - Fix: Added changes for mapping the valid hostname to retrieve proper data for progress task for display.
  - Risk: Low

- Fixed issues where SNMP traps are not displayed with proper information from maxView SNMP agent.
  - Root Cause: SNMP traps was not displaying the source of the trap and mismatched event level.
  - Fix: Added changes for mapping the valid event levels and the source of the event for SNMP traps.
  - Risk: Low

2.3 Limitations

2.3.1 Firmware Limitations

2.3.1.1 Limitations for Firmware Release 3.21 B0

This release includes the following firmware limitations:

- The controller will lockup with code 0x1E10 when a drive with unsupported block size (for example a T10 DIF drive) is attached to slot 0.
  - Workaround: Connect the unsupported block size drive into any other slots than 0.
- A corner case scenario where all but one of the controller connectors are in HBA mode and changing the last connector from Mixed/RAID mode to HBA mode, will cause a lockup with TLB exception (NULL access).
  - Workaround: None
- If redundant data cannot be regenerated during a host write request on a degraded logical volume due to bad blocks on all data drives, the respective LBA will still be marked bad, but it won't be returned with error status. A subsequent read to these LBAs will result in a medium error with sense data (KCQ 03/11/00) because the block is already marked bad by firmware.
  - Workaround: None
- SATA drives attached to a non-Microchip expander may get into a failed state when upgrading the controller firmware from previous releases to this release due to the expander not clearing STP affiliation.
  - Workaround: Power cycle the expanders to clear the STP affiliation.
- A rare corner case scenario where the controller may hang during expander firmware update on multi-level expander/SEP device topology along with I/Os.
  - Workaround: Perform expander firmware update without IOs.
- Controller cache will not be converted into 100% read cache, if any backup power source cable error, charge or charge timeout error occurs when expansion or transformation task is active.
  - Workaround: None

2.3.1.2 Limitations for Firmware Release 1.32 Build 0

- Firmware release 1.32b0 may become unresponsive while attempting to flash firmware or execute other RAID logical volume operations.
  - Description: Refer to entry "Fixed an issue where firmware may become unresponsive while attempting to flash firmware or execute other RAID logical volume operations" in the Firmware fixes section.
  - A fix for this issue is available in the 1.60 B0 firmware release. If a firmware flash failure is occurring, try the following workarounds:
    - Workaround: If there are no target devices (expanders or drives) attached to the controller, attach a target device to the controller and try the host management operation again.
    - Workaround: If the system is operating using UEFI, the HII tool can be used to flash the firmware to this release as outlined in the *Microsemi SmartIOC 2100/SmartROC 3100 Installation and User's Guide (ESC-2170577),* appendix entry “Updating the SmartIOC 2100/SmartROC 3100 Controller Firmware”.

- Workaround: If there are target devices attached to the controller and this issue occurs or none of the workarounds can be used, contact Microsemi Support.

2.3.2 UEFI Limitations

2.3.2.1 Limitations for UEFI Build 1.3.11.1/Legacy BIOS Build 1.3.11.3

There are no known limitations for this release.

2.3.3 Driver Limitations

2.3.3.1 Limitations for Linux Driver Build 1.2.16-040

This release includes the following Linux limitations:

- This release includes the following limitation when doing a driver injection (DUD) install. On some distributions (RHEL8.2, SLES15 SP2), the DUD install will hang if a drive in HBA mode has the Drive Write Cache enabled.
  - Workaround: There are two workarounds:
    - Make sure the Drive Write Cache is disabled for any drive in HBA mode.
    - For RHEL 8.2, add `rd.driver.blacklist=smartpqi` to the grub entry along with `inst.dd`.

2.3.3.2 Limitations for Windows Driver Builds 106.190.4.1062

There are no known limitations for this release.

2.3.3.3 Limitations for FreeBSD Driver Build 4030.0.101

There are no known limitations for this release.

2.3.3.4 Limitations for Solaris Driver Build 4030.0.101

This release includes the following Solaris limitations:

- In Solaris, if the logical drive capacity is changed, the following workaround has to be done in order to reflect the new size.
  - Workaround:
    1. From the format menu select the disk (logical drive undergone size change) then type and autoconfigure and then quit.
    2. Run `devfsadm -c disk`.

2.3.3.5 Limitations for VMware Driver Build 4030.0.101

There are no known limitations in this release.

2.3.4 Hardware Limitations

This release includes the following hardware limitations:

- Two Wire Interface (TWI) address conflicts can cause system DDR memory to not be discovered.

