(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4938418号
(P4938418)

(45)発行日　平成24年5月23日(2012.5.23)　(24)登録日　平成24年3月2日(2012.3.2)

(51)Int.Cl.　F I
H04L 29/06 (2006.01)　H04L 13/00 305C
G06F 13/00 (2006.01)　G06F 13/00 351F

請求項の数 5　(全 37 頁)

(21)出願番号　特願2006-307121 (P2006-307121)
(22)出願日　平成18年11月13日 (2006.11.13)
(62)分割の表示　特願2005-356145 (P2005-356145)の分割
原出願日　平成17年12月9日 (2005.12.9)
(65)公開番号　特開2007-49755 (P2007-49755A)
(43)公開日　平成19年2月22日 (2007.2.22)
審査請求日　平成20年11月21日 (2008.11.21)
(31)優先権主張番号　60/685,008
(32)優先日　平成17年5月25日 (2005.5.25)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　11/182,251
(32)優先日　平成17年7月15日 (2005.7.15)
(33)優先権主張国　米国 (US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　アハメド　モハメド
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

最終頁に続く

(54)【発明の名称】データ通信プロトコル

(57)【特許請求の範囲】
【請求項１】
コンピュータ実行可能命令を格納するコンピュータ可読記憶媒体であって、
前記コンピュータ実行可能命令は、実行されると、ネットワークファイルサーバに、
クライアントから複数のファイルシステムコマンドを含む第１の複合要求を受け取るステップであって、前記複数のファイルシステムコマンドの少なくとも１つは、前記複数のファイルシステムコマンドの第２のコマンドとは異なる、ステップと、
前記第１の複合要求が、ファイルシステムに格納される１つより多いファイルに行われる関連しないファイルシステムコマンドを含むことを判定するステップと、
前記第１の複合要求が、ファイルシステムに格納される１つより多いファイルに行われる関連しないファイルシステムコマンドを含むことを判定する前記ステップに応答して、前記複数のファイルシステムコマンドが順々に処理されることを必要とせずに、前記複数のファイルシステムコマンドのそれぞれを別個のファイルシステムコマンドとして処理するステップと、
前記クライアントから第２の複数のファイルシステムコマンドを含む第２の複合要求を受け取るステップと、
前記第２の複合要求が、前記ファイルシステムに格納される単一のファイルに行われる関連するファイルシステムコマンドを含むことを判定するステップと、
前記第２の複合要求が、前記ファイルシステムに格納される単一のファイルに行われる関連するファイルシステムコマンドを含むことを判定する前記ステップに応答して、前記

第２の複数のファイルシステムコマンドを順次に処理するステップと
を実行させることを特徴とするコンピュータ可読記憶媒体。
【請求項２】
前記第２の複合要求は、作成／オープンコマンドを含む関連するコマンドを含み、前記第２の複数のファイルシステムコマンドを順次に処理するステップは、後続のそれぞれの関連するコマンドについて前記作成／オープンコマンドからのファイルハンドルを使用することを含むことを特徴とする請求項１に記載のコンピュータ可読記憶媒体。
【請求項３】
前記作成コマンドと共に受け取られる追加のコンテキストデータを処理するステップをさらに含むことを特徴とする請求項２に記載のコンピュータ可読記憶媒体。
【請求項４】
前記第１の複合要求が受け取られたチャネルとは別個のデータチャネル上で入出力関連の要求を処理するステップをさらに含むことを特徴とする請求項１に記載のコンピュータ可読記憶媒体。
【請求項５】
前記クライアントから前記ネットワークファイルサーバに送られる１組のコマンドからの少なくとも１つのコマンドを受け取るステップであって、前記１組のコマンドは、追加のコンテキストデータが添付されている作成コマンド、および／または別個のデータチャネル上でのデータ通信を要求するデータ関連コマンドを含む、ステップをさらに含むことを特徴とする請求項１に記載のコンピュータ可読記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
データ通信プロトコルに関する。
【背景技術】
【０００２】
ＳＭＢ（サーバメッセージブロック）プロトコルなど、今日なお使用されている多くのデータ通信プロトコルは、例えば、ネットワーク帯域幅が、通常は、制約されており、メモリが非常に貴重であるといった、コンピューティングリソースが非常に異なっていた時代に開発されたものである。結果として、現代のネットワークで使用されるとき、そのようなプロトコルは、全体的性能を制限することがある。例えば、メモリが不十分であったときに設計されているために、小さいバッファサイズが使用されており、大量のデータを伝達するのにより多くの往復を必要とする。
【０００３】
さらに、既存のＳＭＢプロトコルには、時間の経過と共に明らかになってきた他の限界もある。例えば、既存のＳＭＢプロトコルは、サービス拒否攻撃の影響を受けやすい。すなわち、このプロトコルの設計がこれらの攻撃に対抗するのを難しくしている。同様に、パケットセキュリティを保証する方法も煩雑である。また、例えば、信頼されるクライアントが信頼されないクライアントと同じサーバリソースを獲得するという点で、現在一般的なサービス品質のような操作を行う機構がない。
【発明の開示】
【発明が解決しようとする課題】
【０００４】
時間の経過と共にＳＭＢプロトコルの様々な改訂版、またはダイアレクトが開発されているが、それらのダイアレクトはそれぞれ、本質的には、何らかの追加機能を付加するために様々な部分を微調整するパッチベースの手法である。ゆえに、拡張性は簡単なものではない。要するに、既存のＳＭＢバージョンは、依然として使用頻度が高く、貴重なプロトコルではあるが、現代のネットワークリソースと共に使用されるときに理想的とはいえない。
【課題を解決するための手段】

【０００５】
簡単にいえば、本発明の様々な態様は、クライアントおよびサーバが、ファイル共有などのための通信に使用するデータ通信プロトコルを対象とする。クライアントは、クライアントが理解する１組のプロトコルダイアレクトを識別するサーバに折衝パケット（ｎｅｇｏｔｉａｔｉｏｎ　ｐａｃｋｅｔ）を送る。このパケットは、第２のデータ通信プロトコルによる通信をする能力があるサーバが、別の要求を必要とせずに、第１の通信プロトコルを使用すべきであると示すような形式である。サーバが第２のデータ通信プロトコルによる通信をする能力がある場合、サーバはそのように応答する。クライアントは、サーバによって示される対応するプロトコルを介してサーバとの通信を処理するドライバを呼び出す。１つの例示的実施形態では、第２の通信プロトコルはＳＭＢ２．０以上である。
【０００６】
このプロトコルの他の態様および機能拡張には、追加のコンテキストデータが添付された作成コマンド、複数の関連するコマンドまたは関連しないコマンドを含む複合コマンドが含まれ得る。別の態様および機能拡張には、別個のデータチャネル上でのデータ転送を要求することに関連する多重チャネルコマンド、保護された接続が確立されることを保証するよう求める署名付き機能検証要求、および要求に応答してサーバから拡張された誤りデータを転送することのできる機能が含まれる。
【０００７】
サーバは、複合要求を受け取ると、その複合要求が関連しないコマンドを含むか、それとも関連するコマンドを含むか判定する。複合要求が関連しないコマンドを含むとき、各要求は別々の要求として処理され、そうではなく、複合要求が関連するコマンドを含むとき、各要求は順次に処理される。作成／オープンコマンドを含む関連するコマンドのとき、作成／オープンコマンドからのファイルハンドルは、例えば、クライアントからのハンドル戻りを待たずに、サーバにおいて各後続関連コマンドごとに使用される。
【０００８】
その他の利点は、以下の詳細な説明を図面と併せて読めば明らかになるであろう。
【０００９】
本発明を、例としてあげるにすぎないが、添付の図に示す。図において類似の参照番号は類似の要素を示す。
【発明を実施するための最良の形態】
【００１０】
例示的動作環境
図１に、本発明が実施され得る適切なコンピューティングシステム環境１００の一例を示す。コンピューティングシステム環境１００は、適切なコンピューティング環境の一例にすぎず、本発明の用途または機能の範囲に関するどんな限定を示唆するものでもない。また、コンピューティング環境１００は、例示的動作環境１００に示す構成要素のいずれか１つまたはそれらの組み合わせに関連するどんな依存関係または要件を有するものであるとも解釈すべきではない。
【００１１】
本発明は、他の多数の汎用または専用コンピューティングシステム環境または構成と共に動作する。本発明と共に使用するのに適し得るよく知られているコンピューティングシステム、環境、および／または構成の例には、それだけに限らないが、パーソナルコンピュータ、サーバコンピュータ、ハンドヘルドまたはラップトップ機器、タブレット機器、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、プログラム可能家庭用電化製品、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、上記のシステムまたは機器のいずれか、印刷サーバやプリンタ自体といった様々なネットワークアプライアンス機器の１つ、ならびにＮＡＳ記憶装置を含む分散コンピューティング環境などが含まれる。
【００１２】
本発明は、コンピュータにより実行される、プログラムモジュールなどのコンピュータ

実行可能命令の一般的なコンテキストで説明され得る。一般に、プログラムモジュールには、個々のタスクを実行し、または個々の抽象データ型を実施するルーチン、プログラム、オブジェクト、コンポーネント、データ構造などが含まれる。また、本発明は、タスクが、通信ネットワークを介してリンクされたリモート処理装置によって実行される分散コンピューティング環境でも実施され得る。分散コンピューティング環境では、プログラムモジュールは、メモリ記憶装置を含むローカルおよび／またはリモートのコンピュータ記憶媒体に置くことができる。

