[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

Errata below are for Protocol Document Version V12.0 – 2018/09/12.

<table>
<tr><th>Errata Published *</th><th>Description</th></tr>
<tr><td>2020/02/17</td><td>
In Section 2.2.14, SMB2 CREATE Response, the following was removed:

<49> Section 2.2.14: Windows-based clients never use exclusive oplocks. Because there are no situations where it would require an exclusive oplock where it would not also require an SMB2_OPLOCK_LEVEL_BATCH, it always requests an SMB2_OPLOCK_LEVEL_BATCH.

In Section 2.2.24.1, Oplock Break Acknowledgment, the following was changed from:

OplockLevel (1 byte): The client will set this field to the lowered oplock level that the client accepts for this file. This field MUST contain one of the following values.<55>.

<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>SMB2_OPLOCK_LEVEL_NONE
0x00</td><td>The client has lowered its oplock level for this file to none.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_II
0x01</td><td>The client has lowered its oplock level for this file to level II.</td></tr>
</table>

<55> Section 2.2.24.1: Windows-based clients never use exclusive oplocks. There are no situations where an exclusive oplock would be used instead of using a SMB2_OPLOCK_LEVEL_BATCH.

Changed to:

OplockLevel (1 byte): The client will set this field to the lowered oplock level that the client accepts for this file. This field MUST contain one of the following values.

<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>SMB2_OPLOCK_LEVEL_NONE
0x00</td><td>The client has lowered its oplock level for this file to none.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_II
0x01</td><td>The client has lowered its oplock level for this file to level II.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_EXCLUSIVE
0x08</td><td>The client has lowered its oplock level for this file to level Exclusive.</td></tr>
</table>
</td></tr>
</table>

Errata Published *	Description
	In Section 2.2.25.1, Oplock Break Response, the following was added: SMB2_OPLOCK_LEVEL_EXCLUSIVE 0x08 The server has lowered oplock level for this file to level Exclusive. In Section 3.3.4.6, Object Store Indicates an Oplock Break, the following was changed from: The underlying object store on the local resource indicates the breaking of an opportunistic lock, specifying the LocalOpen and the new oplock level, a status code of the oplock break, and optionally expects the new oplock level in return. The new oplock level MUST be either SMB2_OPLOCK_LEVEL_NONE or SMB2_OPLOCK_LEVEL_II. The conditions under which each oplock level is to be indicated are described in [MS-FSA] section 2.1.5.17.3. Changed to: The underlying object store on the local resource indicates the breaking of an opportunistic lock, specifying the LocalOpen and the new oplock level, a status code of the oplock break, and optionally expects the new oplock level in return. The new oplock level SHOULD<200> be SMB2_OPLOCK_LEVEL_NONE or SMB2_OPLOCK_LEVEL_II or SMB2_OPLOCK_LEVEL_EXCLUSIVE. The conditions under which each oplock level is to be indicated are described in [MS-FSA] section 2.1.5.17.3. <200> Section 3.3.4.6: In Windows-based SMB2 servers, underlying object store never breaks opportunistic lock to SMB2_OPLOCK_LEVEL_EXCLUSIVE oplock level. In Section 3.3.5.22.1, Processing an Oplock Acknowledgment, the following was changed from: The server MUST locate the session, as specified in section 3.3.5.2.9. The server MUST locate the tree connection, as specified in section 3.3.5.2.11. Next, the server MUST locate the open on which the client is acknowledging an oplock break by performing a lookup in Session.OpenTable using FileId.Volatile of the request as the lookup key. If no open is found, or if Open.DurableFileId is not equal to FileId.Persistent, the server MUST fail the request with STATUS_FILE_CLOSED. Otherwise, the server MUST locate the Request in Connection.RequestList for which Request.MessageId matches the MessageId value in the SMB2 header, and set Request.Open to the Open. If the OplockLevel in the acknowledgment is SMB2_OPLOCK_LEVEL_LEASE, the server MUST do the following: If Open.OplockState is not Breaking, stop processing the acknowledgment, and send an error response with STATUS_INVALID_PARAMETER. ● If Open.OplockState is Breaking, complete the oplock break request received from the object store as described in section 3.3.4.6, with a new level SMB2_OPLOCK_LEVEL_NONE in an implementation-specific manner,<380> and set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE, and Open.OplockState to None.

Errata Published *	Description
	If Open.OplockLevel is SMB2_OPLOCK_LEVEL_EXCLUSIVE or SMB2_OPLOCK_LEVEL_BATCH, and if OplockLevel is not SMB2_OPLOCK_LEVEL_II or SMB2_OPLOCK_LEVEL_NONE, the server MUST do the following: ● If Open.OplockState is not Breaking, stop processing the acknowledgment, and send an error response with STATUS_INVALID_OPLOCK_PROTOCOL. If Open.OplockState is Breaking, complete the oplock break request received from the object store, as described in section 3.3.4.6, with a new level SMB2_OPLOCK_LEVEL_NONE in an implementation-specific manner,<381> and set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE and Open.OplockState to None. If Open.OplockLevel is SMB2_OPLOCK_LEVEL_II, and if OplockLevel is not SMB2_OPLOCK_LEVEL_NONE, the server MUST do the following: ● If Open.OplockState is not Breaking, stop processing the acknowledgment, and send an error response with STATUS_INVALID_OPLOCK_PROTOCOL. If Open.OplockState is Breaking, complete the oplock break request received from the object store, as described in section 3.3.4.6, with a new level SMB2_OPLOCK_LEVEL_NONE in an implementation-specific manner,<382> and set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE and Open.OplockState to None. If OplockLevel is SMB2_OPLOCK_LEVEL_II or SMB2_OPLOCK_LEVEL_NONE, the server MUST do the following: ● If Open.OplockState is not Breaking, stop processing the acknowledgment, and send an error response with STATUS_INVALID_DEVICE_STATE. If Open.OplockState is Breaking, complete the oplock break request received from the object store as described in section 3.3.4.6, with a new level received in OplockLevel in an implementation-specific manner.<384> If the object store indicates an error, set the Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE, the Open.OplockState to None, and send the error response with the error code received. If the object store indicates success, update Open.OplockLevel and Open.OplockState as follows: ● If OplockLevel is SMB2_OPLOCK_LEVEL_II, set Open.OplockLevel to SMB2_OPLOCK_LEVEL_II and Open.OplockState to Held. ● If OplockLevel is SMB2_OPLOCK_LEVEL_NONE, set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE and the Open.OplockState to None. The server then MUST construct an oplock break response using the syntax specified in section 2.2.25.1 with the following value: ● OplockLevel MUST be set to Open.OplockLevel. This response MUST then be sent to the client.

