[MS-SMB2]: Server Message Block (SMB) Protocol Versions 2 and 3

This topic lists the Errata found in [MS-SMB2] since it was last published. Since this topic is updated frequently, we recommend that you subscribe to these RSS or Atom feeds to receive update notifications.

Errata are subject to the same terms as the Open Specifications documentation referenced.

RSS

Atom

To view a PDF file of the errata for the previous versions of this document, see the following ERRATA Archives:

October 16, 2015 - Download

June 30, 2015 - Download

July 18, 2016 - Download

September 26, 2016 - Download

March 20, 2017 - Download

June 1, 2017 - Download

September 15, 2017 - Download

December 1, 2017 - Download

March 16, 2018 - Download

Errata below are for Protocol Document Version V55.0 – 2018/03/16.

Errata Published*	**Description**
2018/09/03	In Section 3.3.5.15, Receiving an SMB2 IOCTL Request, product behavior note 317 has been changed from: <317> Section 3.3.5.15: Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, Windows 10, Windows Server 2016, and Windows Server operating system allow only the CtlCode values, as specified in section 2.2.31, and the following CtlCode values, as specified in [MS-FSCC] section 2.3. … Windows 10, Windows Server 2016, and Windows Server operating system allow the additional CtlCode value, as specified in [MS-FSCC]. FSCTL name | FSCTL function number FSCTL_DUPLICATE_EXTENTS_TO_FILE | 0x98344 FSCTL_DUPLICATE_EXTENTS_TO_FILE_EX | 0x983e8 Changed to: <317> Section 3.3.5.15: Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, Windows 10, Windows Server 2016, and Windows Server operating

Errata Published*	Description
	system allow only the CtlCode values, as specified in section 2.2.31, and the following CtlCode values, as specified in [MS-FSCC] section 2.3. … Windows 10, Windows Server 2016, and Windows Server operating system allow the additional CtlCode value, as specified in [MS-FSCC]. FSCTL name | FSCTL function number FSCTL_DUPLICATE_EXTENTS_TO_FILE | 0x98344 Windows 10 v1803 operating system and later and Windows Server v1803 operating system and later allow the additional CrlCode value, as specified in [MS-FSCC]. FSCTL name | FSCTL function number FSCTL_DUPLICATE_EXTENTS_TO_FILE_EX | 0x983e8
2018/09/03	In Section 3.3.2.1, Oplock Break Acknowledgment Timer, the following has been changed from: This timer controls the amount of time the server waits for an oplock break acknowledgment from the client (as specified in section 2.2.24) after sending an oplock break notification (as specified in section 2.2.23) to the client. The server MUST wait for an interval of time greater than or equal to the oplock break acknowledgment timer. This timer MUST be smaller than the client Request Expiration time, as specified in section 3.2.6.1.<176> If the server implements the SMB 2.1 or SMB 3.x dialect family, this timer MUST also be used to control the time a server waits for a Lease Break Acknowledgment from the client (as specified in section 2.2.24.2). <169> Section 3.3.2.1: This timer has a default value of 35 seconds, but its value could be changed by system policy to any range between 5 seconds and infinite (4,294,967,295 seconds). Changed to: This timer controls the amount of time the server waits for an oplock break acknowledgment from the client (as specified in section 2.2.24.1) after sending an oplock break notification (as specified in section 2.2.23.1) to the client. The server MUST wait for an interval of time greater than or equal to the oplock break acknowledgment timer. This timer MUST be smaller than the client Request Expiration time, as specified in section 3.2.6.1.<169> <169> Section 3.3.2.1: Windows SMB2 servers set this timer to 35 seconds. In Section 3.3.6.1, Oplock Break Acknowledgment Timer Event, the following has been changed from: The oplock break acknowledgment timer MUST be started when the server sends an SMB2 OPLOCK_BREAK Notification as specified in section 2.2.23 to the client as a result of the underlying object store indicating an oplock break or lease break on a file. When the oplock break acknowledgment timer expires, the server MUST scan for oplock breaks that have not been acknowledged by the client within the configured time. It does this by enumerating all opens in the GlobalOpenTable. For each open, if Open.OplockState is Breaking and Open.OplockTimeout is earlier than the current time, the server MUST acknowledge the oplock break to the underlying object store represented by Open.LocalOpen, set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE, and set Open.OplockState to None.

Errata Published*	Description
	If Open.Connection.Dialect is "2.1" or belongs to the SMB 3.x dialect family, and the server supports leasing, the server MUST scan for lease breaks that have not been acknowledged by the client within the configured time. It does this by enumerating all lease tables in GlobalLeaseTableList. For each lease table, it enumerates all leases in LeaseTable.LeaseList. For each lease, if Lease.Breaking is TRUE and Lease.LeaseBreakTimeout is earlier than the current time, the server MUST acknowledge the lease break to the underlying object store represented by the opens in Lease.LeaseOpens, and set Lease.LeaseState to NONE. The timer MUST then be restarted to expire again at the time of the next oplock time-out. If no other opens have Open.OplockState equal to Breaking, and no leases (if implemented) have Lease.Breaking set to TRUE, the timer MUST NOT be restarted. Changed to: The oplock break acknowledgment timer MUST be started when the server sends an oplock break notification, as specified in section 2.2.23.1, to the client as a result of the underlying object store indicating an oplock break on a file. When the oplock break acknowledgment timer expires, the server MUST scan for oplock breaks that have not been acknowledged by the client within the configured time. It does this by enumerating all opens in the GlobalOpenTable. For each open, if Open.OplockState is Breaking and Open.OplockTimeout is earlier than the current time, the server MUST acknowledge the oplock break to the underlying object store represented by Open.LocalOpen with SMB2_OPLOCK_LEVEL_NONE as the new oplock level, and MUST set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE, and Open.OplockState to None. The timer MUST be restarted if there is an open where Open.OplockState is equal to “Breaking”. In Section 3.3.4.7, Object Store Indicates a Lease Break, the following has been changed from: If the server succeeds in sending the message on any Open.Connection associated with this Lease, the server MUST start the oplock break acknowledgment timer as specified in section 3.3.2.5. Changed to: If the server succeeds in sending the Lease Break Notification, the server MUST set Lease.BreakNotification to empty and MUST start the lease break acknowledgment timer as specified in section 3.3.2.5. In the same section, the following has been changed from: The server then MUST construct an oplock break response using the syntax specified in section 2.2.25 with the following value: Changed to: The server then MUST construct an oplock break response using the syntax specified in section 2.2.25.1 with the following value: In Section 3.3.5.22.2, Processing a Lease Acknowledgment, the following has been changed from:

Errata Published*	Description
	The server then MUST construct a lease break response using the syntax specified in section 2.2.25 with the following values: Changed to: The server then MUST construct a lease break response using the syntax specified in section 2.2.25.2 with the following values: The following two sections have been added: 3.3.2.5 Lease Break Acknowledgment Timer If the server implements the SMB 2.1 or SMB 3.x dialect family and supports leasing, this timer controls the amount of time the server waits for a Lease Break acknowledgment from the client, as specified in section 2.2.24.2, after sending a lease break notification, as specified in section 2.2.23.2, to the client. The server MUST wait for an interval of time greater than or equal to the lease break acknowledgment timer. This timer MUST be smaller than the client Request Expiration time, as specified in section 3.2.6.1.<172> <172> Section 3.3.2.5: Windows SMB2 servers set this timer to 35 seconds. 3.3.6.5 Lease Break Acknowledgment Timer Event The Lease Break acknowledgment timer MUST be started when the server sends a lease break notification, as specified in section 2.2.23.2, to the client as a result of the underlying object store indicating a lease break on a file. When the lease break acknowledgment timer expires, the server MUST scan for lease breaks that have not been acknowledged by the client within the configured time. It does this by enumerating all lease tables in GlobalLeaseTableList. For each lease table, it enumerates all leases in LeaseTable.LeaseList. For each lease, if Lease.Breaking is TRUE and Lease.LeaseBreakTimeout is earlier than the current time, the server MUST acknowledge the lease break to the underlying object store represented by the opens in Lease.LeaseOpens with NONE as the new lease state and MUST set Lease.LeaseState to NONE and Lease.Breaking to FALSE. The timer MUST be restarted if there is a lease where Lease.Breaking is set to TRUE.
2018/09/03	In Section 3.3.1.12, Per Lease, the following has been added: ● Lease.BreakNotification: A Lease Break Notification, as specified in section 2.2.23.2, if any, to be sent to the client. In Section 3.3.4.7, Object Store Indicates a Lease Break, the processing rules have been changed from: The underlying object store indicates the breaking of a lease by specifying the ClientGuid, the ClientLeaseId, and the new lease state. The new lease state MUST be one of NONE, R, RW, and RH. When the underlying object store indicates the lease break, the server MUST locate the Lease Table by performing a lookup in GlobalLeaseTableList using the provided ClientGuid as the lookup key, and then locate the Lease entry by performing a lookup in the LeaseTable.LeaseList using the provided ClientLeaseId as the lookup key. If no entry is found, the server MUST NOT generate a Lease Break Notification. Instead, the server MUST complete the lease break call from the underlying object store with "NONE" as the new lease state, and take no further action. If a Lease entry is found, the server MUST check the state of Open.Connection for all Opens in Lease.LeaseOpens. If Open.Session.Connection.Dialect belongs to the SMB

Errata Published*	Description
	3.x dialect family and Open.Connection is NULL, the server MUST select an alternate connection in Open.Session.ChannelList and update Open.Connection. If Open.Connection is NULL, Open.IsResilient is FALSE and Open.IsPersistent is FALSE, the server MUST close the Open as specified in section 3.3.4.17 for the following cases: ● Open.IsDurable is FALSE. ● Lease.BreakToLeaseState does not contain SMB2_LEASE_HANDLE_CACHING and Open.IsDurable is TRUE. If Lease.LeaseOpens is empty, the server MUST NOT generate a Lease Break Notification. Instead, the server MUST complete the lease break call from the underlying object store with "NONE" as the new lease state, set Lease.LeaseState to "NONE", and take no further action. If Lease.LeaseOpens is not empty, the server MUST construct a Lease Break Notification (section 2.2.23.2) message to send to the client. The server MUST set the Command field in the SMB2 header to SMB2 OPLOCK_BREAK, and the MessageId field to 0xFFFFFFFFFFFFFFFF. The server MUST set the SessionId and TreeId fields in the SMB2 header to 0. If Lease.LeaseState is SMB2_LEASE_READ_CACHING, the server MUST set the Flags field of the message to zero and MUST set Open.OplockState to “None” for all opens in Lease.LeaseOpens. The server MUST set Lease.Breaking to FALSE, and the LeaseKey field MUST be set to Lease.LeaseKey. Otherwise, the server MUST set the Flags field of the message to SMB2_NOTIFY_BREAK_LEASE_FLAG_ACK_REQUIRED, indicating to the client that lease acknowledgment is required. The LeaseKey field MUST be set to Lease.LeaseKey. The server MUST set Open.OplockState to “Breaking” for all Opens in Lease.LeaseOpens. The server MUST set the CurrentLeaseState field of the message to Lease.LeaseState, set Lease.Breaking to TRUE, set Lease.BreakToLeaseState to the new lease state indicated by the object store, and set Lease.LeaseBreakTimeout to the current time plus an implementation-specific<194> default value in milliseconds. If the server implements the SMB 3.x dialect family and Lease.Version is 2, the server MUST set NewEpoch to Lease.Epoch + 1. Otherwise, NewEpoch MUST be set to zero. The SMB2 Lease Break Notification is sent to the client using the connection specified in Open.Connection of the first Open in Lease.LeaseOpens. The message SHOULD NOT be signed. If the server fails to send the message to the client, the server MUST retry the send using the connection specified in Open.Connection of the next Open in Lease.LeaseOpens. If the server succeeds in sending the message on any Open.Connection associated with this Lease, the server MUST start the oplock break acknowledgment timer as specified in section 3.3.2.5. Otherwise, the server MUST perform the following steps: ● If Open.IsPersistent is TRUE, and Lease.LeaseState is not SMB2_LEASE_READ_CACHING, and Open.DurableOpenTimeout is not earlier than the current time, the server MUST take no further action. ● Otherwise, the server MUST set Open.Lease.Breaking to FALSE, set Lease.Held to FALSE, and MUST complete the lease break call from the underlying object store with "NONE" as the new lease state. Changed to: The underlying object store indicates the breaking of a lease by specifying the ClientGuid, the ClientLeaseId, and the new lease state. The new lease state MUST be one of NONE, R, RW, and RH. When the underlying object store indicates the lease break, the server MUST locate the Lease Table by performing a lookup in GlobalLeaseTableList using the provided ClientGuid as the lookup key, and then locate the Lease entry by performing a

Errata Published*	Description
	lookup in the LeaseTable.LeaseList using the provided ClientLeaseId as the lookup key. If no entry is found, the server MUST complete the lease break call from the underlying object store with "NONE" as the new lease state, set Lease.LeaseState to "NONE", and take no further action. If a Lease entry is found, the server MUST perform the following: If Lease.LeaseOpens is empty, the server MUST complete the lease break call from the underlying object store with "NONE" as the new lease state, set Lease.LeaseState to "NONE", and take no further action. Otherwise, for each Open in Lease.LeaseOpens, if Open.Connection is NULL, Open.IsResilient is FALSE and Open.IsPersistent is FALSE, the server MUST close the Open as specified in section 3.3.4.17 for the following cases: ● Open.IsDurable is FALSE. ● Lease.BreakToLeaseState does not contain SMB2_LEASE_HANDLE_CACHING and Open.IsDurable is TRUE. If Lease.LeaseOpens is not empty, the server MUST construct a Lease Break Notification (section 2.2.23.2) message to send to the client. The server MUST set the Command field in the SMB2 header to SMB2 OPLOCK_BREAK, and the MessageId field to 0xFFFFFFFFFFFFFFFF. The server MUST set the SessionId and TreeId fields in the SMB2 header to 0. If Lease.LeaseState is SMB2_LEASE_READ_CACHING, the server MUST set the Flags field of the message to zero and MUST set Open.OplockState to "None" for all opens in Lease.LeaseOpens. The server MUST set Lease.Breaking to FALSE, and the LeaseKey field MUST be set to Lease.LeaseKey. Otherwise, the server MUST set the Flags field of the message to SMB2_NOTIFY_BREAK_LEASE_FLAG_ACK_REQUIRED, indicating to the client that lease acknowledgment is required. The LeaseKey field MUST be set to Lease.LeaseKey. The server MUST set Open.OplockState to "Breaking" for all Opens in Lease.LeaseOpens. The server MUST set the CurrentLeaseState field of the message to Lease.LeaseState, set Lease.Breaking to TRUE, set Lease.BreakToLeaseState to the new lease state indicated by the object store, and set Lease.LeaseBreakTimeout to the current time plus an implementation-specific<194> default value in milliseconds. If the server implements the SMB 3.x dialect family and Lease.Version is 2, the server MUST set NewEpoch to Lease.Epoch + 1. Otherwise, NewEpoch MUST be set to zero. The message SHOULD NOT be signed. The server MUST set Lease.BreakNotification to the newly constructed Lease Break Notification. The server MUST look up all the connections in ConnectionList where Connection.ClientGuid matches the provided ClientGuid. The server MUST send Lease.BreakNotification using the first available connection. If the server fails to send the notification to the client, the server MUST retry the send using an alternate connection available. If the server succeeds in sending the Lease Break Notification, the server MUST set Lease.BreakNotification to empty and MUST start the lease break acknowledgment timer as specified in section 3.3.2.5. Otherwise, the server MUST perform the following steps: ● If Open.IsPersistent is TRUE and Lease.LeaseState is not SMB2_LEASE_READ_CACHING, the server MUST take no further action. ● Otherwise, the server MUST set Open.Lease.Breaking to FALSE, Lease.Held to FALSE, Open.OplockState to None, Lease.BreakNotification to empty, and MUST complete the lease break call from the underlying object store with "NONE" as the new lease state. In Section 3.3.5.9.7, Handling the SMB2_CREATE_DURABLE_HANDLE_RECONNECT Create Context, the following has been removed:

Errata Published*	Description
	• If Open.IsPersistent is TRUE and the SMB2_DHANDLE_FLAG_PERSISTENT bit is set in the Flags field of the SMB2_CREATE_DURABLE_HANDLE_RECONNECT Create Context request, Open.Lease.Breaking MUST be set to TRUE and the SMB2_LEASE_FLAG_BREAK_IN_PROGRESS bit MUST be set in the Flags field of the response. The server MUST send Lease Break Notification to the client as specified in section 3.3.4.7. In that same section, the following has been added: If Open.IsPersistent is TRUE, Open.Lease.Breaking is TRUE, and Open.Lease.BreakNotification is not empty, the server MUST send Open.Lease.BreakNotification to the client over an available connection in ConnectionList where Open.ClientGuid matches Connection.ClientGuid. If the server succeeds in sending the notification, the server MUST set Open.Lease.BreakNotification to empty and MUST start the lease break acknowledgment timer as specified in section 3.3.2.5. In Section 3.3.5.9.12, Handling the SMB2_CREATE_DURABLE_HANDLE_RECONNECT_V2 Create Context, the following has been removed: • If Open.IsPersistent is TRUE and the SMB2_DHANDLE_FLAG_PERSISTENT bit is set in the Flags field of the SMB2_CREATE_DURABLE_HANDLE_RECONNECT_V2 Create Context request, Open.Lease.Breaking MUST be set to TRUE and the SMB2_LEASE_FLAG_BREAK_IN_PROGRESS bit MUST be set in the Flags field of the response. The server MUST send Lease Break Notification to the client as specified in section 3.3.4.7. In that same section, the following has been added: If Open.IsPersistent is TRUE, Open.Lease.Breaking is TRUE, and Open.Lease.BreakNotification is not empty, the server MUST send Open.Lease.BreakNotification to the client over an available connection in ConnectionList where Open.ClientGuid matches Connection.ClientGuid. If the server succeeds in sending the notification, the server MUST set Open.Lease.BreakNotification to empty and MUST start the lease break acknowledgment timer as specified in section 3.3.2.5. In Section 3.3.5.22.2, Processing a Lease Acknowledgment, the following has been changed from: The server completes the lease break request received from the object store as described in section 3.3.4.7. The server MUST set Lease.LeaseState to LeaseState received in the request, and MUST set Lease.Breaking to FALSE. Changed to: The server completes the lease break request received from the object store as described in section 3.3.4.7. The server MUST set Lease.LeaseState to LeaseState received in the request, Open.OplockState to “Held”, and Lease.Breaking to FALSE.
2018/08/20	In Section 3.3.4.6, Object Store Indicates an Oplock Break, the following has been changed from: If an entry is found, the server MUST check the state of Open.Connection. If Open.Session.Connection.Dialect belongs to the SMB 3.x dialect family and

Errata Published*	Description
	Open.Connection is NULL, the server MUST select an alternate connection in Open.Session.ChannelList and update Open.Connection. If Open.Connection is NULL, Open.IsResilient is FALSE, Open.IsDurable is FALSE and Open.IsPersistent is FALSE, the server SHOULD close the Open as specified in section 3.3.4.17. If Open.Connection is not NULL, the server MUST construct an Oplock Break Notification following the syntax specified in section 2.2.23.1 to send back to the client. The server MUST set the Command in the SMB2 header to SMB2 OPLOCK_BREAK, and the MessageId to 0xFFFFFFFFFFFFFFFF. The server SHOULD<191> set the SessionId in the SMB2 header to Open.Session.SessionId. The server MUST set the TreeId in the SMB2 header to zero. The FileId field of the response structure MUST be set to the values from the Open structure, with the volatile part set to Open.FileId and the persistent part set to Open.DurableFileId. The oplock Level of the response MUST be set to the value provided by the object store. The server MUST set Open.OplockState to Breaking and set Open.OplockTimeout to the current time plus an implementation-specific default value in milliseconds.<192> The SMB2 Oplock Break Notification is sent to the client. The message SHOULD NOT be signed. The server MUST start the oplock break acknowledgment timer as specified in section 3.3.2.1. Changed to: If an entry is found, the server MUST perform the following: If Open.Connection is NULL, Open.IsResilient is FALSE, Open.IsDurable is FALSE and Open.IsPersistent is FALSE, the server SHOULD close the Open as specified in section 3.3.4.17. The server MUST construct an Oplock Break Notification following the syntax specified in section 2.2.23.1 to send back to the client. The server MUST set the Command in the SMB2 header to SMB2 OPLOCK_BREAK, and the MessageId to 0xFFFFFFFFFFFFFFFF. The server SHOULD<191> set the SessionId in the SMB2 header to Open.Session.SessionId. The server MUST set the TreeId in the SMB2 header to zero. The FileId field of the response structure MUST be set to the values from the Open structure, with the volatile part set to Open.FileId and the persistent part set to Open.DurableFileId. The oplock Level of the response MUST be set to the value provided by the object store. The server MUST set Open.OplockState to Breaking and set Open.OplockTimeout to the current time plus an implementation-specific default value in milliseconds.<192> The message SHOULD NOT be signed. If the server implements the SMB 3.x dialect family, SMB2 Oplock Break Notification MUST be sent to the client using the first available connection in Open.Session.ChannelList where Channel.Connection is not NULL. If the server fails to send the notification to the client, the server MUST retry the send using an alternate connection, if available, in Open.Session.ChannelList. Otherwise, SMB2 Oplock Break Notification MUST be sent to the client using Open.Connection. If the notification could not be sent on any connection, the server MUST complete the oplock break from the underlying object store with SMB2_OPLOCK_LEVEL_NONE as the new oplock level and MUST set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE and Open.OplockState to None. If the server succeeds in sending the notification, the server MUST start the oplock break acknowledgment timer as specified in section 3.3.2.1.
2018/08/20	In Section 3.2.4.2, Application Requests a Connection to a Share, the following has been changed from: SpecifiedDialect: An optional dialect to be negotiated. If provided by the application, SpecifiedDialect matches the Connection.Dialect.