【００１３】

図１を参照すると、本発明を実施する例示的システムは、コンピュータ１１０の形で汎用コンピューティングデバイスを含む。コンピュータ１１０の構成要素には、それだけに限らないが、処理装置１２０、システムメモリ１３０、およびシステムメモリを含む様々なシステム構成要素を処理装置１２０に結合するシステムバス１２１が含まれ得る。システムバス１２１は、様々なバスアーキテクチャのいずれかを使用するメモリバスまたはメモリコントローラ、周辺バス、およびローカルバスを含む数種類のバス構造のいずれでもよい。例としてあげるにすぎないが、そのようなアーキテクチャには、ＩＳＡ（Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＭＣＡ（Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＥＩＳＡ（Ｅｎｈａｎｃｅｄ ＩＳＡ）バス、ＶＥＳＡ（Ｖｉｄｅｏ Ｅｌｅｃｔｒｏｎｉｃｓ Ｓｔａｎｄａｒｄｓ Ａｓｓｏｃｉａｔｉｏｎ）ローカルバス、およびメザニンバスとも呼ばれるＰＣＩ（Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ）バスが含まれる。

【００１４】

コンピュータ１１０は、通常、様々なコンピュータ可読媒体を含む。コンピュータ可読媒体は、コンピュータ１１０によってアクセスされ得る任意の使用可能な媒体とすることができ、それには揮発性と不揮発性両方の媒体、取り外し可能と取り外し不能両方の媒体が含まれる。例としてあげるにすぎないが、コンピュータ可読媒体には、コンピュータ記憶媒体および通信媒体が含まれ得る。コンピュータ記憶媒体には、コンピュータ可読命令、データ構造、プログラムモジュール、その他のデータなどの情報を記憶するための任意の方法または技術で実施された、揮発性および不揮発性、取り外し可能および取り外し不能の媒体が含まれる。コンピュータ記憶媒体には、それだけに限らないが、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリなどのメモリ技術、ＣＤ－ＲＯＭ、ディジタル多用途ディスク（ＤＶＤ）などの光ディスク記憶、磁気カセット、磁気テープ、磁気ディスク記憶などの磁気記憶装置、あるいは所望の情報を格納するのに使用でき、コンピュータ１１０によってアクセスされ得る他の任意の媒体が含まれる。通信媒体は、通常、コンピュータ可読命令、データ構造、プログラムモジュールまたはその他のデータを、搬送波や他の搬送機構などの変調されたデータ信号中に具現化するものであり、任意の情報伝達媒体が含まれる。「変調されたデータ信号」という用語は、その特性の１つまたは複数が、その信号に情報を符号化するような方式で設定または変更されている信号を意味する。例としてあげるにすぎないが、通信媒体には、有線ネットワークや直接配線接続などの有線媒体、および音響、ＲＦ、赤外線、その他の無線媒体などの無線媒体が含まれる。また、上記のいずれかの組み合わせも、コンピュータ可読媒体の範囲内に含めるべきである。

【００１５】

システムメモリ１３０は、読出し専用メモリ（ＲＯＭ）１３１やランダムアクセスメモリ（ＲＡＭ）１３２などの揮発性および／または不揮発性メモリの形でコンピュータ記憶媒体を含む。基本入出力システム（ＢＩＯＳ）１３３は、始動時などに、コンピュータ１１０内の諸要素間での情報転送を支援する基本ルーチンを含み、通常、ＲＯＭ１３１に格納される。ＲＡＭ１３２は、通常、処理装置１２０から直ちにアクセス可能であり、かつ／または処理装置１２０によって現在操作されているデータおよび／またはプログラムモジュールを含む。例としてあげるにすぎないが、図１に、オペレーティングシステム１３４、アプリケーションプログラム１３５、その他のプログラムモジュール１３６、およびプログラムデータ１３７を示す。

【００１６】
また、コンピュータ１１０は、他の取り外し可能／取り外し不能、揮発性／不揮発性コンピュータ記憶媒体も含み得る。例にすぎないが、図１に、取り外し不能、不揮発性磁気媒体との間で読取りまたは書込みを行うハードディスクドライブ１４１、取り外し可能、不揮発性磁気ディスク１５２との間で読取りまたは書込みを行う磁気ディスクドライブ１５１、およびＣＤ－ＲＯＭや他の光媒体などの取り外し可能、不揮発性光ディスク１５６との間で読取りまたは書込みを行う光ディスクドライブ１５５を示す。例示的動作環境で使用され得る他の取り外し可能／取り外し不能、揮発性／不揮発性のコンピュータ記憶媒体には、それだけに限らないが、磁気テープカセット、フラッシュメモリカード、ディジタル多用途ディスク、ディジタルビデオテープ、ソリッドステートＲＡＭ、ソリッドステートＲＯＭなどが含まれる。ハードディスクドライブ１４１は、通常、インターフェース１４０などの取り外し不能メモリインターフェースを介してシステムバス１２１に接続され、磁気ディスクドライブ１５１および光ディスクドライブ１５５は、通常、インターフェース１５０などの取り外し可能メモリインターフェースによってシステムバス１２１に接続される。

【００１７】
前述の、図１に示す各ドライブおよびそれらに関連するコンピュータ記憶媒体は、コンピュータ１１０のためのコンピュータ可読命令、データ構造、プログラムモジュールおよびその他のデータの記憶を提供する。図１では、例えば、ハードディスクドライブ１４１は、オペレーティングシステム１４４、アプリケーションプログラム１４５、その他のプログラムモジュール１４６、およびプログラムデータ１４７を格納するものとして示されている。これらのコンポーネントは、オペレーティングシステム１３４、アプリケーションプログラム１３５、その他のプログラムモジュール１３６、およびプログラムデータ１３７と同じでも、異なっていてもよいことに留意されたい。オペレーティングシステム１４４、アプリケーションプログラム１４５、その他のプログラムモジュール１４６、およびプログラムデータ１４７には、少なくともそれらが異なるコピーであることを示すために、図では異なる番号が付与されている。ユーザは、タブレットまたは電子ディジタイザ１６４や、キーボード１６２や、マイクロホン１６３や、一般にマウス、トラックボール、タッチパッドなどと呼ばれるポインティングデバイス１６１といった入力装置を介してコンピュータ１１０にコマンドおよび情報を入力することができる。図１に示していない他の入力装置には、ジョイスティック、ゲームパッド、衛星パラボラアンテナ、スキャナなどが含まれ得る。上記その他の入力装置は、しばしば、システムバスに結合されたユーザ入力インターフェース１６０を介して処理装置１２０に接続されるが、パラレルポート、ゲームポート、ユニバーサルシリアルバス（ＵＳＢ）といった他のインターフェースおよびバス構造によっても接続され得る。また、モニタ１９１または他の種類の表示装置も、ビデオインターフェース１９０などのインターフェースを介してシステムバス１２１に接続される。また、モニタ１９１は、タッチスクリーンパネルなどを用いて一体化することもできる。モニタおよび／またはタッチスクリーンパネルは、タブレット型パーソナルコンピュータなどのように、コンピューティングデバイス１１０が組み込まれているハウジングに物理的に結合され得ることに留意されたい。また、コンピューティングデバイス１１０などのコンピュータは、スピーカ１９５やプリンタ１９６など他の周辺出力装置を含むこともでき、それらは、出力周辺装置インターフェース１９４などを介して接続され得る。

【００１８】
コンピュータ１１０は、リモートコンピュータ１８０など、１つまたは複数のリモートコンピュータへの論理接続を使用するネットワークで接続された環境で動作し得る。リモートコンピュータ１８０は、パーソナルコンピュータ、ハンドヘルド機器、サーバ、ルータ、ネットワークＰＣ、ピアデバイスまたはその他一般のネットワークノードとすることができ、通常は、コンピュータ１１０に関連して前述した要素の多くまたはすべてを含むが、図１には、メモリ記憶装置１８１だけが示されている。図１に示す論理接続には、ロ

ーカルエリアネットワーク（ＬＡＮ）１７１および広域ネットワーク（ＷＡＮ）１７３が含まれるが、他のネットワークも含まれ得る。そのようなネットワーク環境は、オフィス、企業規模のコンピュータネットワーク、イントラネットおよびインターネットではよく見られるものである。

【００１９】

ＬＡＮネットワーク環境で使用されるとき、コンピュータ１１０は、ネットワークインターフェースまたはアダプタ１７０を介してＬＡＮ１７１に接続される。ＷＡＮネットワーク環境で使用されるとき、コンピュータ１１０は、通常、モデム１７２またはインターネットなどのＷＡＮ１７３を介して通信を確立する他の手段を含む。モデム１７２は、内蔵でも外付けでもよく、ユーザ入力インターフェース１６０または他の適切な機構を介してシステムバス１２１に接続され得る。ネットワークで接続された環境では、コンピュータ１１０に関連して示すプログラムモジュール、またはその一部は、リモートのメモリ記憶装置にも格納され得る。例にすぎないが、図１に、リモートアプリケーションプログラム１８５を、メモリ装置１８１上にあるものとして示す。図示のネットワーク接続は例であり、コンピュータ間で通信リンクを確立する他の手段も使用され得ることが理解されるであろう。