Errata Published *	Description
	The status code returned by this operation MUST be one of those defined in [MS-ERREF]. Common status codes returned by this operation include: • STATUS_ACCESS_DENIED • STATUS_FILE_CLOSED • STATUS_INVALID_OPLOCK_PROTOCOL • STATUS_INVALID_PARAMETER • STATUS_INVALID_DEVICE_STATE • STATUS_NETWORK_NAME_DELETED • STATUS_USER_SESSION_DELETED Changed to: The server MUST locate the session, as specified in section 3.3.5.2.9. The server MUST locate the tree connection, as specified in section 3.3.5.2.11. Next, the server MUST locate the open on which the client is acknowledging an oplock break by performing a lookup in Session.OpenTable using FileId.Volatile of the request as the lookup key. If no open is found, or if Open.DurableFileId is not equal to FileId.Persistent, the server MUST fail the request with STATUS_FILE_CLOSED. Otherwise, the server MUST locate the Request in Connection.RequestList for which Request.MessageId matches the MessageId value in the SMB2 header, and set Request.Open to the Open. If Open.OplockState is not Breaking, the server MUST stop processing the acknowledgment, and send an error response with STATUS_INVALID_DEVICE_STATE. If the OplockLevel in the acknowledgment is SMB2_OPLOCK_LEVEL_LEASE, the server MUST complete the oplock break request received from the object store as described in section 3.3.4.6, with a new level SMB2_OPLOCK_LEVEL_NONE in an implementation-specific manner,<381> and set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE, and Open.OplockState to None, send an error response with STATUS_INVALID_PARAMETER and stop processing. If any of the following conditions is TRUE, the server MUST complete the oplock break request received from the object store, as described in section 3.3.4.6, with a new level SMB2_OPLOCK_LEVEL_NONE in an implementation-specific manner<382>, set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE and Open.OplockState to None, send an error response with STATUS_INVALID_OPLOCK_PROTOCOL, and stop processing: • If Open.OplockLevel is SMB2_OPLOCK_LEVEL_EXCLUSIVE, and if OplockLevel is not SMB2_OPLOCK_LEVEL_II or SMB2_OPLOCK_LEVEL_NONE.

Errata Published *	Description
	• If Open.OplockLevel is SMB2_OPLOCK_LEVEL_BATCH and if OplockLevel is not SMB2_OPLOCK_LEVEL_II, or SMB2_OPLOCK_LEVEL_NONE, or SMB2_OPLOCK_LEVEL_EXCLUSIVE. • If Open.OplockLevel is SMB2_OPLOCK_LEVEL_II, and OplockLevel is not SMB2_OPLOCK_LEVEL_NONE. If OplockLevel is SMB2_OPLOCK_LEVEL_EXCLUSIVE, the server MUST complete the oplock break request received from the object store as described in section 3.3.4.6, with a new level SMB2_OPLOCK_LEVEL_NONE in an implementation-specific manner.<383> If OplockLevel is SMB2_OPLOCK_LEVEL_II or SMB2_OPLOCK_LEVEL_NONE, the server MUST complete the oplock break request received from the object store as described in section 3.3.4.6, with a new level received in OplockLevel in an implementation-specific manner.<384> If the object store indicates an error, the server MUST set the Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE, the Open.OplockState to None, send the error response with the error code received, and stop processing. If the object store indicates success, the server MUST update Open.OplockLevel and Open.OplockState as follows: • If OplockLevel is SMB2_OPLOCK_LEVEL_EXCLUSIVE, set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE and Open.OplockState to None. • If OplockLevel is SMB2_OPLOCK_LEVEL_II, set Open.OplockLevel to SMB2_OPLOCK_LEVEL_II and Open.OplockState to Held. • If OplockLevel is SMB2_OPLOCK_LEVEL_NONE, set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE and the Open.OplockState to None. The server then MUST construct an oplock break response using the syntax specified in section 2.2.25.1 with the following value: • OplockLevel MUST be set to Open.OplockLevel. This response MUST then be sent to the client. The status code returned by this operation MUST be one of those defined in [MS-ERREF]. Common status codes returned by this operation include: • STATUS_ACCESS_DENIED • STATUS_FILE_CLOSED • STATUS_INVALID_OPLOCK_PROTOCOL • STATUS_INVALID_PARAMETER • STATUS_INVALID_DEVICE_STATE

Errata Published *	Description
	• STATUS_NETWORK_NAME_DELETED • STATUS_USER_SESSION_DELETED <381> Section 3.3.5.22.1: Windows-based servers complete the oplock break indication request with the object store by providing the following SMB2 parameters as input parameters, as specified [MS-FSA] section 2.1.5.18: Object Store parameter SMB2 parameter Open Open.LocalOpen Type SMB2_OPLOCK_LEVEL_NONE <383> Section 3.3.5.22.1: Windows-based servers complete the oplock break indication request with the object store by providing the following SMB2 parameters as input parameters, as specified [MS-FSA] section 2.1.5.18: Object Store parameter SMB2 parameter Open Open.LocalOpen Type SMB2_OPLOCK_LEVEL_NONE
2020/02/03	In Section 3.3.5.9, Receiving an SMB2 CREATE Request, the following was added: • If Connection.Dialect belongs to the SMB 3.x dialect family and Open.LocalOpen is a reparse point, set the SMB2_CREATE_FLAG_REPARSEPOINT bit in the Flags field. In Section 3.3.5.9.4, Handling the SMB2_CREATE_TIMEWARP_TOKEN Create Context, the following was removed: • If Connection.Dialect belongs to the SMB 3.x dialect family, the server MUST set the SMB2_CREATE_FLAG_REPARSEPOINT bit in the Flags field in SMB2 CREATE response.
2020/02/03	In Section 3.2.4.3, Application Requests Opening a File, the following was removed: The RequestedOplockLevel field is set to the oplock level that is requested by the application. If the application does not provide a requested oplock level, the client MUST choose an implementation-specific oplock level.<116> In Section 3.2.4.3, Application Requests Opening a File, the following was added: The RequestedOplockLevel field is set as below: If CreateOptions includes FILE_DIRECTORY_FILE, If Connection.SupportsDirectoryLeasing is TRUE, the client SHOULD set RequestedOplockLevel field to SMB2_OPLOCK_LEVEL_LEASE. Otherwise, RequestedOplockLevel field is set to SMB2_OPLOCK_LEVEL_ NONE. Otherwise, If the filename is stream name as defined in [MS-FSCC] section 2.1.5.3, RequestedOplockLevel field is set to SMB2_OPLOCK_LEVEL_NONE.