Errata Published*	Description
	Changed to: SpecifiedDialects: An optional list of dialects to be negotiated. If provided, this MUST be one or more values as specified in Dialects field of SMB2 NEGOTIATE Request in section 2.2.3. If provided by the application, the highest dialect in the SpecifiedDialects matches the Connection.Dialect. In Section 3.2.4.2.2.2, SMB2-Only Negotiate, the following has been changed from: If the application provided a dialect in SpecifiedDialect, the client MUST do the following: Set the DialectCount to 1. Set the value in Dialects[0] array to SpecifiedDialect. Changed to: If the application has provided SpecifiedDialects, the client MUST do the following: Set the DialectCount to number of elements in the SpecifiedDialects. Set the value in Dialects array to the values in SpecifiedDialects. In Section 3.2.5.2, Receiving an SMB2 NEGOTIATE Response, the following has been added: If the DialectRevision field in the SMB2 NEGOTIATE Response is equal to one of the values in the Dialects field of the SMB2 NEGOTIATE request, the client MUST set Connection.Dialect to DialectRevision. Otherwise, the client MUST close the connection and SHOULD fail the application request. The following has been changed from: If the DialectRevision in the SMB2 NEGOTIATE Response is 0x02FF, the client MUST issue a new SMB2 NEGOTIATE request as described in section 3.2.4.2.2.2 with the only exception that the client MUST allocate sequence number 1 from Connection.SequenceWindow, and MUST set MessageId field of the SMB2 header to 1. Otherwise, the client MUST proceed as follows. Changed to: If the DialectRevision field in the SMB2 NEGOTIATE Response is 0x02FF, the client MUST issue a new SMB2 NEGOTIATE request as described in section 3.2.4.2.2.2 with the only exception that the client MUST allocate sequence number 1 from Connection.SequenceWindow, and MUST set MessageId field of the SMB2 header to 1. Otherwise, the client MUST proceed as follows. The following has been removed: The client MUST set Connection.Dialect to DialectRevision in the SMB2 NEGOTIATE Response.

Errata Published*	Description
2018/07/16	In Section 1.2.2, Informative References, the following reference was removed: [MSKB-978491] Microsoft Corporation, "FIX: A server that is running Server Message Block Version 2 does not respond to certain FSCTL_SRV_NOTIFY_TRANSACTION requests from clients that are running Windows Vista or Windows Server 2008", 2011, In Section 3.3.5.15, Receiving an SMB2 IOCTL Request, a product behavior note was changed from: The server SHOULD<317> fail the request with STATUS_NOT_SUPPORTED when an FSCTL is not allowed on the server, and SHOULD<318> fail the request with STATUS_INVALID_DEVICE_REQUEST when the FSCTL is allowed, but is not supported on the file system on which the file or directory handle specified by the FSCTL exists, as specified in [MS-FSCC] section 2.2. <318> Section 3.3.5.15: Windows Vista SP1 and Windows Server 2008 servers without , and Windows 7 and Windows Server 2008 R2 without Service Pack 1 ignore a FSCTL_SRV_NOTIFY_TRANSACTION request specifying a valid FileId, don't send a response to the client, and reply to a FSCTL_SRV_NOTIFY_TRANSACTION with an invalid or -1 FileId with STATUS_INVALID_PARAMETER. For the following FSCTLs, Windows Vista SP1, Windows Server 2008, Windows 7, and Windows Server 2008 R2 return STATUS_FILE_CLOSED instead of STATUS_INVALID_DEVICE_REQUEST: ● FSCTL_QUERY_NETWORK_INTERFACE_INFO ● FSCTL_DFS_GET_REFERRALS_EX ● FSCTL_VALIDATE_NEGOTIATE_INFO Changed to: The server SHOULD<317> fail the request with STATUS_NOT_SUPPORTED when an FSCTL is not allowed on the server, and SHOULD<318> fail the request with STATUS_INVALID_DEVICE_REQUEST when the FSCTL is allowed, but is not supported on the file system on which the file or directory handle specified by the FSCTL exists, as specified in [MS-FSCC] section 2.2. <318> Section 3.3.5.15: For the following FSCTLs, Windows Vista SP1, Windows Server 2008, Windows 7, and Windows Server 2008 R2 return STATUS_FILE_CLOSED instead of STATUS_INVALID_DEVICE_REQUEST: ● FSCTL_QUERY_NETWORK_INTERFACE_INFO ● FSCTL_DFS_GET_REFERRALS_EX ● FSCTL_VALIDATE_NEGOTIATE_INFO
2018/07/02	In Section 3.3.1.10, Per Open, the following was changed from: ● Open.DurableOpenTimeout: A time value that indicates when a handle that has been preserved for durability will be closed by the system if a client has not reclaimed it. Changed to: ● Open.DurableOpenTimeout: The time the server waits before closing a handle that has been preserved for durability, if a client has not reclaimed it. ● Open.DurableOpenScavengerTimeout: A time stamp value, if non-zero, representing the maximum time to preserve the open for reclaim.

Errata Published*	Description
	In Section 3.3.3, Initialization, the following was added: ● Open.DurableOpenScavengerTimeout MUST be set to zero. In Section 3.3.5.9.6, Handling the SMB2_CREATE_DURABLE_HANDLE_REQUEST Create Context, the following paragraph has been changed from: In the "Successful Open Initialization" phase, if the underlying object store does not grant durability, the server MUST skip the rest of the processing in this phase. Otherwise, the server MUST set Open.IsDurable to TRUE and Open.DurableOwner to a security descriptor accessible only by the user represented by Open.Session.SecurityContext. Changed to: In the "Successful Open Initialization" phase, if the underlying object store does not grant durability, the server MUST skip the rest of the processing in this phase. Otherwise, the server MUST set Open.IsDurable to TRUE and Open.DurableOwner to a security descriptor accessible only by the user represented by Open.Session.SecurityContext and Open.DurableOpenTimeout MUST be set to an implementation specific value<268>. <268> Section 3.3.5.9.6: Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 set Open.DurableOpenTimeout to 16 minutes. Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, Windows 10, Windows Server 2016, and Windows Server set Open.DurableOpenTimeout to 2 minutes. In Section 3.3.5.9.10, Handling the SMB2_CREATE_DURABLE_HANDLE_REQUEST_V2 Create Context, the following was changed from: Open.DurableOpenTimeout SHOULD<276> be set to the Timeout value in the response. Changed to: Open.DurableOpenTimeout MUSTS be set to the Timeout value in the response. The following was removed: <285> Section 3.3.5.9.10: Windows 8 and Windows Server 2012 R2 SMB2 servers set Open.DurableOpenTimeout to 60 seconds. The following was changed from: <275> Section 3.3.5.9.10: If the Timeout value in the request is not zero, Windows 8 and Windows Server 2012 SMB2 servers set Timeout to the Timeout value in the request. <276> Section 3.3.5.9.10: If the Timeout value in the request is zero and Share.CATimeout is not zero, Windows 8 and Windows Server 2012 SMB2 servers set Timeout to Share.CATimeout. If the Timeout value in the request is zero and Share.CATimeout is zero, Windows 8 and Windows Server 2012 SMB2 servers set Timeout to 60 seconds.