【００２０】

データ通信プロトコル

本明細書で説明する技術の様々な態様は、ＳＭＢプロトコルの後期バージョン（２．ｘ以上）などのデータ通信プロトコルを対象とする。本明細書で一般的に説明する１つの例示的実装形態では、ＳＭＢプロトコルがファイルデータ転送に使用される。しかしながら、容易に理解され得るように、本発明は、本明細書で説明する任意の特定の実装形態や例はもちろんのこと、ファイルデータにも限定されるものではない。そうではなく、プリンタ、名前付きデータパイプ、一般の装置などとの通信での使用を含めて、本発明を実施する多数のやり方が実行可能である。したがって、本発明は、本明細書で使用する特定のファイルベースの例のいずれにも限定されず、コンピューティング一般において利益および利点を提供する多数のやり方で使用され得る。

【００２１】

本明細書で説明する技術の他の様々な態様は、ファイルサーバ対話がその上に構築され得るＳＭＢの新しい改訂版を対象とする。理解されるように、より拡張性があり、新しい機能を用いて更新するのがより容易であると同時に、既存の（上位）機能をサポートするより軽量なプロトコルが提供される。

【００２２】

図面の図２を見ると、クライアント２０２が１つまたは複数の通信チャネルを介してサーバ２０４と通信するネットワーク環境の一例を表すブロック図が示されている。クライアントマシン２０２とサーバ２０４の機能およびコンポーネントを、図１の主コンピュータシステム１１０とリモートコンピュータシステム１８０のような、２つの別々のコンピュータ内に位置するものとして説明するが、これら２つのコンピュータのコンポーネントまたはそれらによって実行される機能は、１つのマシン上で提供することもでき、いくつかのコンピュータにわたって分散させることもできる。

【００２３】

アプリケーションプログラム２０６からのネットワークファイルシステムコマンドは、ファイルシステム２１２に対するコマンドを実行するために相手先の共通ネットワークモジュール（ＳＲＶＮＥＴ）２１０と通信する、クライアントリダイレクタコンポーネント２０８によって処理される。そのようなコマンドが処理される前に、クライアントとサーバが同意する通信プロトコル、一般に、両方の理解する最新バージョン／ダイアレクトが折衝される。

【００２４】

一般に、クライアント２０２は、接続を確立し、次いで、図３に一般的に表すように、サーバ２０４と折衝して最終的にセッションをセットアップする。クライアントは、サー

バに対して、それがＳＭＢ２．ｘクライアント（本明細書で使用する場合、２．ｘという数は、既存のＳＭＢ１．ｘバージョンに対する任意のより新しいバージョンを表す）であることを直接示すこともできるが、クライアントは、後方互換の折衝パケットによって折衝することもできる。このようにして、クライアントは、ＳＭＢ１．ｘだけにしか対応しないサーバとも通信し、しかも、より上位レベルの折衝での試行に失敗した場合でも、別の接続をセットアップすることを必要とせずに通信することができる。同時に、各プロトコルを実施するコードは、それ独自の独立ドライバにパッケージされ得る。

【００２５】

１つの例示的実装形態では、クライアントＳＭＢエンジンコンポーネント２２０は、サーバ（サーバ２０４など）に対して、クライアント２０２が少なくとも１つのＳＭＢ１．０セッションを折衝していることを示すパケットを提供する。ＳＭＢ１ダイアレクトと、このプロトコルの新しいＳＭＢ２改訂版の両方を話すクライアント２０２では、クライアントは、従来のＳＭＢ１折衝パケットではあるが、さらに、このパケットが、利用可能であれば、実際にはＳＭＢ２．ｘを要求しているという指示も含む折衝パケットを送ることができる。ＳＭＢ２対応サーバは、この要求を検出し、ＳＭＢ２折衝応答で応答する。より具体的には、クライアント２０２がＳＭＢ２．ｘに対応することを示すために、このＳＭＢ１．０折衝パケットは、そのうちの１つが、クライアントがＳＭＢ２．ｘ通信にも対応することを示す、１組のダイアレクトストリングを含む。

【００２６】

ゆえに、クライアント２０２は、クライアント２０２がサポートするマイナーバージョン番号を含む初期折衝を送る。１つの最新の改訂版は０（ゼロ）、すなわち、ＳＭＢ２．０である。将来において、クライアントは、ダイアレクト改訂版の任意のサブセットをサポートすると主張し得る。

【００２７】

サーバ２０４は、このパケットを受け取ると、その対応能力に基づいて応答する。サーバ２０４は、任意の１．ｘダイアレクト情報と共に、ＳＭＢ１．０折衝応答で応答することができ、ＳＭＢ２．ｘ通信対応の場合には、ＳＭＢ２．０折衝応答で応答する。また、通常は、クライアント２０２が提供したダイアレクトバージョンの中でサーバ２０４が処理し得る最大数のバージョンである、ダイアレクトストリングの１つにマッチする特定のＳＭＢダイアレクト改訂版も返され得る。

【００２８】

このために、サーバ２０４は、クライアント２０２がどのダイアレクト改訂版を話すかを知ると、これらを、サーバ２０４が理解する改訂版と比較し、好ましい共通ダイアレクト改訂版（通常は最高のものになる）を返す。例えば、サーバはダイアレクト１～８をサポートするが、クライアントは１、２および４だけをサポートする場合、サーバは４を返す。これにより、クライアント２０２は、それがサーバ２０４にどのコマンドを送ることができるか明確に理解することができる。どのダイアレクトを使用すべきか選択するために、ＳＲＶＮＥＴモジュール２１０は、基本的に、折衝を開始し、１つのＳＭＢプロバイダがパケット内容に基づいてこの通信セッションを処理することに同意するまで、バージョン／ダイアレクトに関して最高から最低の順で、ＳＲＶＮＥＴモジュール２１０が持つ各ＳＭＢプロバイダ２２２_{1}～２２２_{m}にパケットを渡す。その後、この接続上での通信がそのプロバイダ、この例ではＳＭＢ２．０プロバイダ２２２_{m}に経路指定される。

【００２９】

クライアント側において、ＳＭＢエンジンコンポーネント２２０は応答を受け取り、応答中のバージョン／ダイアレクト情報に基づいて、サーバと通信するのにどのクライアントＳＭＢコンポーネント２２４_{1}～２２４_{n}を使用すべきか知る。このようにして、クライアント２０２とサーバ２０４は共に、所与のセッションにどのＳＭＢダイアレクトを使用すべきかについて合意する。クライアントは、それぞれ異なるバージョン／ダイアレクトの、同時に走る複数のＳＭＢコンポーネント２２４_{1}～２２４_{n}を持つことができ、それによって、例えば、クライアントは、ＳＭＢ２．ｘを介して別のサーバと通信するのと同時

に、ＳＭＢ１．ｘを介してあるサーバと通信することができることに留意されたい。

【００３０】

また、サーバ２０４は、クライアントに対して、サーバ２０４がセキュリティ署名を必要とするかどうかクライアント２０２に知らせるセキュリティモードを含めて、他の情報も返す。以前は、セキュリティ署名は利用可能であったが、最初の少数の（例えば、対応能力折衝）パケットは妨げられず、そのため、攻撃者はクライアントに、攻撃者がその脆弱性を知っているより下位のプロトコルを強制することができたことに留意されたい。

【００３１】

保護された接続は、（署名が使用可能であるか否かを問わず）署名付きの別の対応能力検証往復を提供することによって動作する。図７に、セットアップに続くそのような要求／応答を示す。サーバが、別のエンティティではなくそれが応答したことを実際に検証することができるように、ＩＰアドレスなど他の情報がパケットに入れられ得る。署名は、ＩＰＳＥＣまたは他の任意の形のネットワークセキュリティがアクティブである場合には、オフにされ得る。

【００３２】

サーバ２０４は、サーバの対応能力ビット（ｃａｐａｂｉｌｉｔｉｅｓ　ｂｉｔｓ）、例えば、サーバがＤＦＳ（分散ファイルシステム）対応であるかどうか、およびサーバがＬＷＩＯ（軽量入力）対応であるかどうかなどを返すことができる。クライアント２０２は、それが理解しないどんな対応能力ビットも無視し、これは、サーバ２０４がクライアントの対応するバージョンより新しいバージョンを持つ場合に起こり得る。折衝のやり取りで返され得る他の情報には、サーバの一意のＩＤ、サーバが受け入れる最大読取り／書込みサイズ、より高速の書込み処理のためのデータオフセットヒント、サーバの現在のシステム時刻、および拡張されたセキュリティの場合に認証をシードするのに使用されるセキュリティ情報が含まれる。

【００３３】

セッションセットアップは新しいセッションの認証プロセスを処理し、複数の往復イベントとすることができる。クライアント２０２は、ローカルセキュリティシステムに、ネットワークを介して送るセキュリティブロブを問い合わせ、以下で説明するように、対応能力、最大サイズフィールド、およびＶｃＮｕｍｂｅｒを入力して、最初のセッションセットアップを送る。サーバ２０４は、このブロブを受け取り、それをセキュリティシステムに渡す。サーバ２０４は、さらなる情報が必要であると判断した場合、誤りコードＳＴＡＴＵＳ＿ＭＯＲＥ＿ＰＲＯＣＥＳＳＩＮＧ＿ＲＥＱＵＩＲＥＤと共に、それ自体のセキュリティブロブを返す。クライアント２０２は、このブロブをローカルセキュリティシステムに戻し、プロセスは、失敗が発生し、または認証に成功するまで繰り返す。