<table>
<tr><th>Errata Published *</th><th>Description</th></tr>
<tr><td></td><td>Otherwise, if Connection.SupportsFileLeasing is TRUE, the client SHOULD set RequestedOplockLevel field to SMB2_OPLOCK_LEVEL_LEASE.
Otherwise, if the oplock level requested by the application is SMB2_OPLOCK_LEVEL_NONE or SMB2_OPLOCK_LEVEL_II or SMB2_OPLOCK_LEVEL_BATCH, the client MUST set RequestedOplockLevel field to the oplock level that is requested by the application.
Otherwise, RequestedOplockLevel field is set to an implementation-specific oplock level.<116>
<116> Section 3.2.4.3: Windows-based clients will request a batch oplock for file creates when application does not provide a requested oplock level, or an exclusive oplock is specified, or a lease is requested.</td></tr>
<tr><td>2020/01/20</td><td>In Section 2.2.33, SMB2 QUERY_DIRECTORY Request, clarified the meaning of the Flags field values. Removed product behavior note <64> from the SMB2_INDEX_SPECIFIED Flags field value that stated that Windows-based servers do not support resuming an enumeration at a specified FileIndex.

Changed from:
...
Flags (1 byte): Flags indicating how the query directory operation MUST be processed. This field MUST be a logical OR of the following values, or zero if none are selected:

<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>SMB2_RESTART_SCANS
0x01</td><td>The server MUST restart the enumeration from the beginning as specified in section 3.3.5.18.</td></tr><tr><td>SMB2_RETURN_SINGLE_ENTRY
0x02</td><td>The server MUST only return the first entry of the search results.</td></tr><tr><td>SMB2_INDEX_SPECIFIED
0x04</td><td>The server SHOULD<64> return entries beginning at the byte number specified by FileIndex.</td></tr><tr><td>SMB2_REOPEN
0x10</td><td>The server MUST restart the enumeration from the beginning, and the search pattern MUST be changed to the provided value. ....</td></tr></table>
...

Changed to:
...
Flags (1 byte): Flags indicating how the query directory operation MUST be processed. This field MUST be a logical OR of the following values, or zero if none are selected:

<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>SMB2_RESTART_SCANS
0x01</td><td>The server is requested to restart the enumeration from the beginning as specified in section 3.3.5.18.</td></tr><tr><td>SMB2_RETURN_SINGLE_ENTRY
0x02</td><td>The server is requested to only return the first entry of the search results.</td></tr><tr><td>SMB2_INDEX_SPECIFIED
0x04</td><td>The server is requested to return entries beginning at the byte number specified by FileIndex.</td></tr></table></td></tr>
</table>

Errata Published *	Description
	SMB2_REOPEN 0x10 — The server is requested to restart the enumeration from the beginning, and the search pattern is to be changed to the provided value. .... (Removed the following product behavior note) <64> Section 2.2.33: Windows-based servers do not support resuming an enumeration at a specified FileIndex. The server will ignore this flag. In Section 3.3.5.18, Receiving an SMB2 QUERY_DIRECTORY Request, clarified product behavior note <351> when Windows-based servers perform query directory requests by updating the FileIndex input parameter. Also updated product behavior note <352> by clarifying that Windows-based servers ignore SMB2_INDEX_SPECIFIED in the Flags field and in the FileIndex value. Changed from: ... The server MUST invoke the query directory procedure from the underlying object store in an implementation-specific manner<352>. An underlying object store MAY<353> choose to support resuming enumerations by index number, if SMB2_INDEX_SPECIFIED is set in the Flags field and an index number is specified in the FileIndex field of the SMB2 QUERY_DIRECTORY Request. ... <352> Section 3.3.5.18: Windows-based servers perform query directory requests, as specified in [MS-FSA] section 2.1.5.5 with the following input parameters: • Open is set to Open.LocalOpen. • FileInformationClass is set to the InformationClass that is received in the SMB2 QUERY_DIRECTORY Request. • OutputBufferSize is set to the OutputBufferLength that is received in the SMB2 QUERY_DIRECTORY Request. • If SMB2_RESTART_SCANS or SMB2_REOPEN is set in the Flags field of the SMB2 QUERY_DIRECTORY Request, RestartScan is set to TRUE. • If SMB2_RETURN_SINGLE_ENTRY is set in the Flags field of the request, ReturnSingleEntry is set to TRUE. • FileIndex is set to FileIndex received in the SMB2 QUERY_DIRECTORY Request. • FileNamePattern is set to the search pattern specified in the SMB2 QUERY_DIRECTORY by FileNameOffset and FileNameLength. <353> Section 3.3.5.18: Windows-based servers do not support resuming an enumeration at a specified FileIndex. The server will ignore this flag. Changed to: ... The server MUST invoke the query directory procedure from the underlying object store in an implementation-specific manner<351>.

<table>
<tr><th>Errata Published *</th><th>Description</th></tr>
<tr><td></td><td>The server MAY<352> choose to support resuming enumerations by index number, if SMB2_INDEX_SPECIFIED is set in the Flags field and an index number is specified in the FileIndex field of the SMB2 QUERY_DIRECTORY Request.
...

<351> Section 3.3.5.18: Windows-based servers perform query directory requests, as specified in [MS-FSA] section 2.1.5.5 with the following input parameters:
● Open is set to Open.LocalOpen.
● FileInformationClass is set to the InformationClass that is received in the SMB2 QUERY_DIRECTORY Request.
● OutputBufferSize is set to the OutputBufferLength that is received in the SMB2 QUERY_DIRECTORY Request.
● If SMB2_RESTART_SCANS or SMB2_REOPEN is set in the Flags field of the SMB2 QUERY_DIRECTORY Request, RestartScan is set to TRUE.
● If SMB2_RETURN_SINGLE_ENTRY is set in the Flags field of the request, ReturnSingleEntry is set to TRUE.
● FileIndex is set to 0.
● FileNamePattern is set to the search pattern specified in the SMB2 QUERY_DIRECTORY by FileNameOffset and FileNameLength.
<352> Section 3.3.5.18: Windows-based servers ignore SMB2_INDEX_SPECIFIED in Flags field and FileIndex value.</td></tr>
<tr><td>2020/01/06</td><td>In Section 2.2.33, SMB2 QUERY_DIRECTORY Request, clarified the meaning of the Flags field values. Removed product behavior note <64> from the SMB2_INDEX_SPECIFIED Flags field value that stated that Windows-based servers do not support resuming an enumeration at a specified FileIndex.

Changed from:
...
Flags (1 byte): Flags indicating how the query directory operation MUST be processed. This field MUST be a logical OR of the following values, or zero if none are selected:

<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>SMB2_RESTART_SCANS
0x01</td><td>The server MUST restart the enumeration from the beginning as specified in section 3.3.5.18.</td></tr>
<tr><td>SMB2_RETURN_SINGLE_ENTRY
0x02</td><td>The server MUST only return the first entry of the search results.</td></tr>
<tr><td>SMB2_INDEX_SPECIFIED
0x04</td><td>The server SHOULD<64> return entries beginning at the byte number specified by FileIndex.</td></tr>
<tr><td>SMB2_REOPEN
0x10</td><td>The server MUST restart the enumeration from the beginning, and the search pattern MUST be changed to the provided value. ....</td></tr>
</table>

...