- Description: The SmartRAID 3100 and SmartHBA 2100 boards include two TWI targets on the host-facing SMBUS interface with the following slave addresses:
  - 0xA0 – Field Replaceable Unit (FRU) SEEPROM
  - 0xDE – PBSI (default)

    According to the JEDEC specification, the default TWI addresses for the DDR SPD is 0xA0-0xAE (the spec uses 7 bit addressing which is 0x50-0x57). On platform system board designs with SMBUS wiring that has both PCIe slots and DDR slots shared on the same TWI bus, the TWI devices for the DDR and Smart controller are exposed to address conflicts which can result in the system memory not being discovered. The Smart controller PBSI interface defaults to a value of 0xDE (0x6F in 7-bit addressing) and is not a problem unless it is changed to an address that conflicts with the JEDEC defined values. The Smart controller FRU SEEPROM is hardwired to 0xA0.
- Workaround: None available. If this issue is encountered, contact your Microsemi support engineer to determine the next steps for your system.
- Performance with workaround: Not applicable
- Performance without workaround: Not applicable

2.3.5 Management Software Limitations

2.3.5.1 Limitations for Arcconf/maxView Build B23821

This release includes the following limitations:

- In a system where two or more Smart controllers are connected, the listing order of the controller was not predictable when user installs different versions of operating systems on the same system.
  - Workaround: In order to address this issue, the controllers are listed based on the physical slot IDs. Due to this change, the controller listing order in multi-controller system may change from the current release onwards.

3 Updating the Board Firmware for PQI Operation

This section describes how to update the board's firmware components to the latest release.

3.1 Updating Controllers to latest (PQI) Firmware

This procedure describes how to prepare your board to be programmed with the latest board PQI firmware.

Note:

1. If the running firmware is older than 1.98 and a transformation is in progress, complete the transformation before proceeding with the following steps to upgrade the firmware.
2. Complete these procedures exactly as described for proper functionality. If you do not follow all of the steps correctly, you could encounter unusual runtime behavior.

Flashing the board to the latest PQI firmware:

This section describes how to update all the firmware components on SmartHBA 2100 controller boards to the latest release.

If the controller is currently running 1.60 b0 firmware or newer, follow these steps:

1. **Mandatory:** Flash the target with the provided " SmartFWx100.bin" image with arcconf/maxView software.
2. **Mandatory:** Use the OS shutdown/restart operation to gracefully reboot the system to complete the firmware update process.

   Note:

   After completing the firmware update, if the firmware version is still showing the prior version, retry the firmware update steps.

If the controller is currently running 1.32 b0 firmware, follow these steps:

1. **Mandatory:** Flash the target with the provided “SmartFWx100.bin" image with arcconf/maxView software.
   - If the arcconf/maxView software becomes unresponsive or hangs then power cycle the system to recover and refer to firmware limitation section *Limitations for Firmware Release 1.32 Build 0* on page 25.
2. **Mandatory:** If flashing completes, use the OS shutdown/restart operation to gracefully reboot the system to complete the firmware update process.

   Note:

   After completing the firmware update, if the firmware version is still showing the prior version, retry the firmware update steps.

If the controller is currently running 1.04 b0 firmware, follow these steps:

1. **Mandatory:** Flash the controller with the provided “SmartFWx100_ v1.29_b314.bin" image with arcconf/maxView software.
2. **Mandatory:** Reboot the system to refresh all components**.**
3. **Mandatory**: Flash the target with the provided " SmartFWx100.bin" image with arcconf/maxView software.

4. **Mandatory**: Use the OS shutdown/restart operation to gracefully reboot the system to complete the firmware update process.

At this point, the controller would be updated and would be ready to use. Install the SmartPQI driver and the latest version of the Arcconf/maxView management utility to monitor and configure the controller.

Note: Downgrading firmware could lead to unexpected behavior due to an incompatibility in SEEPROMs between this release and the prior release.

4 Installing the Drivers

See the "Microsemi Adaptec® SmartRAID 3100 Series and SmartHBA 2100 Series Host Bus Adapters Installation and User's Guide (ESC-2171547)" for complete driver installation instructions.

Microsemi
2355 W. Chandler Blvd.
Chandler, AZ 85224 USA

Within the USA: +1 (480) 792-7200
Fax: +1 (480) 792-7277

Microsemi Corporation, a subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), and its corporate affiliates are leading providers of smart, connected and secure embedded control solutions. Their easy-to-use development tools and comprehensive product portfolio enable customers to create optimal designs which reduce risk while lowering total system cost and time to market. These solutions serve more than 120,000 customers across the industrial, automotive, consumer, aerospace and defense, communications and computing markets. Headquartered in Chandler, Arizona, the company offers outstanding technical support along with dependable delivery and quality. Learn more at **www.microsemi.com**.

ESC-2161026