【００３４】

ＶｃＮｕｍｂｅｒは、サーバ２０４に、この同じクライアント２０２から確立された他の接続があり得るかどうか知らせる。これがゼロである場合、サーバ２０４は、このクライアントから他の接続が行われていないと仮定し、見つかったそのようなどんな接続も（それらが失効しているものと仮定して）切断する。ＶｃＮｕｍｂｅｒが１以上である場合、サーバ２０４は、既存のどんな接続も切断しない。

【００３５】

チャネルは、サーバ２０４に、このクライアント２０２が既存のセッションと別の接続を確立しようとしていることを知らせる。このセッションは、このセッションセットアップがそこから受け取られたユーザ／コンピュータ対によって識別され得る。チャネルは、同じＴｒｅｅＩｄ／ＵｓｅｒＩｄ／ＰｒｏｃｅｓｓＩｄ／ＦｉｌｅＩｄ情報を共有する。チャネル認証では、認証ブロブは、クライアント２０２とサーバ２０４が相互に相手を認証し合うことができるように、第１のチャネルを介して暗号化され、第２のチャネルを介して送り返されるチャレンジ／レスポンスとすることができる。正常な応答時に、サーバ２０４は、クライアント２０２に、そのどちらかが妥当な場合には、クライアントがゲストとして認証されているか、それともヌルユーザとして認証されているかも通知する。

【００３６】
セッションがセットアップされると、クライアント２０２は、作成、読取り、書込みおよびクローズを含む、以下で説明する様々なコマンドを使ってデータ転送を行うと共に、ファイルロッキングおよびディレクトリ関連の操作を行うことができる。これらのコマンドを使用すると、サーバは、クライアントによるサーバリソース使用を制御することができる。また、プロトコルは、どんな情報が伝達されるか、およびそれがどのようにして伝達されるかに関して、いくつかの効率改善も提供する。
【００３７】
図４に一般的に表すように、この作成コマンドは、コマンドにコンテキスト情報を添付させるように拡張されている。一般に、コンテキスト情報は、作成コマンドにタグ付けされる随意の追加の作成パラメータを含む。例えば、トランザクションファイルシステム関連の作成コマンド用のトランザクション識別子が添付され得る。サーバがその追加コンテキスト情報を理解する限り、サーバは、拡張情報の知らせを受け（サーバは、それらが理解しない追加データを無視することに留意されたい）、そのコンテキストに関連付けられた情報を返すことができる。容易に理解されるように、これは、プロトコルを変更せずに追加機能を提供し、本質的に組み込みの拡張可能性を提供するものである。
【００３８】
コマンドＩＤおよびダイアレクト改訂番号が、以下に示すように、新しいＳＭＢヘッダで提供される。このヘッダは、コマンドフィールドに、ＵＣＨＡＲではなく、ＵＳＨＯＲＴを持つ。このＵＳＨＯＲＴの最初のバイトを使ってダイアレクトを表し、後のバイトを使ってコマンドを表すことによって、コマンド表が既存のコマンドのために明確に定義され、大部分は後で拡張するためにオープンとされる。一般に、クライアントは、所与のコマンドを発行する関数を指し示すポインタを含む各ダイアレクトごとの表を維持することができる。単一のダイアレクトをサポートするクライアントでは、この表は以下のように示されるはずである。
【００３９】
【表１】

コマンド	ダイアレクト#1
Create	SmbCreate1
Read	SmbRead1
Write	SmbWrite1
Close	SmbClose1

【００４０】
キャッシュ機能では、クローズに際して、ファイルからより多くの情報が検索され得る。そのため、この新しい機能をサポートするために新しいクローズコマンドが提供される。今やクライアントは２つのダイアレクトをサポートしており、表は以下のように示される。
【００４１】
【表２】

コマンド	ダイアレクト#2	ダイアレクト#1
Create	SmbCreate1	SmbCreate1
Read	SmbRead1	SmbRead1
Write	SmbWrite1	SmbWrite1
Close	SmbClose2	SmbClose1

【００４２】
変更されたクローズコマンドを除いて、機能の大部分は同じままであることに留意され

たい。また、クライアントは、今や、ダイアレクト２サーバと話し、新しい機能を使用することもできるが、ダイアレクト１サーバには依然として古い機能を使用する。ダイアレクト１サーバとの通信に変更はない。

【００４３】

技術が進化するにつれて、相対的にはるかに大きい読取りおよび書込みを実行することができるなど、新しいネットワークハードウェアが利用可能になる。このリリースでは、ダイアレクト＃３が提供され、それによって表が以下のように拡大される。

【００４４】

【表３】

コマンド	ダイアレクト#3	ダイアレクト#2	ダイアレクト#1
Create	SmbCreate1	SmbCreate1	SmbCreate1
Read	SmbRead3	SmbRead1	SmbRead1
Write	SmbWrite3	SmbWrite1	SmbWrite1
Close	SmbClose2	SmbClose2	SmbClose1

【００４５】

そのような表を備えるクライアントは、３つのダイアレクトを話すことができ、各ダイアレクトで利用可能な機能を利用する。この方法を使用することの利点の中には、各ＳＭＢコマンドが、（ダイアレクト｜コマンド）で構成されるために、それがそこに導入されたダイアレクトに逆マップされ得ることが含まれる。これは、コマンドがいつ導入されたか、およびどんなサーバがそれらのコマンドをサポートしているか判断するのを容易にする。所与のコマンドの機能が新しいダイアレクトで変更されない場合、そのコードは変化しない。機能が変更される場合、下位レベルのインターフェースコードは変化せず、むしろ、新しい機能をサポートするための新しいコードが追加される。

【００４６】

サーバ側では、サーバディスパッチ表が、（ダイアレクト）と（コマンド）の間のダブルスイッチ（ｄｏｕｂｌｅ　ｓｗｉｔｃｈ）になる。これは、コードにおいて新しい機能を論理的に分離し、それを理解し変更するのを容易にすることができる。

【００４７】

効率性を提供するプロトコルの一態様を見ると、複数のコマンドが単一のパケット（またはいくつかのより少ない数のパケット）に複合され得る。ゆえに、複雑なタスクが、クライアント２０２とサーバ２０４の間の往復回数を低減するようなやり方で実行され得る。例をあげると、複合要求パケットは、ファイルを作成する／開くコマンド、ファイルに書き込むコマンドおよびファイルから読み取るコマンドを含み得る。このようにして、複合は、（例えば、同じファイルハンドルを持つ）関連する操作を処理すると共に、関連しない操作が組み合わされることも可能にする。

【００４８】

関連する要求を複合する一例を、図５に一般的に表す。図５では、（例えば、図４などとは対照的に）単一の要求で書込みおよび読取りを処理し、適切なパラメータを提供することができる。図５に表すように、単一の要求は、１つの複合応答および／または個々の応答を、例えば、それらがいつ完了するかに応じて、受け取ることができることに留意されたい。作成／オープンし、読み取り、書き込み、閉じるといった、より複雑な要求も単一の要求に含まれ得る。これは、関連する操作を持つとパケットにマークすることによって達成される。すなわち、サーバは、それが作成／オープンに続いて受け取るファイルハンドルが、複合要求のその他のコマンドに適用されることを知る。しかしながら、関連する複合要求は、それらがパッケージされている順序で処理され、ゆえに、それらが送信前に正しく順序付けられることを保証するのはクライアントの責任であることに留意されたい。

【００４９】

ＳＭＢ２における複合は、ＳＭＢ１に存在していた複雑な規則より簡単である。このために、（以下で詳述する）ＳＭＢ２＿ＨＥＡＤＥＲは、現在のコマンドのヘッダからの次のコマンドのヘッダのオフセットを識別するのに使用される「ＮｅｘｔＯｆｆｓｅｔ」を含む。各コマンドは、別々のＭｅｓｓａｇｅＩｄを含む、独自のＳＭＢ２＿ＨＥＡＤＥＲを備える。１つまたは複数のサーバ応答は、図５に示すように、単一の複合応答として、または別々の応答としてもたらされ得る。失敗の場合、応答は、他の任意の失敗したコマンドと同じになるはずである。

【００５０】

関連しないメッセージでは、コマンドは、常に、あたかもそれらが別々に受け取られたかのように処理される。これは、リダイレクタまたは中間コンポーネントが関連しないパケットを自動的に複合することを可能にする。特に、遅延時間が往復時間に対して相対的に小さい場合には、遅延を使って複合すべきパケットが獲得され得る。サーバはそれらのパケットが別々に受け取られたとみなすため、サーバは、そのような複合の関連しない要求をアンパックするように別に変更される必要がない。しかしながら、その複合を実行したエンティティは、サーバが別に別々の応答を結合することができるため、任意の複合応答を分離しなければならないことがある。