Changed to:
...</td></tr>
</table>

<table>
<tr><th>Errata Published *</th><th>Description</th></tr>
<tr><td></td><td>
Flags (1 byte): Flags indicating how the query directory operation MUST be processed. This field MUST be a logical OR of the following values, or zero if none are selected:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>SMB2_RESTART_SCANS
0x01</td><td>The server is requested to restart the enumeration from the beginning as specified in section 3.3.5.18.</td></tr>
<tr><td>SMB2_RETURN_SINGLE_ENTRY
0x02</td><td>The server is requested to only return the first entry of the search results.</td></tr>
<tr><td>SMB2_INDEX_SPECIFIED
0x04</td><td>The server is requested to return entries beginning at the byte number specified by FileIndex.</td></tr>
<tr><td>SMB2_REOPEN
0x10</td><td>The server is requested to restart the enumeration from the beginning, and the search pattern is to be changed to the provided value.</td></tr>
</table>
....

(Removed the following product behavior note)

<64> Section 2.2.33: Windows-based servers do not support resuming an enumeration at a specified FileIndex. The server will ignore this flag.

In Section 3.3.5.18, Receiving an SMB2 QUERY_DIRECTORY Request, clarified product behavior note <351> when Windows-based servers perform query directory requests by updating the FileIndex input parameter. Also updated product behavior note <352> by clarifying that Windows-based servers ignore SMB2_INDEX_SPECIFIED in the Flags field and in the FileIndex value.

Changed from:
...
The server MUST invoke the query directory procedure from the underlying object store in an implementation-specific manner<352>.

An underlying object store MAY<353> choose to support resuming enumerations by index number, if SMB2_INDEX_SPECIFIED is set in the Flags field and an index number is specified in the FileIndex field of the SMB2 QUERY_DIRECTORY Request.
...

<352> Section 3.3.5.18: Windows-based servers perform query directory requests, as specified in [MS-FSA] section 2.1.5.5 with the following input parameters:
● Open is set to Open.LocalOpen.
● FileInformationClass is set to the InformationClass that is received in the SMB2 QUERY_DIRECTORY Request.
● OutputBufferSize is set to the OutputBufferLength that is received in the SMB2 QUERY_DIRECTORY Request.
</td></tr>
</table>

Errata Published *	Description
	• If SMB2_RESTART_SCANS or SMB2_REOPEN is set in the Flags field of the SMB2 QUERY_DIRECTORY Request, RestartScan is set to TRUE. • If SMB2_RETURN_SINGLE_ENTRY is set in the Flags field of the request, ReturnSingleEntry is set to TRUE. • FileIndex is set to FileIndex received in the SMB2 QUERY_DIRECTORY Request. • FileNamePattern is set to the search pattern specified in the SMB2 QUERY_DIRECTORY by FileNameOffset and FileNameLength. <353> Section 3.3.5.18: Windows-based servers do not support resuming an enumeration at a specified FileIndex. The server will ignore this flag. Changed to: ... The server MUST invoke the query directory procedure from the underlying object store in an implementation-specific manner<351>. The server MAY<352> choose to support resuming enumerations by index number, if SMB2_INDEX_SPECIFIED is set in the Flags field and an index number is specified in the FileIndex field of the SMB2 QUERY_DIRECTORY Request. ... <351> Section 3.3.5.18: Windows-based servers perform query directory requests, as specified in [MS-FSA] section 2.1.5.5 with the following input parameters: • Open is set to Open.LocalOpen. • FileInformationClass is set to the InformationClass that is received in the SMB2 QUERY_DIRECTORY Request. • OutputBufferSize is set to the OutputBufferLength that is received in the SMB2 QUERY_DIRECTORY Request. • If SMB2_RESTART_SCANS or SMB2_REOPEN is set in the Flags field of the SMB2 QUERY_DIRECTORY Request, RestartScan is set to TRUE. • If SMB2_RETURN_SINGLE_ENTRY is set in the Flags field of the request, ReturnSingleEntry is set to TRUE. • FileIndex is set to 0. • FileNamePattern is set to the search pattern specified in the SMB2 QUERY_DIRECTORY by FileNameOffset and FileNameLength. <352> Section 3.3.5.18: Windows-based servers ignore SMB2_INDEX_SPECIFIED in Flags field and FileIndex value.
2019/12/16	In Section 3.2.5.1.3, Verifying the Signature, clarified when verification is not required and described under what circumstances the client retrieves SessionId. Also, removed product behavior note <149> that described when Windows-based clients will not disconnect the connection but simply disregard the incorrectly signed response. Changed from: If the client implements the SMB 3.x dialect family and if the decryption in section 3.2.5.1.1.1 succeeds, the client MUST skip the processing in this section. If the MessageId is 0xFFFFFFFFFFFFFFFF, no verification is necessary. If the SMB2 header of the response has SMB2_FLAGS_SIGNED set in the Flags field and the message is not encrypted, the client MUST verify the signature as follows: The client MUST look up the session in the Connection.SessionTable using the SessionId in the SMB2 header of the response. If the session is not found, the response MUST be discarded as invalid. If Connection.Dialect belongs to the SMB 3.x dialect family, and the received message is an SMB2 SESSION_SETUP Response without a status code equal to STATUS_SUCCESS in the

Errata Published *	Description
	header, the client MUST verify the signature of the message as specified in section 3.1.5.1, using Session.SigningKey as the signing key, and passing the response message. For all other messages, the client MUST look up the Channel in Session.ChannelList, where the Channel.Connection matches the connection on which this message is received, and MUST use Channel.SigningKey for verifying the signature as specified in section 3.1.5.1. Otherwise, the client MUST verify the signature of the message as specified in section 3.1.5.1, using Session.SessionKey as the signing key, and passing the response message. If signature verification fails, the client MUST discard the received message and do no further processing for it. The client MAY also choose to disconnect the connection. If signature verification succeeds, the client MUST continue processing the packet, as specified in subsequent sections. If the SMB2 header of the response does not have SMB2_FLAGS_SIGNED set in the Flags field, the client MUST determine if the server failed to sign a packet that required signing. If the message is an interim response or an SMB2 OPLOCK_BREAK notification, signing validation MUST NOT occur. Otherwise, the client MUST look up the session in the Connection.SessionTable using the SessionId in the SMB2 header of the response. If the session is found, the Session.SigningRequired is equal to TRUE, the message is not an interim response, and the message is not an SMB2 OPLOCK_BREAK notification, the client MUST discard the received message and do no further processing for it. The client MAY also choose to disconnect the connection. If there is no SessionId, if the session is not found, or if Session.SigningRequired is FALSE, the client continues processing on the packet, as specified in subsequent sections.<149> Changed to: The client MUST skip the processing in this section if any of the following is TRUE: ● Client implements the SMB 3.x dialect family and decryption in section 3.2.5.1.1.1 succeeds ● MessageId is 0xFFFFFFFFFFFFFFFF ● Status in the SMB2 header is STATUS_PENDING For SMB2 SESSION_SETUP, the client MUST retrieve SessionId from SMB2 header of the response. For all other messages, the client MUST retrieve SessionId from the corresponding Request.Message. The client MUST look up the session in the Connection.SessionTable using the SessionId. If the session is not found, the response MUST be discarded as invalid. Otherwise if Session.SigningRequired is TRUE, the client MUST perform the following: ● If Connection.Dialect belongs to the SMB 3.x dialect family, and the received message is an SMB2 SESSION_SETUP Response without a status code equal to STATUS_SUCCESS in the header, the client MUST verify the signature of the message as specified in section 3.1.5.1, using Session.SigningKey as the signing key, and passing the response message. For all other messages, the client MUST look up the Channel in Session.ChannelList, where the Channel.Connection matches the connection on which this message is received, and MUST use Channel.SigningKey for verifying the signature as specified in section 3.1.5.1. ● Otherwise, the client MUST verify the signature of the message as specified in section 3.1.5.1, using Session.SessionKey as the signing key, and passing the response message. If signature verification fails, the client MUST discard the received message. The client MAY also choose to disconnect the connection. (removed the following product behavior note) <149> Section 3.2.5.1.3: Windows-based clients will not disconnect the connection but simply disregard the incorrectly signed response.
2019/11/25	In Section 3.3.5.14, Receiving an SMB2 LOCK Request, addressed when the server verifies whether the lock/unlock request along with the LockSequence value has been successfully