Errata Published*	Description
	If the Timeout value in the request is zero, Windows 8.1 and Windows Server 2012 R2 SMB2 servers set Timeout to 180 seconds. Changed to: <275> Section 3.3.5.9.10: If the Timeout value in the request is not zero, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2 SMB2 servers set Timeout to the Timeout value in the request. <276> Section 3.3.5.9.10: If the Timeout value in the request is zero and Share.CATimeout is not zero, Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, Windows 10, Windows Server 2016, and Windows Server SMB2 servers set Timeout to Share.CATimeout. If the Timeout value in the request is zero and Share.CATimeout is zero, Windows 8 and Windows Server 2012 SMB2 servers set Timeout to 60 seconds. In Section 3.3.6.2, Durable Open Scavenger Timer Event, the following was changed from: When the durable open scavenger timer expires, the server MUST scan for durable opens that have not been reclaimed by a client within the configured time. It does this by enumerating all opens in the GlobalOpenTable. For each open, if Open.IsDurable is TRUE, Open.Connection is NULL, and Open.DurableOpenTimeout is earlier than the current time, the server MUST close the open as specified in section 3.3.4.17. Changed to: When the durable open scavenger timer expires, the server MUST scan for durable opens that have not been reclaimed by a client within the configured time. It does this by enumerating all opens in the GlobalOpenTable. For each open, if Open.IsDurable is TRUE, Open.Connection is NULL, and Open.DurableOpenScavengerTimeout is earlier than the system time, the server MUST close the open as specified in section 3.3.4.17. In Section 3.3.7.1, Handling Loss of a Connection, the following was changed from: ● If Open.Connection.Dialect belongs to the SMB 3.x dialect family, and if Open.DurableOpenTimeOut is not zero, the server MUST add the current time to its value. ● Otherwise, the server MUST set Open.DurableOpenTimeOut to the current time plus an implementation-specific default value. <381> <381> Section 3.3.7.1: Windows-based servers set this value to 16 minutes. Changed toL ● The server MUST set Open.DurableOpenScavengerTimeout to the system time plus Open.DurableOpenTimeOut.
2018/07/02	In Section 3.3.5.11, Receiving an SMB2 FLUSH Request, the following was added: If the Open is on a file and Open.GrantedAccess includes neither FILE_WRITE_DATA nor FILE_APPEND_DATA, the server MUST fail the request with STATUS_ACCESS_DENIED.

Errata Published*	Description
	If the Open is on a directory and Open.GrantedAccess includes neither FILE_ADD_FILE nor FILE_ADD_SUBDIRECTORY, the server MUST fail the request with STATUS_ACCESS_DENIED. If Open.IsPersistent is TRUE, the server MUST succeed the operation and MUST respond with an SMB2 FLUSH Response specified in section 2.2.18. The following was changed from: The server MUST issue a request to the underlying object store to flush any cached data for Open.LocalOpen. Changed to: Otherwise, the server MUST issue a request to the underlying object store to flush any cached data for Open.LocalOpen.
2018/06/18	In Section 3.2.4.6, Application Requests Reading from a File or Named Pipe, the bullet list was changed from: Otherwise, the following fields of the request MUST be initialized as follows: ● The Channel field MUST be set to 0. ● The first byte of the Buffer field MUST be set to 0. ● The ReadChannelInfoOffset field MUST be set to 0. ● The ReadChannelInfoLength field MUST be set to 0. Changed to: Otherwise, the following fields of the request MUST be initialized as follows: ● If Connection.Dialect belongs to the SMB 3.x dialect family: ● The Channel field MUST be set to SMB2_CHANNEL_NONE. ● The ReadChannelInfoOffset field MUST be set to 0. ● The ReadChannelInfoLength field MUST be set to 0. ● The first byte of the Buffer field MUST be set to 0. In Section 3.2.4.7, Application Requests Writing to a File or Named Pipe, the following was changed from: ● The DataOffset field is set to the offset from the beginning of the SMB2 header to the data being written. This value SHOULD be 0x70, which is the default offset for write requests. ● If Connection.Dialect is not "2.0.2", and application-supplied WriteThrough is TRUE, the SMB2_WRITEFLAG_WRITE_THROUGH bit in the Flags field MUST be set. ● If Connection.Dialect is "3.0.2" or "3.1.1", and the application-supplied UnbufferedWrite is TRUE, the SMB2_WRITEFLAG_WRITE_UNBUFFERED bit in the Flags field MUST be set. If the number of bytes to write exceeds the Connection.MaxWriteSize, the client MUST split the write into separate write operations no larger than the Connection.MaxWriteSize. The client MAY<111> send these separate writes in any order. If the connection is not established in RDMA mode or if the size of the operation is less than or equal to an implementation-specific threshold <112>or if either Open.TreeConnect.Session.SigningRequired or Open.TreeConnect.Session.EncryptData is TRUE, then

Errata Published*	Description
	<118> Section 3.2.4.7: Windows-based clients always put the payload at the beginning of the Buffer field and do not insert padding. Changed to: ● The DataOffset field MUST be set to an implementation-specific<110> value. ● If Connection.Dialect is not "2.0.2", and application-supplied WriteThrough is TRUE, the SMB2_WRITEFLAG_WRITE_THROUGH bit in the Flags field MUST be set. ● If Connection.Dialect is "3.0.2" or "3.1.1", and the application-supplied UnbufferedWrite is TRUE, the SMB2_WRITEFLAG_WRITE_UNBUFFERED bit in the Flags field MUST be set. If the number of bytes to write exceeds the Connection.MaxWriteSize, the client MUST split the write into separate write operations no larger than the Connection.MaxWriteSize. The client MAY<111> send these separate writes in any order. If the connection is not established in RDMA mode or if the size of the operation is less than or equal to an implementation-specific threshold <112>or if either Open.TreeConnect.Session.SigningRequired or Open.TreeConnect.Session.EncryptData is TRUE, the following fields of the request MUST be initialized as follows: ● If Connection.Dialect belongs to the SMB 3.x dialect family, ● The Channel field MUST be set to SMB2_CHANNEL_NONE. ● The WriteChannelInfoOffset field MUST be set to 0. ● The WriteChannelInfoLength field MUST be set to 0. ● The RemainingBytes field MUST be set to 0. <110> Section 3.2.4.7: Windows-based clients set the DataOffset field to 0x70, which indicates that the payload is always placed at the beginning of the Buffer field. In Section 3.3.5.12, Receiving an SMB2 READ Request, the following was changed from: ● Channel is not equal to SMB2_CHANNEL_RDMA_V1_INVALIDATE, SMB2_CHANNEL_RDMA_V1, or SMB2_CHANNEL_NONE. ● Connection.Dialect is "3.0" and Channel is equal to SMB2_CHANNEL_RDMA_V1_INVALIDATE. ● Channel is equal to SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE and the underlying Connection is not RDMA. ● Channel is equal to SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE and Length or ReadChannelInfoOffset or ReadChannelInfoLength is equal to 0. ● If the server implements the SMB 3.x dialect family, if Connection.Dialect belongs to the SMB 3.x dialect family, and if Channel is equal to SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE, and if any of the following conditions is TRUE, the server MUST fail the request with STATUS_INVALID_PARAMETER. ● Underlying Connection is not RDMA. ● The Length or ReadChannelInfoOffset or ReadChannelInfoLength is equal to 0. Changed to: ● Connection.Dialect is "3.0.2" or “3.1.1” and Channel is not equal to SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_NONE. ● Connection.Dialect is "3.0" and Channel is not equal to SMB2_CHANNEL_RDMA_V1_INVALIDATE.