【００５１】

関連するモードは、クライアントが、あるコマンドの結果が潜在的に次のコマンドで使用される、順次に実行されるべき一連のコマンドを送ることを可能にする。そのようなコマンドは、同じセッション／プロセス／ツリー／ファイルＩＤを共有し、順次に実行され、最初の誤り発生時に処理を停止する。失敗の後に処理すべき他のコマンドがあった場合、それらの操作は、ＳＴＡＴＵＳ＿ＮＯＴ＿ＰＲＯＣＥＳＳＥＤで直ちに失敗する。これがどのようにして使用され得るかの一例が、セッションセットアップをツリー接続と組み合わせることである。セッションを確立するのに失敗した場合、ツリー接続は決して試行されず、ＳＴＡＴＵＳ＿ＮＯＴ＿ＰＲＯＣＥＳＳＥＤで失敗する。セッションセットアップに成功した場合、セッションセットアップコマンドからのＳｅｓｓｉｏｎＩｄを使ってツリー接続が行われる。同じ方法を使って、作成に続けてＱｕｅｒｙＦｉｌｅＩｎｆｏｒｍａｔｉｏｎ、または作成／読取り／クローズのセットさえも行うことができるはずである。

【００５２】

また、条件付きまたは暗黙の複合も実行可能である。例えば、１回の往復で、小さいファイルは開いて自動的に獲得するが、大きいファイルは開くだけとする、開き、かつ、ファイルが６４ＫＢ未満であれば読み取るなどの条件付複合コマンドを送ることができる。また、明示的に要求されなくとも、ディレクトリを開く要求に応答してディレクトリ列挙データを自動的に返すなどの暗黙の複合も、往復回数を削減することができる。そのような拡張された複合の利益および利点は、高遅延ネットワークにおいて増大する。

【００５３】

このプロトコルによってうまく効率が改善される別のやり方は、多重チャネル通信によるものである。クライアントとサーバの間で、データをストリーミングする交替チャネルを指定するコマンドを用いて、コマンド用トランスポート接続が使用され得る。例えば、読取り要求は、オフセットおよび長さ、ならびにデータを読み込む交替チャネルを指定することができる。書込み要求も同様に動作する。図６に、オフセット０から開始し、データがデータチャネル５にストリーミングされるよう要求する１ＧＢの読取り要求の一例を示す。

【００５４】

交替チャネル上のストリーミングデータは、パケットヘッダを含み、それを処理する必要がなくなることを含めて、いくつかの利益を提供する。クライアントは、バッファを事前に配置し、そこにデータをストリーミングさせることができ、従来方式の単一チャネル通信の場合のように、あるバッファから別のバッファにコピーする必要がなくなる。別の利益は公平性であり、それは、例えば、制御チャネル上のある要求は、その他の要求が処

理される前に、完了すべき大量のデータ（例えば５ＧＢ）が送信されるのを待つ必要がないという点においてである。というのは、その５ＧＢはデータチャネルを介して進むからである。

【００５５】

複数のＮＩＣがより一般的になるにつれて、プロトコルは、任意の利用可能なネットワーク帯域幅を利用する。これは、接続が確立されているトランスポート（またはＮＩＣ）を問わず、同じセッションのために複数の接続にまたがって処理することを含む。専用ハードウェアも用いられ得る。

【００５６】

ゆえに、ＳＭＢ２．ｘを用いると、セッションは接続にバインドされない。そうではなく、異なる物理接続にまたがって存在する多重「チャネル」が確立され得る。セッションはこれらの接続のそれぞれに存在することができ、ファイルおよびプロセスを参照するのに使用されるＩＤは、チャネル間で共通である。これは、名前空間操作および作成を行うための通常のチャネルを備えるが、利用可能なときには、読取りおよび書込みに専用ネットワークハードウェアを使用することを可能にする。しかも、小さいネットワークグリッチはデータ損失を生じ得ない。というのは、１つのチャネルがあるセッションに対して開いたままである限り、そのセッションは活動状態のままだからである。本明細書では、セッションセットアップコマンドおよび読取り／書込みコマンドに関連して様々な実装詳細を説明する。

【００５７】

例として、企業の公衆網を介してサーバに単純なＴＣＰによる接続を確立するクライアントを考える。これは最初の接続であり、そのため、常にチャネル０である。クライアントとサーバ両方の側が、それらがデータ転送を行うための私設網を備えている（例えば、それぞれがギガビットカードを備えている）ことを検出すると、クライアントおよびサーバは、チャネル１として、このカードを介した第２の接続を確立することができる。クライアントがいくつかのファイルをブラウズしている間、ディレクトリ問い合わせはチャネル０を介して送られ、他方、データはチャネル１を介して送られている。クライアントが、サーバ上で暗号化されているいくつかのディレクトリにブラウズしようとする場合、クライアントがデータを要求すると、リダイレクタは、そのデータが機密であることを理解し、そのため、その上でＩＰ　Ｓｅｃ（ＩＰセキュリティ）をアクティブにしているサーバへの新しいチャネル（チャネル２）を確立する。クライアントは、機密データを要求するときに、それがチャネル２を介して送られるよう求めるが、通常の機密性の低いデータは（より高速であるため）引き続きチャネル１を介してもたらされ得る。

【００５８】

容易に理解されるように、サービス品質およびセキュリティ改善の機会は、単純な帯域幅ゲインと共に、著しい利益を提供する。チャネル読取り／書込みに際して、サーバ／クライアントは、データが読み取られる前に受信バッファを置くことができ、そのため、この機構は、さらに、データ移動からコピーする必要をなくすことができ、これは、サーバ／クライアント拡張性も改善し得ることに留意されたい。

【００５９】

さらに、ＳＭＢ誤りパケットには、随意のデータをタグ付けすることができる。ゆえに、何かがなぜ失敗したかが記述され、それが値を提供し得る。図８に一般的に表すように、シンボリックリンク評価は、随意データをタグ付けすることがクライアントに役立つ情報を提供する一例である。本質的に、クライアント作成要求は、実際には別のパスへのシンボリックリンクであるパスを求めることによって失敗し得る。単にその要求を失敗させるのではなく、新しいパスを提供する情報は、クライアントが最終的に成功することになる再解析パスに変更することを可能にする。成功するパスを見つけるには、いくつかの要求に及ぶ反復が必要とされ得ることに留意されたい。

【００６０】

例示的プロトコル定義

新しいヘッダは６４バイト構造（例えば、１つの現在の構造の２倍のサイズなど）である。

【００６１】

【表４】

```
typedef struct _SMB2_HEADER {
    UCHAR           Protocol[4];        // Contains 0xFE,'S','M','B'
    USHORT          StructureSize;      // = sizeof(SMB2_HEADER).
                                        //   (versioning)
    USHORT          Epoch;              // incremented every time the
                                        // server restarts

    NTSTATUS        Status;             // Status of the command

    USHORT          Command;            // The command for this packet
                                        // union
    {
        USHORT      CreditsRequested;   // On client send, request for more
                                        // credits
        USHORT      CreditsGranted;     // On server response, credits
                                        // granted to client
    };

    ULONG           Flags;
    ULONG           Reserved;

    UINT64          MessageId;          // Identifies this message
                                        // send/response
    union
    {
        struct
        {
            UINT64  ProcessId;          // Process identifier
            UINT64  SessionId;          // Session identifier
            ULONG   TreeId;             // Tree Connect identifier
        };

        struct
        {
            UINT64  AsyncId;            // Used to identify long standing
                                        // commands
        };
    };

    UCHAR           Signature[8];       // Signature for the packet

    ULONG           NextCommand;        // Offset from to next
} _SMB2_HEADER, *PSMB2_HEADER;
```

【００６２】

Ｐｒｏｔｏｃｏｌは、単に、パケットを識別するプロトコル識別子である。既存のＳＭＢ実装形態では、これは、｛０ｘＦＦ，’Ｓ’，’Ｍ’，’Ｂ’｝からなる。新しいプロトコルでは、これを｛０ｘＦＥ，’Ｓ’，’Ｍ’，’Ｂ’｝とする。

【００６３】

ＳｔｒｕｃｔｕｒｅＳｉｚｅは、ＳＭＢ２＿ＨＥＡＤＥＲ構造のサイズを識別し、後で他の変更が導入される場合に、ヘッダ自体の内でのマイナーバージョン管理に使用される。

【００６４】

Ｅｐｏｃｈはサーバの「バージョン数」を表す。これは、サーバがサイクル動作をした（またはサーバサービスが停止され、開始された）ときに、クライアントに対して、サーバが切断にまたがって状態を維持することができているかどうか示すために増分される。これは、永続的ハンドルを用いたその後の使用のためであり、当面の間は「予約済み」とみなされ得る。

【００６５】

Ｓｔａｔｕｓは、既存のＳＭＢ実装形態の場合と同様に、所与の操作での誤り状況を提供する。

【００６６】

Ｃｏｍｍａｎｄは本明細書で説明するように、パケットのコマンドを識別する。

【００６７】

ＣｒｅｄｉｔｓＧｒａｎｔｅｄ／ＣｒｅｄｉｔｓＲｅｑｕｅｓｔｅｄは、クライアントによって送信時により多くのクレジットを要求するために使用され、サーバによってその応答時に、新しいクレジット管理方式内でより多くのクレジットを許可するために使用される。