Errata Published *	Description
	processed. Clarified when lock sequence verification is neither resilient nor persistent in product behavior note <316>. Changed from: ... If the LockSequence value in the SMB2 LOCK Request (section 2.2.26) is not zero, and either one of the following conditions is TRUE, the server SHOULD verify whether the lock/unlock request with that LockSequence value has been successfully processed before: ● Connection.Dialect is "2.1" and Open.IsResilient is TRUE. ● Connection.Dialect belongs to the SMB 3.x dialect family.<316> The server verifies the LockSequence by performing the following steps: ● The server MUST use LockSequenceIndex as an index into Open.LockSequenceArray in order to locate the sequence number entry. If the index exceeds the maximum extent of the Open.LockSequenceArray, or LockSequenceIndex is 0, or if the Open.LockSequenceArray.Valid is FALSE, the server MUST skip step 2 and continue lock/unlock processing. ● The server MUST compare LockSequenceNumber to the SequenceNumber of the entry located in step 1. If the sequence numbers are equal, the server MUST complete the lock/unlock request with success. Otherwise, the server MUST reset the entry by setting Valid to FALSE and continue lock/unlock processing.... <316> Section 3.3.5.14: Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2 do not verify the LockSequence value in the SMB2 LOCK Request (section 2.2.26) when both Open.IsResilient and Open.IsPersistent are FALSE. Changed to: ... If Connection.Dialect is not "2.0.2" and LockSequence value in the SMB2 LOCK Request (section 2.2.26) is not zero, the server SHOULD<316> verify whether the lock/unlock request with that LockSequence value has been successfully processed by performing the following steps: ● The server MUST use LockSequenceIndex as an index into Open.LockSequenceArray in order to locate the sequence number entry. If the index exceeds the maximum extent of the Open.LockSequenceArray, or LockSequenceIndex is 0, or if the Open.LockSequenceArray.Valid is FALSE, the server MUST skip step 2 and continue lock/unlock processing. ● The server MUST compare LockSequenceNumber to the SequenceNumber of the entry located in step 1. If the sequence numbers are equal, the server MUST complete the lock/unlock request with success. Otherwise, the server MUST reset the entry by setting Valid to FALSE and continue lock/unlock processing.... <316> Section 3.3.5.14: Windows 7 operating system and Windows Server 2008 R2 operating system do not perform lock sequence verification when Open.IsResilient is FALSE. Windows 8 operating system through Windows 10 v1909 and Windows Server 2012 operating system through Windows Server v1909 do not perform lock sequence verification when both Open.IsResilient and Open.IsPersistent are FALSE.
2019/11/25	In Section 2.2.14, SMB2 CREATE Response, changed the value OPLOCK_LEVEL_LEASE to SMB2_OPLOCK_LEVEL_LEASE in the OplockLevel table.

<table>
<tr><th>Errata Published *</th><th>Description</th></tr>
<tr><td></td><td>
Changed from:
...
OplockLevel (1 byte): The oplock level that is granted to the client for this open. This field MUST contain one of the following values.<49>
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>SMB2_OPLOCK_LEVEL_NONE
0x00</td><td>No oplock was granted.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_II
0x01</td><td>A level II oplock was granted.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_EXCLUSIVE
0x08</td><td>An exclusive oplock was granted.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_BATCH
0x09</td><td>A batch oplock was granted.</td></tr>
<tr><td>OPLOCK_LEVEL_LEASE
0xFF</td><td>A lease is requested. If set, the response packet MUST contain an SMB2_CREATE_RESPONSE_LEASE create context.</td></tr>
</table>
...

Changed to:
...
OplockLevel (1 byte): The oplock level that is granted to the client for this open. This field MUST contain one of the following values.<49>
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>SMB2_OPLOCK_LEVEL_NONE
0x00</td><td>No oplock was granted.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_II
0x01</td><td>A level II oplock was granted.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_EXCLUSIVE
0x08</td><td>An exclusive oplock was granted.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_BATCH
0x09</td><td>A batch oplock was granted.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_LEASE
0xFF</td><td>A lease is requested. If set, the response packet MUST contain an SMB2_CREATE_RESPONSE_LEASE create context.</td></tr>
</table>
...
</td></tr>
</table>

<table>
<tr><th>Errata Published *</th><th>Description</th></tr>
<tr><td></td><td>
In Section 2.2.23.1, Oplock Break Notification, added the value SMB2_OPLOCK_LEVEL_EXCLUSIVE to the OplockLevel table and removed product behavior note <54> from the OplockLevel field description.

Changed from:
OplockLevel (1 byte): The server sets this to the maximum value of the OplockLevel that the server will accept for an acknowledgment from the client. This field MUST contain one of the following values.<54>

<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>SMB2_OPLOCK_LEVEL_NONE
0x00</td><td>No oplock is available.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_II
0x01</td><td>A level II oplock is available.</td></tr>
</table>

Changed to:
...
OplockLevel (1 byte): The server sets this to the maximum value of the OplockLevel that the server will accept for an acknowledgment from the client. This field MUST contain one of the following values.

<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>SMB2_OPLOCK_LEVEL_NONE
0x00</td><td>No oplock is available.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_II
0x01</td><td>A level II oplock is available.</td></tr>
<tr><td>SMB2_OPLOCK_LEVEL_EXCLUSIVE
0x08</td><td>Exclusive oplock is available.</td></tr>
</table>

...
(removed the following product behavior note)
<54> Section 2.2.23.1: Windows-based clients never use exclusive oplocks. Because there are no situations where it would require an exclusive oplock where it would not also require an SMB2_OPLOCK_LEVEL_BATCH, it always requests an SMB2_OPLOCK_LEVEL_BATCH.