Errata Published*	Description
	● Channel is equal to SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE and any of the following conditions is TRUE: ● The underlying Connection is not RDMA. ● Length, ReadChannelInfoOffset, or ReadChannelInfoLength is equal to 0. In Section 3.3.5.13, Receiving an SMB2 WRITE Request, the following was changed from: ● Channel is not equal to SMB2_CHANNEL_RDMA_V1_INVALIDATE, SMB2_CHANNEL_RDMA_V1, or SMB2_CHANNEL_NONE. ● Connection.Dialect is "3.0" and Channel is equal to SMB2_CHANNEL_RDMA_V1_INVALIDATE. ● Channel is equal to SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE and the underlying Connection is not RDMA. ● Channel is equal to SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE and Length or DataOffset are not equal to 0. ● Channel is equal to SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE and RemainingBytes or WriteChannelInfoOffset or WriteChannelInfoLength are equal to 0. If Channel is equal to 0 and DataOffset is greater than 0x100, the server MUST fail the request with STATUS_INVALID_PARAMETER. If Channel is equal to 0 a nd the number of bytes received in Buffer is less than (DataOffset + Length), the server MUST fail the request with STATUS_INVALID_PARAMETER. If Connection.SupportsMultiCredit is TRUE, the server MUST validate CreditCharge based on Length, as specified in section 3.3.5.2.5. If the validation fails, it MUST fail the write request with STATUS_INVALID_PARAMETER. Changed to: ● Connection.Dialect is "3.0.2" or "3.1.1" and Channel is not equal to SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_NONE. ● Connection.Dialect is "3.0" and Channel is not equal to SMB2_CHANNEL_RDMA_V1_INVALIDATE. ● Channel is equal to SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE and any of the following conditions is TRUE: ● The underlying Connection is not RDMA. ● Length or DataOffset are not equal to 0. ● RemainingBytes, WriteChannelInfoOffset, or WriteChannelInfoLength are equal to 0. If Channel is equal to SMB2_CHANNEL_NONE and DataOffset is greater than 0x100, the server MUST fail the request with STATUS_INVALID_PARAMETER. If Channel is equal to SMB2_CHANNEL_NONE a nd the number of bytes received in Buffer is less than (DataOffset + Length), the server MUST fail the request with STATUS_INVALID_PARAMETER. The following was removed If Channel is not equal to one of the values specified in section 2.2.21, the server SHOULD<305> consider the Channel field value as SMB2_CHANNEL_NONE and MUST continue processing the request.

Errata Published*	Description
	If Connection.Dialect belongs to the SMB 3.x dialect family and Channel is equal to SMB2_CHANNEL_RDMA_V1 and any of the following conditions is TRUE, the server MUST fail the request with STATUS_INVALID_PARAMETER. ● Underlying Connection is not RDMA. ● RemainingBytes is equal to 0. ● Length or DataOffset is not equal to 0. ● WriteChannelInfoOffset or WriteChannelInfoLength is equal to 0. <305> Section 3.3.5.13: If the Channel value is not equal to 0x00000000 or 0x00000001, Windows Server 2012 fails the request with STATUS_INVALID_PARAMETER. If the Channel value is not equal to 0x00000000, Windows 8 fails the request with STATUS_INVALID_PARAMETER.
2018/06/18	In Section 3.2.4.6, Application Requests Reading from a File or Named Pipe, the following was changed from: ● The Channel field MUST be set to 0. Changed to: ● The Channel field MUST be set to SMB2_CHANNEL_NONE.
2018/06/18	In Section 2.2.21, SMB2 WRITE Request, the descriptions of WriteChannelInfoOffset and WriteChannelInfoLengthwere were changed from: WriteChannelInfoOffset (2 bytes): For the SMB 2.0.2 and 2.1 dialects, this field MUST NOT be used and MUST be reserved. The client MUST set this field to 0, and the server MUST ignore it on receipt. For the SMB 3.x dialect family, it contains the length, in bytes, of the channel data as specified by the Channel field of the request. WriteChannelInfoLength (2 bytes): For the SMB 2.0.2 and SMB 2.1 dialects, this field MUST NOT be used and MUST be reserved. The client MUST set this field to 0, and the server MUST ignore it on receipt. For the SMB 3.x dialect family, it contains the offset, in bytes, from the beginning of the SMB2 header to the channel data as described by the Channel field of the request. Changed to: WriteChannelInfoOffset (2 bytes): For the SMB 2.0.2 and 2.1 dialects, this field MUST NOT be used and MUST be reserved. The client MUST set this field to 0, and the server MUST ignore it on receipt. For the SMB 3.x dialect family, it contains the offset, in bytes, from the beginning of the SMB2 header to the channel data as specified by the Channel field of the request. WriteChannelInfoLength (2 bytes): For the SMB 2.0.2 and SMB 2.1 dialects, this field MUST NOT be used and MUST be reserved. The client MUST set this field to 0, and the server MUST ignore it on receipt. For the SMB 3.x dialect family, it contains the length, in bytes, of the channel data as specified by the Channel field of the request.
2018/06/18	In Section 2.2.19, SMB2 READ Request, the first sentence for the description of SMB2_CHANNEL_RDMA_V1_INVALIDATE was changed from: This flag is not valid for the SMB 2.0.2, 2.1, and 3.0 dialects. Changed to:

Errata Published*	Description
	This flag is not valid for the SMB 3.0 dialect. In Section 2.2.21, SMB2 WRITE Request, the first sentence for the description of SMB2_CHANNEL_RDMA_V1_INVALIDATE was changed from: This flag is not valid for the SMB 2.0.2, 2.1, and 3.0 dialects. Changed to: This flag is not valid for the SMB 3.0 dialect. In Section 3.2.4.6, Application Requests Reading from a File or Named Pipe, the following bullet point was changed from: ● If Connection.Dialect is "3.0", the Channel field of the request MUST be set to SMB2_CHANNEL_RDMA_V1. If Connection.Dialect is "3.0.2" or "3.1.1", the Channel field of the request SHOULD be set to SMB2_CHANNEL_RDMA_V1_INVALIDATE. Changed to: ● If Connection.Dialect is "3.0.2" or "3.1.1" and processing of received remote invalidation is supported as specified in [MS-SMBD] section 3.1.5.8, the Channel field of the request SHOULD be set to SMB2_CHANNEL_RDMA_V1_INVALIDATE. Otherwise, the Channel field of the request MUST be set to SMB2_CHANNEL_RDMA_V1. In Section 3.2.4.7, Application Requests Writing to a File or Named Pipe, the following bullet point was changed from: ● If Connection.Dialect is "3.0", the Channel field of the request MUST be set to SMB2_CHANNEL_RDMA_V1. If Connection.Dialect is "3.0.2" or "3.1.1", the Channel field of the request SHOULD be set to SMB2_CHANNEL_RDMA_V1_INVALIDATE. Changed to: ● If Connection.Dialect is "3.0.2" or "3.1.1" and processing of received remote invalidation is supported as specified in [MS-SMBD] section 3.1.5.8, the Channel field of the request SHOULD be set to SMB2_CHANNEL_RDMA_V1_INVALIDATE. Otherwise, the Channel field of the request MUST be set to SMB2_CHANNEL_RDMA_V1.
2018/06/18	In Section 3.2.5.1.1, Decrypting the Message, a new bullet point was added: For each response in a compounded response, if the SessionId field of SMB2 header is not equal to the SessionId field in the SMB2 TRANSFORM_HEADER, the client SHOULD<139> discard the entire compounded response and stop processing. <139> Section 3.2.5.1.1: Windows 8.1 and Windows Server 2012 R2 do not discard the entire compounded response if SMB2_FLAGS_RELATED_OPERATIONS is set in the Flags field of the SMB2 header of the response. In Section 3.2.5.1.9, Handling Compounded Responses, the following was removed:

Errata Published*	Description
	For the first response: If SMB2_FLAGS_RELATED_OPERATIONS is set in the Flags field of the SMB2 header of the response, the client SHOULD<149> discard the message. If the SessionId field of SMB2 header is not equal to the SessionId field in SMB2 TRANSFORM_HEADER of the response, the client MUST discard the message. For each subsequent response: If SMB2_FLAGS_RELATED_OPERATIONS is not set in the Flags field of the SMB2 header of the response, the client SHOULD<150> discard the message. If the SessionId field of SMB2 header is not equal to the SessionId field in the SMB2 TRANSFORM_HEADER of the response, the client MUST discard the message.
2018/06/18	In Section 3.2.4.1.4, Sending Compounded Requests, the third step was changed from: 3. The client MUST construct the subsequent request as it would do normally. For any subsequent requests the client MUST set SMB2_FLAGS_RELATED_OPERATIONS in the Flags field of the SMB2 header to indicate that it is using the SessionId, TreeId, and FileId supplied in the previous request (or generated by the server in processing that request). The client SHOULD<89> set SessionId to 0xFFFFFFFFFFFFFFFF and TreeId to 0xFFFFFFFF, and SHOULD<90> set FileId to { 0xFFFFFFFFFFFFFFFF, 0xFFFFFFFFFFFFFFFF }. <89> Section 3.2.4.1.4: Windows-based clients set the SessionId and TreeId fields of subsequent requests with the SessionId and TreeId values of the previous request in the compound chain. <90> Section 3.2.4.1.4: When the Windows-based client compounds a FileId-bearing operation with an SMB2 CREATE request, the FileId field is set to an indeterminate value, which the server ignores as specified in section 3.3.5.2.7.2. Changed to: 3. The client MUST construct the subsequent request as it would do normally. For any subsequent requests the client MUST set SMB2_FLAGS_RELATED_OPERATIONS in the Flags field of the SMB2 header to indicate that it is using the SessionId, TreeId, and FileId supplied in the previous request (or generated by the server in processing that request). For an operation compounded with an SMB2 CREATE request, the FileId field SHOULD be set to { 0xFFFFFFFFFFFFFFFF, 0xFFFFFFFFFFFFFFFF }.
2018/05/07	In Section 2.2.3, SMB2 NEGOTIATE Request, product behavior note <9> was deleted. Changed from: … Dialects (variable): An array of one or more 16-bit integers specifying the supported dialect revision numbers. The array MUST contain at least one of the following values.<9> <9> Section 2.2.3: A Windows Vista RTM–based client would send a value of zero in the Dialects array in SMB2 NEGOTIATE Request and a Windows Vista RTM-based server would acknowledge with a value of 6 in DialectRevision in SMB2 NEGOTIATE Response. This behavior is deprecated. Changed to: …

Errata Published*	Description
	Dialects (variable): An array of one or more 16-bit integers specifying the supported dialect revision numbers. The array MUST contain at least one of the following values. In Section 2.2.4, SMB2 NEGOTIATE Response, product behavior note <14> was deleted. Changed from: … DialectRevision (2 bytes): The preferred common SMB 2 Protocol dialect number from the Dialects array that is sent in the SMB2 NEGOTIATE Request (section 2.2.3) or the SMB2 wildcard revision number. The server SHOULD set this field to one of the following values.<14> <14> Section 2.2.4: A Windows Vista RTM–based client would send a value of zero in the Dialects array in SMB2 NEGOTIATE Request and a Windows Vista RTM–based server would acknowledge with a value of 6 in DialectRevision in SMB2 NEGOTIATE Response. This behavior is deprecated. Changed to: … DialectRevision (2 bytes): The preferred common SMB 2 Protocol dialect number from the Dialects array that is sent in the SMB2 NEGOTIATE Request (section 2.2.3) or the SMB2 wildcard revision number. The server SHOULD set this field to one of the following values. In Section 3.2.4.2.2.1, Multi-Protocol Negotiate, changed from: If the client implements the SMB 2.0.2 dialect, it MUST perform the following: ● the client MUST include the dialect string "SMB 2.002"<101> in the list of dialects, along with any other SMB dialects that it implements. The remaining fields in the request MUST be set up as specified in [MS-SMB] section 3.2.4.2. Otherwise it MUST perform the following: The client MUST include the dialect strings "SMB 2.002" and "SMB 2.???" in the list of dialects, along with any SMB dialects that it implements. The remaining fields in the request MUST be set up as specified in [MS-SMB] section 3.2.4.2. Changed to: If the client implements the SMB 2.0.2 dialect, the client MUST also include the dialect string “SMB 2.002” in the SMB_Data.Bytes.Dialects[] array of the request. If the client implements the SMB 2.1 dialect or SMB 3.x dialect family, the client MUST also include the dialect string “SMB 2.???” in the SMB_Data.Bytes.Dialects[] array of the request. This request MUST be sent to the server. In Section 3.3.5.3.2, SMB 2.0.2 Support, product behavior notes <226> and <227> were removed. Changed from: The server MUST scan the dialects provided for the dialect string "SMB 2.002".<226> If the string is present, the client understands SMB2, and the server

Errata Published*	Description
	MUST respond with an SMB2 NEGOTIATE Response. If the string is not present in the dialect list and the server also implements SMB as specified in [MS-SMB], it MUST terminate SMB2 processing on this connection and start SMB processing on this connection. If the string is not present in the dialect list and the server does not implement SMB, the server MUST disconnect the connection, as specified in section 3.3.7.1, without sending a response. The server MUST set the command of the SMB2 header to SMB2 NEGOTIATE. All other values MUST be set following the syntax specified in section 2.2.1, and any value not defined there with a default MUST be set to 0. The header is followed by an SMB2 NEGOTIATE Response that MUST be constructed as specified in section 2.2.4, with the following specific values: ● SecurityMode MUST have the SMB2_NEGOTIATE_SIGNING_ENABLED bit set. ● If RequireMessageSigning is TRUE, the server MUST also set SMB2_NEGOTIATE_SIGNING_REQUIRED in the SecurityMode. ● DialectRevision MUST be set to 0x0202.<227> …. <226> Section 3.3.5.3.2: When a Windows-based client sends the deprecated "SMB 2.001" dialect, a Windows Vista RTM-based server would acknowledge with a value of 6 in DialectRevision in the SMB2 NEGOTIATE Response. This behavior is deprecated. <227> Section 3.3.5.3.2: A Windows Vista RTM–based server sets DialectRevision to 6. Changed to: The server MUST scan the dialects provided for the dialect string "SMB 2.002". If the string is present, the client understands SMB2, and the server MUST respond with an SMB2 NEGOTIATE Response. If the string is not present in the dialect list and the server also implements SMB as specified in [MS-SMB], it MUST terminate SMB2 processing on this connection and start SMB processing on this connection. If the string is not present in the dialect list and the server does not implement SMB, the server MUST disconnect the connection, as specified in section 3.3.7.1, without sending a response. The server MUST set the command of the SMB2 header to SMB2 NEGOTIATE. All other values MUST be set following the syntax specified in section 2.2.1, and any value not defined there with a default MUST be set to 0. The header is followed by an SMB2 NEGOTIATE Response that MUST be constructed as specified in section 2.2.4, with the following specific values: ● SecurityMode MUST have the SMB2_NEGOTIATE_SIGNING_ENABLED bit set. ● If RequireMessageSigning is TRUE, the server MUST also set SMB2_NEGOTIATE_SIGNING_REQUIRED in the SecurityMode. ● DialectRevision MUST be set to 0x0202. ….
2018/05/07	In Section 6, Appendix A: Product Behavior, product behavior note <96> has been removed. Deleted: <96> Section 3.2.4.2: Windows will establish a new connection for every SMB2 session being created. In Section 6, Appendix A: Product Behavior, product behavior notes <95> and <97> have been changed from:

Errata Published*	Description
	<95> Section 3.2.4.2: Windows will reuse an existing session if the access is by the same logged-on user and the target server name matches exactly. This means that Windows will establish a new session with the same credentials if the same user is logged on to the client multiple times, or if the user is accessing the server through two different names that resolve to the same server. (NetBIOS and fully qualified domain name, for example.) <97> Section 3.2.4.2: Windows establishes a new connection for each new session. Changed to: <95> Section 3.2.4.2: Windows will reuse an existing session only if the access is by the same logged-on user and the Connection.ServerName matches the application-supplied ServerName. <97> Section 3.2.4.2: Windows will reuse the connection to establish a new session, if a connection is available and Connection.ServerName matches the application-supplied ServerName
2018/05/07	In Section 2.2.1.1, SMB2 Packet Header – ASYNC, changed from: NextCommand (4 bytes): For a compounded request, this field MUST be set to the offset, in bytes, from the beginning of this SMB2 header to the start of the subsequent 8-byte aligned SMB2 header. If this is not a compounded request, or this is the last header in a compounded request, this value MUST be 0. Changed to: NextCommand (4 bytes): For a compounded request or response, this field MUST be set to the offset, in bytes, from the beginning of this SMB2 header to the start of the subsequent 8-byte aligned SMB2 header. If this is not a compounded request or response, or this is the last header in a compounded request or response, this value MUST be 0.
2018/05/07	In Section 3.3.5.9.10, Handling the SMB2_CREATE_DURABLE_HANDLE_REQUEST_V2 Create Context, the following has been changed from: The server MUST skip the construction of the SMB2_CREATE_DURABLE_HANDLE_RESPONSE_V2 create context if the SMB2_DHANDLE_FLAG_PERSISTENT bit is not set in the Flags field of the request and if neither of the following conditions is met: Open.OplockLevel is equal to SMB2_OPLOCK_LEVEL_BATCH. Open.Lease.LeaseState has the SMB2_LEASE_HANDLE_CACHING bit set. Changed to: The server MUST skip the construction of the SMB2_CREATE_DURABLE_HANDLE_RESPONSE_V2 create context if the SMB2_DHANDLE_FLAG_PERSISTENT bit is not set in the Flags field of the request and if any of the following conditions is satisfied: Open.FileAttributes includes FILE_ATTRIBUTE_DIRECTORY. Open.OplockLevel is not equal to SMB2_OPLOCK_LEVEL_BATCH and Open.Lease.LeaseState does not contain SMB2_LEASE_HANDLE_CACHING.

Errata Published*	**Description**
	In Section 3.3.5.9.11, Handling the SMB2_CREATE_REQUEST_LEASE_V2 Create Context, the following was changed from: If the FileAttributes field in the request indicates that this operation is on a directory and LeaseState includes SMB2_LEASE_WRITE_CACHING, the server MUST clear the bit SMB2_LEASE_WRITE_CACHING in the LeaseState field. Changed to: If the FileAttributes field in the request includes FILE_ATTRIBUTE_DIRECTORY and LeaseState includes SMB2_LEASE_WRITE_CACHING, the server MUST clear the bit SMB2_LEASE_WRITE_CACHING in the LeaseState field.
2018/05/07	In Section 3.2.4.3.8, Requesting a Lease on a File or a Directory, the third bullet point of the first list has been changed from: If Connection.Dialect is equal to "2.1" and the open is on a directory. Changed to: If Connection.Dialect is equal to "2.1" and the application provided create options includes FILE_DIRECTORY_FILE.
2018/05/07	In Section 2.2.13, SMB2 CREATE Request, the descriptions of FILE_WRITE_THROUGH and FILE_NO_INTERMEDIATE_BUFFERING have been changed from:

Value	Meaning
…	…
FILE_WRITE_THROUGH 0x00000002	The server MUST propagate writes to this open to persistent storage before returning success to the client on write operations.
…	…
FILE_NO_INTERMEDIATE_BUFFERING 0x00000008	The server or underlying object store SHOULD NOT cache data at intermediate layers and SHOULD allow it to flow through to persistent storage.
…	…

Changed to:

Value	Meaning
…	…
FILE_WRITE_THROUGH 0x00000002	The server performs file write-through; file data is written to the underlying storage before completing the write operation on this open.

Errata Published*	Description
	<table><tr><td>…</td><td>…</td></tr><tr><td>FILE_NO_INTERMEDIATE_BUFFERING 0x00000008</td><td>File buffering is not performed on this open; file data is retained in memory before writing or after reading it from the underlying storage.</td></tr><tr><td>…</td><td>…</td></tr></table> In Section 2.2.19, SMB2 READ Request, the description of SMB2_READFLAG_READ_UNBUFFERED has been changed from: The server or underlying object store SHOULD NOT cache the read data at intermediate layers. Changed to: The data is read directly from the underlying storage. In Section 2.2.21, SMB2 WRITE Request, FILE_WRITE_THROUGH and FILE_NO_INTERMEDIATE_BUFFERING have been changed from: The write data is written to persistent storage before the response is sent regardless of how the file was opened. This value is not valid for the SMB 2.0.2 dialect. The server or underlying object store SHOULD NOT cache the write data at intermediate layers and SHOULD allow it to flow through to persistent storage. This bit is not valid for the SMB 2.0.2, 2.1, and 3.0 dialects. Changed to: The server performs File write-through on the write operation. This value is not valid for the SMB 2.0.2 dialect. File buffering is not performed. This bit is not valid for the SMB 2.0.2, 2.1, and 3.0 dialects. In Section 3.3.5.11, Receiving an SMB2 FLUSH Request, "persistent storage" was changed to "underlying storage". In Section 3.3.5.13, Receiving an SMB2 WRITE Request, "persistent storage" was changed to "underlying storage".
2018/04/09	In Section 3.3.1.10, Per Open, changed from: ● Open.IsDurable: A Boolean that indicates whether the underlying object store supports durable operation for this Open. Changed to: ● Open.IsDurable: A Boolean that indicates whether the Open is preserved for reconnect.

Errata Published*	Description
	In Section 3.3.5.9.6, Handling the SMB2_CREATE_DURABLE_HANDLE_REQUEST Create Context, changed from: In the "Successful Open Initialization" phase, if the underlying object store does not grant durability, the server MUST ignore the SMB2_CREATE_DURABLE_HANDLE_REQUEST create context and skip the rest of the processing in this phase. Otherwise, the server MUST set Open.IsDurable to TRUE. This permits the client to use Open.DurableFileId to request a reopen of the file on a subsequent request as specified in section 3.3.5.9.7. The server MUST also set Open.DurableOwner to a security descriptor accessible only by the user represented by Open.Session.SecurityContext. Changed to: In the "Successful Open Initialization" phase, if the underlying object store does not grant durability, the server MUST skip the rest of the processing in this phase. Otherwise, the server MUST set Open.IsDurable to TRUE and Open.DurableOwner to a security descriptor accessible only by the user represented by Open.Session.SecurityContext. In Section 3.3.5.9.10, Handling the SMB2_CREATE_DURABLE_HANDLE_REQUEST_V2 Create Context, changed from: ● In the "Successful Open Initialization" phase , the server MUST set Open.IsDurable to TRUE. The server MUST also set Open.DurableOwner to a security descriptor accessible only by the user represented by Open.Session.SecurityContext. If the SMB2_DHANDLE_FLAG_PERSISTENT bit is set in the Flags field of the request, TreeConnect.Share.IsCA is TRUE, and Connection.ServerCapabilities includes SMB2_GLOBAL_CAP_PERSISTENT_HANDLES, the server MUST set Open.IsPersistent to TRUE. Changed to: ● In the "Successful Open Initialization" phase, if the underlying object store does not grant durability, the server MUST skip the rest of the processing in this section. Otherwise , the server MUST set Open.IsDurable to TRUE. The server MUST also set Open.DurableOwner to a security descriptor accessible only by the user represented by Open.Session.SecurityContext. If the SMB2_DHANDLE_FLAG_PERSISTENT bit is set in the Flags field of the request, TreeConnect.Share.IsCA is TRUE, and Connection.ServerCapabilities includes SMB2_GLOBAL_CAP_PERSISTENT_HANDLES, the server MUST set Open.IsPersistent to TRUE.

*Date format: YYYY/MM/DD

[MS-SMBD]: SMB2 Remote Direct Memory Access (RDMA) Transport Protocol