【００６８】

メッセージに関連するＦｌａｇには以下が含まれる。

【００６９】

【表５】

```
#define SMB2_FLAGS_SERVER_TO_REDIR          0x00000001
```

【００７０】

これが存在するとき、そのメッセージが要求に対する応答であることを示す。

【００７１】

【表６】

```
#define SMB2_FLAGS_ASYNC_COMMAND            0x00000002
```

【００７２】

応答時、サーバは、それを非同期的に処理していることを示すようにこのフラグを設定してＳＴＡＴＵＳ＿ＰＥＮＤＩＮＧを返す。

【００７３】

【表７】

```
#define SMB2_FLAGS_RELATED_OPERATIONS       0x00000004
```

【００７４】

複合メッセージのクライアントメッセージ送信時に、操作が関連しており、そのため、作成で開かれているファイルが後の操作でＦｉｌｅＩｄとして使用されることを示すために設定される。

【００７５】

【表８】

```
#define SMB2_FLAGS_SIGNED                   0x00000008
```

【００７６】

パケットが署名されているときに設定される。受信側は、その署名を検証する必要がある。署名に使用される鍵は、そのパケットを送信したセッションに基づくものである。

【００７７】

【表９】

```
#define SMB2_FLAGS_DFS_OPERATION            0x10000000
```

【００７８】

これはＤＦＳ操作である。サーバは、ＤＦＳに名前を変更（ｍｕｎｇｅ）させる必要がある。これは、作成オプションで置換され得る。

【００７９】

ＭｅｓｓａｇｅＩｄは、その応答と共に送られるメッセージを識別する。

【００８０】

ＰｒｏｃｅｓｓＩｄは、コマンドを発行するプロセスのクライアント側識別を記述する

。
【００８１】
ＳｅｓｓｉｏｎＩｄは、コマンドの確立されたセッション、またはセッションが使用されていない場合には０を識別する。
【００８２】
ＴｒｅｅＩｄは、コマンドのツリー接続、またはツリー接続が使用されていない場合には０を識別する。
【００８３】
ＡｓｙｎｃＩｄ：メッセージＩＤは実際には連番であり、利用可能な連番のウィンドウは、常に、右側にスライドするように設定される。極端に長期間実行されるコマンド（そのいずれもが無期限にブロックし得る、名前付きパイプ読取りまたは変更通知、あるいはｏｐｌｏｃｋ解除時に保留する作成など）は、ウィンドウがスライドする機能を停止させることがある。この問題に対処するために、サーバは、任意選択で、任意のコマンドにＳＴＡＴＵＳ＿ＰＥＮＤＩＮＧを用いて応答することができ、前述のＳＭＢ２＿ＦＬＡＧＳ＿ＡＳＹＮＣ＿ＣＯＭＭＡＮＤフラグを設定し、Ｓｅｓｓｉｏｎ／Ｔｒｅｅ／ＰｒｏｃｅｓｓＩｄの代わりに一意の識別子を提供する。これは、クライアントが、あたかも応答を受け取ったかのようにウィンドウをスライドし続けられることを意味する。その後のある時点に、本物の応答が、要求を満たすマッチングＡｓｙｎｃＩｄ（およびＣｏｍｍａｎｄＩｄ）と共にもたらされる。クライアントがそのようなコマンドを取り消そうとする場合には、クライアントは、フラグ設定およびマッチングＡｓｙｎｃＩｄと共に取消しを送る。
【００８４】
セキュリティ署名は、もはや隠れたインデックス番号が無いことを除けば、以前のプロトコルと同じである。インデックスは、必ずしもＭＩＤでの連番の使用を伴わない（これは、直接再現可能性を防止する）。これは、操作がトランスポートに向かう途中で順序付けられるよう強制せずに、セキュリティ署名の使用を可能にする。
【００８５】
ＮｅｘｔＣｏｍｍａｎｄは、このヘッダの先頭からメッセージ中の次のコマンドへのオフセットである。メッセージは、クワッド整列（ｑｕａｄ－ａｌｉｇｎ）される必要がある。ＳＭＢ２＿ＦＬＡＧＳ＿ＲＥＬＡＴＥＤ＿ＣＯＭＭＡＮＤの使用は、前述のように、複合のための様々な機能を可能にする。
【００８６】
コマンド形式
ＮＥＧＯＴＩＡＴＥ
前述ように、クライアントとサーバは、相互の対応能力を判定するのに役立つハンドシェークの一部として、折衝要求と応答を交換する。
【００８７】
形式
【００８８】

【表１０】

```
#define SMB2_NEGOTIATE_SIGNING_ENABLED          0x01
#define SMB2_NEGOTIATE_SIGNING_REQUIRED         0x02

#define SMB2_GLOBAL_CAP_DFS                     0x00000001
#define SMB2_GLOBAL_CAP_LWIO                    0x00000002
#define SMB2_GLOBAL_CAP_TXF                     0x00000004
#define SMB2_GLOBAL_CAP_CSE                     0x00000008

typedef struct _SMB2_REQ_NEGOTIATE
{
    USHORT StructureSize;               // = sizeof(SMB2_REQ_NEGOTIATE)
    USHORT Reserved;                    // = 0
    ULONG  DialectCount;                // Number of dialects we support
    USHORT Dialects[];                  // Array of dialect revision
                                        // numbers.
} SMB2_REQ_NEGOTIATE, *PSMB2_REQ_NEGOTIATE;

typedef struct _SMB2_RESP_NEGOTIATE
{
    USHORT StructureSize;               // = sizeof(SMB2_RESP_NEGOTIATE)
    USHORT SecurityMode;                // = signing flags

    USHORT DialectRevision;             // Server selected dialect from
                                        // received list
    USHORT Reserved;

    GUID   ServerGuid;                  // Server generated GUID

    ULONG  Capabilities;                // Global server capability flags
    ULONG  MaxTransactSize;             // Largest transact buffer we will
                                        // accept or send

    ULONG  MaxReadSize;                 // Largest size read supported
    ULONG  MaxWriteSize;                // Largest size write supported

    UINT64 SystemTime;                  // System (UTC) time on the server

    USHORT EncryptionKeyOffset;
    USHORT EncryptionKeyLength;

    USHORT SecurityBufferOffset;
    USHORT SecurityBufferLength;

    UCHAR  Buffer[1];
} SMB2_RESP_NEGOTIATE, *PSMB2_RESP_NEGOTIATE;
```

【００８９】

ＳＥＳＳＩＯＮ　ＳＥＴＵＰ

前述のように、Ｓｅｓｓｉｏｎ　Ｓｅｔｕｐ（セッションセットアップ）は、新しいセッションの認証プロセスを処理する。

【００９０】

形式

【００９１】

【表１１】

```
typedef struct _SMB2_REQ_SESSION_SETUP
{
    USHORT StructureSize;                   // = sizeof(SMB2_REQ_SESSION_SETUP)
    USHORT VcNumber;                        // 0 = first connection, nonzero =
                                            // additional

    ULONG  Capabilities;                    // Capabilities of the client.
    ULONG  Channel;                         // nonzero = binding new channel to
                                            // session

    USHORT SecurityBufferOffset;
    USHORT SecurityBufferLength;

    UCHAR  Buffer[1];                       // Security buffer
} SMB2_REQ_SESSION_SETUP, *PSMB2_REQ_SESSION_SETUP;

#define SMB2_SESSION_FLAG_IS_GUEST  0x0001
#define SMB2_SESSION_FLAG_IS_NULL   0x0002

typedef struct _SMB2_RESP_SESSION_SETUP
{
    USHORT StructureSize;                   // =
                                            // sizeof(SMB2_RESP_SESSION_SETUP)
    USHORT SessionFlags;

    USHORT SecurityBufferOffset;
    USHORT SecurityBufferLength;

    UCHAR  Buffer[1];                       // Security buffer
}
```

【００９２】

ＬＯＧＯＦＦ

既存のセッションをログオフする。

【００９３】

形式

【００９４】

【表１２】

```
typedef struct _SMB2_REQ_LOGOFF {
    USHORT StructureSize;
    USHORT Reserved;
} SMB2_REQ_LOGOFF;

typedef struct _SMB2_RESP_LOGOFF {
    USHORT StructureSize;
    USHORT Reserved;
} SMB2_RESP_LOGOFF;
```

【００９５】

このコマンドは、ヘッダで指定されるＳｅｓｓｉｏｎＩＤを持つセッションを切断する。開いているファイルは閉じられ、他の既存の構造（ツリー接続など）は切断される。所与のＳｅｓｓｉｏｎＩｄでは、それ以上の操作は処理され得ない。