In Section 3.2.4.3, Application Requests Opening a File, clarified when clients will request a batch oplock for file creates in product behavior note <117>.

Changed from:
….

The SMB2 CREATE Request MUST be initialized as follows:
</td></tr>
</table>

Errata Published *	Description
	• The SecurityFlags field is set to 0. • The RequestedOplockLevel field is set to the oplock level that is requested by the application. If the application does not provide a requested oplock level, the client MUST choose an implementation-specific oplock level.<117> ...<117> Section 3.2.4.3: Windows-based clients will request a batch oplock for file creates. Changed to: …. The SMB2 CREATE Request MUST be initialized as follows: • The SecurityFlags field is set to 0. • The RequestedOplockLevel field is set to the oplock level that is requested by the application. If the application does not provide a requested oplock level, the client MUST choose an implementation-specific oplock level.<117> ... <117> Section 3.2.4.3: Windows-based clients will request a batch oplock for file creates, when no oplock level is specified or an exclusive oplock is specified. In Section 3.2.5.19.1, Receiving an Oplock Break Notification, clarified client actions based on the the Open.OplockLevel and the new OplockLevel that is received in the Oplock Break Notification. Removed product behavior note <164>. Changed from: The client MUST locate the open in the Session.OpenTable using the FileId in the Oplock Break Notification following the SMB2 header. If the open is not found, the oplock break indication MUST be discarded, and no further processing is required. If the open is found, the client MUST take action based on the Open.OplockLevel and the new OplockLevel that is received in the Oplock Break Notification. If the Open.OplockLevel is SMB2_OPLOCK_LEVEL_NONE, no action is required, and no further processing is required. If the Open.OplockLevel is SMB2_OPLOCK_LEVEL_II, and the OplockLevel is SMB2_OPLOCK_LEVEL_NONE, the client MUST set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE and an Oplock Break Acknowledgment MUST NOT be sent. If the Open.OplockLevel is SMB2_OPLOCK_LEVEL_BATCH, and the OplockLevel is SMB2_OPLOCK_LEVEL_NONE, the client MUST set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE. The client MUST flush any writes or byte range locks that it has cached locally to the server. When that is complete, the client MUST send an oplock break acknowledgment, as specified in the following sections. If the Open.OplockLevel is SMB2_OPLOCK_LEVEL_BATCH, and the OplockLevel is SMB2_OPLOCK_LEVEL_II, the client MUST set Open.OplockLevel to SMB2_OPLOCK_LEVEL_II. The client MUST flush any writes or byte range locks that it has cached locally to the server. When that is complete, the client MUST send an oplock break acknowledgment, specified as follows. The client MAY<164> choose to request and support SMB2_OPLOCK_LEVEL_EXCLUSIVE. If it does, the break operation would match those specified above for SMB2_OPLOCK_LEVEL_BATCH. It MUST NOT break from batch to exclusive. If the client is required to send an oplock break acknowledgment, it MUST construct a request following the syntax that is specified in section 2.2.24.1. The SMB2 header is initialized as follows: • Command MUST be set to SMB2 OPLOCK_BREAK. • The MessageId field is set as specified in section 3.2.4.1.3.

Errata Published *	Description
	• The client MUST set SessionId to Open.TreeConnect.Session.SessionId. • The client MUST set TreeId to Open.TreeConnect.TreeConnectId. The Oplock Break Acknowledgment request is initialized as follows: • The FileId MUST be set to Open.FileId. • The OplockLevel MUST be set to Open.OplockLevel. The request MUST be sent to the server. Changed to: The client MUST locate the open in the Session.OpenTable using the FileId in the Oplock Break Notification following the SMB2 header. If the open is not found, the client MUST stop processing. If the open is found, the client MUST take action based on the Open.OplockLevel and the new OplockLevel that is received in the Oplock Break Notification. • If Open.OplockLevel is SMB2_OPLOCK_LEVEL_II, and the new OplockLevel is SMB2_OPLOCK_LEVEL_NONE, the client MUST set Open.OplockLevel to SMB2_OPLOCK_LEVEL_NONE and MUST stop processing. • If Open.OplockLevel is SMB2_OPLOCK_LEVEL_EXCLUSIVE and the new OplockLevel is SMB2_OPLOCK_LEVEL_NONE or SMB2_OPLOCK_LEVEL_II, locate the File in GlobalFileTable using Open.FileName. The client MUST flush any writes or byte range locks that it has cached locally to the server. The client MUST set Open.OplockLevel to new OplockLevel and send an oplock break acknowledgment. • If Open.OplockLevel is SMB2_OPLOCK_LEVEL_BATCH and the new OplockLevel is SMB2_OPLOCK_LEVEL_EXCLUSIVE, locate the File in GlobalFileTable using Open.FileName. The client MUST process as below: • Close any cached handles that have already been closed by the application, as specified in section 3.2.4.5. • If File.OpenTable is empty, stop processing. • Otherwise, set Open.OplockLevel to new OplockLevel and send an oplock break acknowledgment. • If Open.OplockLevel is SMB2_OPLOCK_LEVEL_BATCH, and the new OplockLevel is SMB2_OPLOCK_LEVEL_NONE or SMB2_OPLOCK_LEVEL_II, locate the File in GlobalFileTable using Open.FileName. The client MUST process as below: • For all cached handles in File.OpenTable, • Flush any writes or byte range locks that it has cached locally to the server. • Close any cached handles that have already been closed by the application, as specified in section 3.2.4.5. • If File.OpenTable is empty, stop processing. • Otherwise, set Open.OplockLevel to new OplockLevel and send an oplock break acknowledgment. • Otherwise, the client MUST stop processing. The client MUST construct Oplock Break Acknowledgment following the syntax that is specified in section 2.2.24.1. The SMB2 header is initialized as follows: • Command MUST be set to SMB2 OPLOCK_BREAK. • The MessageId field is set as specified in section 3.2.4.1.3. • The client MUST set SessionId to Open.TreeConnect.Session.SessionId. • The client MUST set TreeId to Open.TreeConnect.TreeConnectId. The Oplock Break Acknowledgment is initialized as follows: • The FileId MUST be set to Open.FileId. • The OplockLevel MUST be set to Open.OplockLevel.