【００９６】

ＴＲＥＥ　ＣＯＮＮＥＣＴ

サーバマシン上の共有リソースへのツリー接続を作成する。

【００９７】

形式

【００９８】

【表１３】

```
typedef struct _SMB2_REQ_TREE_CONNECT
{
    USHORT StructureSize;              // = sizeof(SMB2_REQ_TREE_CONNECT)
    USHORT Reserved;

    USHORT PathOffset;                 // Full path (i.e. \\SERVER\SHARE)
    USHORT PathLength;

    UCHAR  Buffer[1];
} SMB2_REQ_TREE_CONNECT, *PSMB2_REQ_TREE_CONNECT;

#define SMB2_SHAREFLAG_MANUAL_CACHING      0x00000000
#define SMB2_SHAREFLAG_AUTO_CACHING        0x00000001
#define SMB2_SHAREFLAG_VDO_CACHING         0x00000002
#define SMB2_SHAREFLAG_NO_CACHING          0x00000003
#define SMB2_SHAREFLAG_CACHING_FLAGS       0x00000003

// Reserved share cap 0x00000001
// Reserved share cap 0x00000002
// Reserved share cap 0x00000004
#define SMB2_SHARE_CAP_DFS                 0x00000008  // This is a DFS share

#define SMB2_SHARE_TYPE_DISK                   0x01
#define SMB2_SHARE_TYPE_PIPE                   0x02
typedef struct _SMB2_RESP_TREE_CONNECT
{
    USHORT StructureSize;              // = sizeof(SMB2_RESP_TREE_CONNECT)
    UCHAR ShareType;
    UCHAR Reserved;

    ULONG ShareFlags;
    ULONG  Capabilities;
} SMB2_RESP_TREE_CONNECT, *PSMB2_RESP_TREE_CONNECT;
```

【００９９】

クライアントは、ツリー接続を確立するためにサーバにこのコマンドを発行する。パスは、＼＼ｓｅｒｖｅｒ＼ｓｈａｒｅの形のものであり、バッファに記入される。サーバ名を包含することは、共有スコーピング（ｓｈａｒｅ　ｓｃｏｐｉｎｇ）のような機能を可能にする。

【０１００】

サーバからの正常な応答時に、クライアントは、ヘッダでＳｈａｒｅＦｌａｇｓおよびＳｈａｒｅＣａｐａｂｉｌｉｔｉｅｓと共にＴｒｅｅＩｄを受け取る。現在、共有フラグは、クライアントに、共有用のＣＳＣキャッシュ特性が何であるかを示すが、後でさらに多くが付加され得る。対応能力は、クライアントに、その共有を支持するファイルシステムが、ファイルレベルのセキュリティ、タイムワープ、ＴｘＦ（トランザクションファイルシステム）、またはクライアント側の暗号化をサポートするかどうか知らせる。ファイルシステムが、これらの特性を、全部ではなく一部のサブツリー上でサポートしている場合（マウントポイントの場合など）、ファイルシステムはこれらの特性をサポートしていると返し、それが許容されない場合には、単に、これらの特性の使用を求める個々の要求を失敗させるはずである。クライアントは、それが理解しないどんなフラグも対応能力も無視するはずである。

【０１０１】

ＴＲＥＥ　ＤＩＳＣＯＮＮＥＣＴ

既存のＴｒｅｅＣｏｎｎｅｃｔを切断する。

【０１０２】

形式

【０１０３】

【表１４】

```
typedef struct _SMB2_REQ_TREE_DISCONNECT {
    USHORT StructureSize;
    USHORT Reserved;
} SMB2_REQ_TREE_DISCONNECT;

typedef struct _SMB2_REQ_TREE_DISCONNECT {
    USHORT StructureSize;
    USHORT Reserved;
} SMB2_REQ_TREE_DISCONNECT;
```

【０１０４】

コマンドが処理された後は、所与のＴｒｅｅＩｄに関してそれ以上の操作は正常に完了され得ない。ＴｒｅｅＩｄは、ヘッダから取られる。

【０１０５】

ＣＲＥＡＴＥ

ファイル、プリンタ、またはパイプを開く。

【０１０６】

形式

【０１０７】

【表１５】

```
#define SMB2_OPLOCK_LEVEL_NONE       0
#define SMB2_OPLOCK_LEVEL_II         1
#define SMB2_OPLOCK_LEVEL_EXCLUSIVE  8
#define SMB2_OPLOCK_LEVEL_BATCH      9
#define SMB2_OPLOCK_LEVEL_DIRCHANGE  16

typedef struct _SMB2_REQ_CREATE
{
    USHORT StructureSize;               // = sizeof(SMB2_REQ_CREATE)
    UCHAR  SecurityFlags;               // QOS security flags
    UCHAR  RequestedOplockLevel;        // Desired oplock level

    ULONG       ImpersonationLevel;     // QOS security info

    UINT64 SmbCreateFlags;
    UINT64 RootDirectoryFid;            // For relative opens

    ACCESS_MASK DesiredAccess;
    ULONG       FileAttributes;
    ULONG       ShareAccess;
    ULONG       CreateDisposition;
    ULONG       CreateOptions;

    USHORT      NameOffset;             // Name relative to share
    USHORT      NameLength;

    ULONG       CreateContextsOffset;   // Extra create parameters
    ULONG       CreateContextsLength;

    UCHAR       Buffer[1];              // Name[], CreateContexts[]
} SMB2_REQ_CREATE, *PSMB2_REQ_CREATE;

typedef struct _SMB2_CREATE_CONTEXT
{
    ULONG       Next;
    USHORT      NameOffset;
    USHORT      NameSize;
    USHORT      Reserved;
    USHORT      DataOffset;
```

【０１０８】

【表１６】

```
    ULONG       DataSize;
    UCHAR       Buffer[1];  // Name[], Data[]
} SMB2_CREATE_CONTEXT, *PSMB2_CREATE_CONTEXT;

typedef struct _SMB2_RESP_CREATE
{
    USHORT StructureSize;               // = sizeof(SMB2_RESP_CREATE)
    UCHAR  OplockLevel;                 // The oplock granted on the file
    UCHAR  Reserved;

    ULONG  CreateAction;                // Action taken by create

    UINT64 FileId;                      // ID for this open

    UINT64 CreationTime;                // File time information
    UINT64 LastAccessTime;
    UINT64 LastWriteTime;
    UINT64 LastChangeTime;

    UINT64 AllocationSize;              // File size information
    UINT64 EndOfFile;

    ULONG  FileAttributes;              // NT attributes of the file

    ULONG Reserved2;                    // For 8-byte alignment

    ULONG CreateContextsOffset;         // Responses for Extra Create
                                        // Parameters
    ULONG CreateContextsLength;

    UCHAR Buffer[1];                    // CreateContexts[]
} SMB2_RESP_CREATE, *PSMB2_RESP_CREATE;

#define SMB2_CREATE_EA_BUFFER               (ULONG)('AtxE')
#define SMB2_CREATE_SD_BUFFER               (ULONG)('DceS')
#define SMB2_CREATE_MARSHALLED_TRANSACTION  (ULONG)('xTrM')
#define SMB2_CREATE_MINIVERSION             (ULONG)('rVnM')
#define SMB2_CREATE_VERSION                 (ULONG)('sreV')
#define SMB2_CREATE_NTFS_FID                (ULONG)('diFN')
#define SMB2_CREATE_TIMEWARP_TOKEN          (ULONG)('prWT')
#define SMB2_CREATE_EFS_STREAM              (ULONG)('sfE$')
#define SMB2_CREATE_CLIENT_SIDE_ENCRYPTION  (ULONG)('lESC')
```

【０１０９】

作成要求は、従来の明確な属性以外の様々な属性を持つファイルの作成を可能にするよう求める可変長要求である。（拡張された属性が存在しない）標準的な場合は簡単明瞭である。すなわち、クライアントは、ＲｏｏｔＤｉｒｅｃｔｏｒｙＦｉｄ（必要に応じて相対オープンのため）、ＤｅｓｉｒｅｄＡｃｃｅｓｓ、ＦｉｌｅＡｔｔｒｉｂｕｔｅｓ、ＳｈａｒｅＡｃｃｅｓｓ、ＣｒｅａｔｅＤｉｓｐｏｓｉｔｉｏｎ、およびＣｒｅａｔｅＯｐｔｉｏｎｓに記入する。これらは、所望のｏｐｌｏｃｋレベルを設定し、サービス品質のためのＳｅｃｕｒｉｔｙＦｌａｇｓおよびＩｍｐｅｒｓｏｎａｔｉｏｎ（偽装）レベルに記入する。現在、ＳｍｂＣｒｅａｔｅＦｌａｇｓは定義されていないが、その使用ためのスペースが割り振られている。クライアントはこのパケットをサーバに送り、サーバはファイルを開いて、失敗コードを返し、または、ファイルを識別するＦｉｌｅＩｄ、Ｃｒｅａｔｉｏｎ／ＬａｓｔＡｃｃｅｓｓ／ＬａｓｔＷｒｉｔｅ／ＬａｓｔＣｈａｎｇｅＴｉｍｅ、ＡｌｌｏｃａｔｉｏｎＳｉｚｅおよびＥｎｄＯｆＦｉｌｅ情報、ならびにＦｉｌｅＡｔｔｒｉｂｕｔｅｓと共にＳｕｃｃｅｓｓ（成功）を返す。

【０１１０】

それは、現在のプロトコルが行うのとほぼ同じように動作する通常の場合である。より高度な場合について、ユーザが拡張された属性（ＥＡ）を持つファイルを作成しようとしていると考える。以前のプロトコルでは、Ｔｒａｎｓａｃｔ（トランザクション）呼び出