Errata Published *	Description
	The Oplock Break Acknowledgment MUST be sent to the server. (removed the following product behavior note) <164> Section 3.2.5.19.1: Windows-based clients will not request exclusive oplocks. In Section 3.3.1.10, Per Open, changed OPLOCK_LEVEL_LEASE to SMB2_OPLOCK_LEVEL_LEASE in the Open.OplockLevel description. Changed from: ... ● Open.OplockLevel: The current oplock level for this open. This value MUST be one of the OplockLevel values defined in section 2.2.14: SMB2_OPLOCK_LEVEL_NONE, SMB2_OPLOCK_LEVEL_II, SMB2_OPLOCK_LEVEL_EXCLUSIVE, SMB2_OPLOCK_LEVEL_BATCH, or OPLOCK_LEVEL_LEASE. ... Changed to: ... ● Open.OplockLevel: The current oplock level for this open. This value MUST be one of the OplockLevel values defined in section 2.2.14: SMB2_OPLOCK_LEVEL_NONE, SMB2_OPLOCK_LEVEL_II, SMB2_OPLOCK_LEVEL_EXCLUSIVE, SMB2_OPLOCK_LEVEL_BATCH, or SMB2_OPLOCK_LEVEL_LEASE. ...
2019/11/11	In Section 2.2.32.5.1.2, SOCKADDR_IN6, added a product behavior note to clarify Windows behavior when the ScopeId field should be set to zero. Changed from: ... ScopeId (4 bytes): The server SHOULD set this field to zero, and the client MUST ignore it on receipt. Changed to: ... ScopeId (4 bytes): The server SHOULD<64> set this field to zero, and the client MUST ignore it on receipt. <64> Section 2.2.32.5.1.2: Windows 10 v1709 operating system through Windows 10 v1909 operating system and Windows Server v1709 operating system through Windows Server v1909 operating system set this field to any value.
2019/11/11	In Section 2.2.2, SMB2 ERROR Response, clarified when the ErrorData variable field should match the ErrorData format. Changed from: ... ErrorData (variable): A variable-length data field that contains extended error information. If the ErrorContextCount field in the response is nonzero, this field MUST be formatted as a variable-length array of SMB2 ERROR Context structures as specified in section 2.2.2.1. Each

Errata Published *	Description
	SMB2 ERROR Context MUST start at an 8-byte aligned boundary relative to the start of the SMB2 ERROR Response. Otherwise, it MUST be formatted as specified in section 2.2.2.2. If the ByteCount field is zero then the server MUST supply an ErrorData field that is one byte in length, and SHOULD set that byte to zero; the client MUST ignore it on receipt.<4> Changed to: ... ErrorData (variable): A variable-length data field that contains extended error information. If the ErrorContextCount field in the response is nonzero, this field MUST be formatted as a variable-length array of SMB2 ERROR Context structures as specified in section 2.2.2.1. Each SMB2 ERROR Context MUST start at an 8-byte aligned boundary relative to the start of the SMB2 ERROR Response. Otherwise, it SHOULD<4> be formatted as specified in section 2.2.2.2. In Section 6, Appendix A: Product Behavior, clarified the applicable product versions that set ErrorData to one uninitialized byte when ByteCount is zero in product behavior note <4>. Changed from: <4> Section 2.2.2: Windows–based SMB2 servers leave this one byte of ErrorData uninitialized and it can contain any value. Changed to: <4> Section 2.2.2: Windows 10 v1703 operating system and prior and Windows Server 2016 and prior set ErrorData to one uninitialized byte when ByteCount is zero.
2019/11/11	In Section 3.2.1.2, Per SMB2 Transport Connection, added the Connection.OfferedDialects behavior when the client implements the SMB 3.x dialect family. Changed from: ... If the client implements the SMB 3.x dialect family, it MUST also implement the following: ... ● Connection.Server: A reference to the server entry to which the connection is established. ... Changed to: ... If the client implements the SMB 3.x dialect family, it MUST also implement the following: ... ● Connection.Server: A reference to the server entry to which the connection is established. ● Connection.OfferedDialects: An array of dialects sent in the SMB2 NEGOTIATE Request on this connection. ... In Section 3.2.4.2.2.2, SMB2-Only Negotiate, described when the client should set DialectCount to 1 and set Dialects array to Server.DialectRevision. Also added product behavior note <106> to describe the Windows product versions that set Dialects array to all the dialects the client implements and DialectCount to the number of dialects in Dialects array.

Errata Published *	Description
	Changed from: ... 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. Otherwise, • Set DialectCount to 0. ... • If RequireMessageSigning is TRUE, the client MUST set the SMB2_NEGOTIATE_SIGNING_REQUIRED bit to TRUE in SecurityMode. If RequireMessageSigning is FALSE, the client MUST set the SMB2_NEGOTIATE_SIGNING_ENABLED bit to TRUE in SecurityMode. The client MUST store the value of the SecurityMode field in Connection.ClientSecurityMode. ... 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. Otherwise, if the client implements the SMB 3.x dialect family and an alternate connection is being established to an already connected Server, the client SHOULD<106> set DialectCount to 1 and set Dialects array to Server.DialectRevision. Otherwise, • Set DialectCount to 0.... • If the client implements SMB 3.x dialect family, Connection.OfferedDialects MUST be set to the values in Dialects array. • If RequireMessageSigning is TRUE, the client MUST set the SMB2_NEGOTIATE_SIGNING_REQUIRED bit to TRUE in SecurityMode. If RequireMessageSigning is FALSE, the client MUST set the SMB2_NEGOTIATE_SIGNING_ENABLED bit to TRUE in SecurityMode. The client MUST store the value of the SecurityMode field in Connection.ClientSecurityMode. ... <106> Section 3.2.4.2.2.2: Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2 set Dialects array to all the dialects the client implements and DialectCount to the number of dialects in Dialects array. In Section 3.2.5.5, Receiving an SMB2 TREE_CONNECT Response, revised the description of the VALIDATE_NEGOTIATE_INFO request structure. Added product behavior note<156> to describe Windows behavior when the client sets Dialects array to Connection.OfferedDialects. Changed from: ...

Errata Published *	Description
	• The SMB2 IOCTL Request MUST be initialized as specified in section 2.2.31, with the exception of the following values: ... • The VALIDATE_NEGOTIATE_INFO request structure is constructed as follows and copied into the request at InputOffset bytes from the beginning of the SMB2 header: • Capabilities is set to Connection.ClientCapabilities. • Guid is set to the Connection.ClientGuid value. • SecurityMode is set to Connection.ClientSecurityMode. • Set DialectCount to 0. • If the client implements the SMB 2.0.2 dialect, it MUST do the following: • Increment the DialectCount by 1. • Set the value in Dialects[DialectCount-1] array to 0x0202. • If the client implements the SMB 2.1 dialect, it MUST do the following: • Increment the DialectCount by 1. • Set the value in Dialects[DialectCount-1] array to 0x0210. • If the client implements the SMB 3.0 dialect, it MUST do the following: • Increment the DialectCount by 1. • Set the value in the Dialects[DialectCount-1] array to 0x0300. • If the client implements the SMB 3.0.2 dialect, it MUST do the following: • Increment the DialectCount by 1. • Set the value in the Dialects[DialectCount-1] array to 0x0302. • The OutputOffset field offset to the Buffer[], in bytes, from the beginning of the SMB2 header. Changed to: ... • The SMB2 IOCTL Request MUST be initialized as specified in section 2.2.31, with the exception of the following values: ... • The VALIDATE_NEGOTIATE_INFO request structure is constructed as follows and copied into the request at InputOffset bytes from the beginning of the SMB2 header: • Capabilities is set to Connection.ClientCapabilities. • Guid is set to the Connection.ClientGuid value. • SecurityMode is set to Connection.ClientSecurityMode. • Dialects array SHOULD<156> be set to Connection.OfferedDialects. • Set DialectCount to the number of dialects in Dialects array. • The OutputOffset field offset to the Buffer[], in bytes, from the beginning of the SMB2 header….. <156> Section 3.2.5.5: Windows 10 v1507 operating system through Windows 10 v1909, Windows Server 2016, Windows Server v1709 through Windows Server v1909, and Windows Server 2019 set Dialects array to all the dialects the client implements.
2019/11/11	In Section 2.2.21, SMB2 WRITE Request, described the SMB2_CHANNEL_RDMA_V1 and SMB2_CHANNEL_RDMA_V1_INVALIDATE values as it relates to the SMB 3.x dialect family and the Channel field of the request in the RemainingBytes, WriteChannelInfoOffset, and WriteChannelInfoLength field descriptions. Also added RemainingBytes to the value descriptions in the Channel table.

<table>
<tr><th>Errata Published *</th><th>Description</th></tr>
<tr><td></td><td>
Changed from:
...
Channel (4 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, this field MUST contain exactly one of the following values:

<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>SMB2_CHANNEL_NONE
0x00000000</td><td>No channel information is present in the request. The WriteChannelInfoOffset and WriteChannelInfoLength fields MUST be set to zero by the client and MUST be ignored by the server.</td></tr>
</table>
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>SMB2_CHANNEL_RDMA_V1
0x00000001</td><td>One or more SMB_DIRECT_BUFFER_DESCRIPTOR_V1 structures as specified in [MS-SMBD] section 2.2.3.1 are present in the channel information specified by WriteChannelInfoOffset and WriteChannelInfoLength fields.</td></tr>
<tr><td>SMB2_CHANNEL_RDMA_V1_INVALIDATE
0x00000002</td><td>This flag is not valid for the SMB 3.0 dialect. One or more SMB_DIRECT_BUFFER_DESCRIPTOR_V1 structures as specified in [MS-SMBD] section 2.2.3.1 are present in the channel information specified by the WriteChannelInfoOffset and WriteChannelInfoLength fields. The server is requested to perform remote invalidation when responding to the request as specified in [MS-SMBD] section 3.1.4.2.</td></tr>
</table>

RemainingBytes (4 bytes): The number of subsequent bytes the client intends to write to the file after this operation completes. This value is provided to facilitate write caching and is not binding on the server.

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.
...

Changed to:
...
Channel (4 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, this field MUST contain exactly one of the following values:
</td></tr>
</table>

Errata Published *	Description

Value	Meaning
SMB2_CHANNEL_NONE 0x00000000	No channel information is present in the request. The **RemainingBytes**, **WriteChannelInfoOffset** and **WriteChannelInfoLength** fields MUST be set to zero by the client and MUST be ignored by the server.

Value	Meaning
SMB2_CHANNEL_RDMA_V1 0x00000001	One or more SMB_DIRECT_BUFFER_DESCRIPTOR_V1 structures as specified in [MS-SMBD] section 2.2.3.1 are present in the channel information specified by **RemainingBytes**, **WriteChannelInfoOffset** and **WriteChannelInfoLength** fields.
SMB2_CHANNEL_RDMA_V1_INVALIDATE 0x00000002	This flag is not valid for the SMB 3.0 dialect. One or more SMB_DIRECT_BUFFER_DESCRIPTOR_V1 structures as specified in [MS-SMBD] section 2.2.3.1 are present in the channel information specified by the **RemainingBytes**, **WriteChannelInfoOffset** and **WriteChannelInfoLength** fields. The server is requested to perform remote invalidation when responding to the request as specified in [MS-SMBD] section 3.1.4.2.

RemainingBytes (4 bytes): For the SMB 3.x dialect family and the Channel field of the request contains SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE, this field contains the length, in bytes, of the data being written.

WriteChannelInfoOffset (2 bytes): For the SMB 3.x dialect family and the Channel field of the request contains SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE, 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 3.x dialect family and the Channel field of the request contains SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE, it contains the length, in bytes, of the channel data as specified by the Channel field of the request.

...

In Section 3.3.5.13, Receiving an SMB2 WRITE Request, clarified what the server must do if the Connection.Dialect belongs to the SMB 3.x dialect family and the Channel field contains the value SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE.

Changed from:

...

The server SHOULD<309> ignore undefined bits in the Flags field.

If the server implements the SMB 3.0.2 or SMB 3.1.1 dialect, Connection.Dialect is not "3.0.2" or "3.1.1", and the SMB2_WRITEFLAG_WRITE_UNBUFFERED bit is set in the Flags field, the server MUST ignore the bit.

If the request Channel field contains the value SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE, then the data MUST be first obtained via the processing specified in [MS-SMBD] section 3.1.4.6 RDMA Read from Peer Buffer, providing the Connection, a newly allocated buffer to receive the data, and the array of

Errata Published *	Description
	SMB_DIRECT_BUFFER_DESCRIPTOR_V1 structures passed in the request at offset WriteChannelInfoOffset and of length WriteChannelInfoLength fields. ... Changed to: ... The server SHOULD<309> ignore undefined bits in the Flags field. If the server implements the SMB 3.0.2 or SMB 3.1.1 dialect, Connection.Dialect is not "3.0.2" or "3.1.1", and the SMB2_WRITEFLAG_WRITE_UNBUFFERED bit is set in the Flags field, the server MUST ignore the bit. If Connection.Dialect belongs to the SMB 3.x dialect family and the Channel field contains the value SMB2_CHANNEL_RDMA_V1 or SMB2_CHANNEL_RDMA_V1_INVALIDATE, the server MUST do the following: ● The server MUST return STATUS_INVALID_PARAMETER to the client in the following conditions: ● RemainingBytes field is greater than Connection.MaxWriteSize. ● Length field of the first SMB_DIRECT_BUFFER_DESCRIPTOR_V1 structure is zero. ● Sum of the values of Length fields in all SMB_DIRECT_BUFFER_DESCRIPTOR_V1 structures is less than RemainingBytes. ● The data MUST be first obtained via the processing specified in [MS-SMBD] section 3.1.4.6, providing the Connection, a newly allocated buffer to receive the data, and the array of SMB_DIRECT_BUFFER_DESCRIPTOR_V1 structures passed in the request at offset WriteChannelInfoOffset and of length WriteChannelInfoLength fields. ...

*Date format: YYYY/MM/DD