しを介した、これを処理する全く異なるやり方があった。今や、クライアントは、通常どおり作成要求を組み立て、しかも、その作成要求の末尾にＣｒｅａｔｅＣｏｎｔｅｘｔを付加することもできる。その要求は、「ＥｘｔＡ」という名前を持ち、データはファイルに関して設定するＥＡを含むはずである。サーバは、これを受け取ると、ＥＡデータを構文解析し、それを作成と共に発行するはずである。また、作成コンテキストは、作成応答時に追加情報を提供するためにも返され得る。最初の反復では、名前は、長さ４のものになり、そのため、それらをｌｏｎｇ型としてフォーマットし、それらをオンにすることができる。現在のＣｒｅａｔｅＣｏｎｔｅｘｔのリストは以下の通りである。

【０１１１】

１）「ＥｘｔＡ」－データは、作成ファイルに持たせる拡張された属性を含む。

【０１１２】

２）「ＳｅｃＤ」－データは、作成ファイルに持たせる自己相対セキュリティ記述子を含む。

【０１１３】

３）「ＴＷｒｐ」－データは、開くファイルを検索するのに使用されるべきタイムワープタイムスタンプを含む。タイムスタンプはシステム時刻形式である。

【０１１４】

４）「ＭｒＴｘ」－データは、トランザクションでファイルを開くときに使用されるマーシャリングされたトランザクションを含む。

【０１１５】

５）「ＭｎＶｒ」－データは、処理ファイルを開くためのミニバージョン番号（ＵＬＯＮＧ）を含む。

【０１１６】

６）「Ｖｅｒｓ」－データは、開かれたファイルのバージョン番号（ＵＬＯＮＧ）を含む（作成応答）。

【０１１７】

７）「ＮＦｉｄ」－データは、開かれたファイルのＮＴＦＳ　Ｆｉｄ（ＬＡＲＧＥ＿ＩＮＴＥＧＥＲ）を含む（作成応答）。

【０１１８】

８）「＄Ｅｆｓ」－データは、新しい暗号化ファイルにスタンプされる＄ＥＦＳストリームを含む。

【０１１９】

９）「ＣＳＥ１」－データは、開かれた暗号化ファイルの＄ＥＦＳストリームを含む（作成応答）。

サーバがサポートする際により多くのＣｒｅａｔｅＣｏｎｔｅｘｔ値が追加され得る。（クライアントが、作成要求を発行する前に、サーバがどのタグをサポートしているかわかるように、値が追加される際、それらの値は、それらに関連付けられた対応能力ビットを持ち、または新しいダイアレクト改訂版に関連付けられるはずである。）未認識コンテキストタグを持つ作成要求を受け取るサーバは、その要求を失敗させるはずである。

【０１２０】

ＣＬＯＳＥ

クライアントは、以前に開かれたファイルのインスタンスを閉じるためにＣｌｏｓｅ（クローズ）を送る。クローズが処理されると、以前のＦＩＤに関してどんな操作も許容されない。

【０１２１】

形式

【０１２２】

【表１７】

```
typedef struct _SMB2_REQ_CLOSE {
    USHORT      StructureSize;
    USHORT      Reserved;
    ULONG       Flags;

    UINT64      FileId;
    UINT64      LastWriteTime;
} SMB2_REQ_CLOSE, *PSMB2_REQ_CLOSE;

typedef struct _SMB2_RESP_CLOSE {
    USHORT      StructureSize;
    USHORT      Reserved;
} SMB2_RESP_CLOSE, *PSMB2_RESP_CLOSE;
```

【０１２３】

クローズコマンドで、クライアントは、閉じられるファイルのＦｉｌｅＩｄを、（ＳｙｓｔｅｍＴｉｍｅ形式の）ＬａｓｔＷｒｉｔｅＴｉｍｅと共に指定する。これは、クライアントが、ファイルにキャッシュ書込みが行われた最後の時刻をファイルに対する最後の書込み時刻として設定することを可能にする。また、クライアントは、ＬａｓｔＷｒｉｔｅＴｉｍｅを指定しないことを示すために、ＬａｓｔＷｒｉｔｅＴｉｍｅについてゼロを送ることもできる。また、この構造は、現在は未定義であるが、後の時点において使用され得るＣｌｏｓｅフラグのための場所も割り当てる。

【０１２４】

ＦＬＵＳＨ

フラッシュコマンドは、サーバに、所与のファイルに関するすべてのキャッシュデータをフラッシュするよう知らせる。

【０１２５】

形式

【０１２６】

【表１８】

```
typedef struct _SMB2_REQ_FLUSH {
    USHORT StructureSize;
    USHORT Reserved1;
    ULONG  Reserved2;

    UINT64 FileId;

} SMB2_REQ_FLUSH, *PSMB2_REQ_FLUSH;

typedef struct _SMB2_RESP_FLUSH {
    USHORT StructureSize;
    USHORT Reserved;
}
```

【０１２７】

サーバからの正常な応答時に、クライアントは、すべてのキャッシュデータがその永続的補助記憶にフラッシュされていることを保証される。クライアントは、フラッシュしようとするファイルのＦｉｌｅＩｄを指定する。パイプのフラッシュは、そのパイプからすべてのデータが消費されるまで戻らず、それにはしばらく時間がかかることがある。

【０１２８】

ＲＥＡＤ

開いたファイルからデータを読み取る。

【０１２９】

形式

【０１３０】

【表１９】

```
typedef struct _SMB2_REQ_READ
{
    USHORT StructureSize;           // = sizeof(SMB2_REQ_READ)
    UCHAR  Padding;                 // Requested padding of read data
                                    // response from beginning of header
    UCHAR  Reserved;

    ULONG  Length;                  // The length of the read to send on
                                    // this channel
    UINT64 Offset;                  // Offset of the read
    UINT64 FileId;                  // Identifier of the file
// being read

    ULONG  MinimumCount;            // The minimum bytes to read and
                                    // consider success

    ULONG  Channel;                 // The channel to send the remaining
                                    // data on
    ULONG  RemainingBytes;        // If channel != 0, additional bytes to
                                  // be read and sent on channel,
                                  // otherwise how much more planned to read

    USHORT ReadChannelInfoOffset;   // If channel != 0, information about
    USHORT ReadChannelInfoLength;   // channel to send additional data on.

    UCHAR  Buffer[1];               // ReadChannelInfo
} SMB2_REQ_READ, *PSMB2_REQ_READ;

typedef struct _SMB2_RESP_READ
{
    USHORT StructureSize;           // = sizeof(SMB2_RESP_READ)
    UCHAR  DataOffset;              // Offset to data in the packet
    UCHAR  Reserved;

    ULONG  DataLength;              // Length of data returns as part of
                                    // this packet
    ULONG  DataRemaining;           // Length of data that is being sent on
                                    // the alternate channel if specified,
                                    // otherwise how much more we are ready
                                    // to have read

    UCHAR  Buffer[1];             // Pad[], Data[]
} SMB2_RESP_READ, *PSMB2_RESP_READ;
```

【０１３１】

Ｒｅａｄ（読取り）は非常に分かりやすい。クライアントがファイル（ＦｉｌｅＩｄによる）、オフセット、および読取りの長さを指定し、サーバがデータを返す。クライアントが指定し得ることが他にいくつかある。ＭｉｎＣｏｕｎｔは、サーバに、正常な戻りのためにそれがファイルから読み取る最小量を知らせる。読取りがその量に達しない場合、サーバは、単に、データバッファ全体を返すのではなく失敗を返す。また、クライアントは、より適切な処理のためのＰａｄｄｉｎｇ（パッディング）を推奨することもできる。これは、サーバがデータを入れるべき読取り応答パケットへのオフセットである。これは、クライアントが、情報をトランスポートから外れて受け取るときに、より効率のいいやり方で読取り応答バッファを設定することを可能にする。残りのフィールドは、サーバに、これがその読取りの一部にすぎない場合、読取り全体がどれほどになるか示す。ゆえに、クライアントが１ｋチャンク単位で８ｋを読み取ることになる場合、クライアントは、Ｒｅｍａｉｎｉｎｇ（残り）＝７ｋとして１ｋの読取りを発行するはずである。これは、サーバに、８ｋ全部を１回の操作で読み取り、そのデータをクライアントに戻してバッファすることにより最適化するオプションを可能にする。

【０１３２】

サーバは、サーバ応答時に、読取りコマンドで指定されたＤａｔａＲｅｍａｉｎｉｎｇ

と共に（ＤａｔａＬｅｎｇｔｈフィールドで）どれほどのデータを返すか示す。

【０１３３】

コマンドで指定されたチャネルが、そのコマンドがもたらされたチャネルでない場合、ユーザは、チャネル読取りを求めている。これは、チャネル０上で、「ｃｈａｎｎｅｌ＝１、Ｌｅｎｇｔｈ＝０、Ｒｅｍａｉｎｉｎｇ＝６４ｋ」として読取りを要求する場合、サーバは、「ＤａｔａＬｅｎｇｔｈ＝０、ＤａｔａＲｅｍａｉｎｉｎｇ＝６４ｋ」で応答し、チャネル１を介してもたらされる次の６４ｋバイトがそのデータであることを意味する。クライアントは、このコマンドが発行されるときにチャネル１上でどんなデータ応答も未処理にならないようにこれを同期させる役割を果たす。また、クライアントは、応答が先頭の１ｋのデータを含み、データの残り（後の７ｋ）がチャネル１を介してストリーミングされるように、（チャネル０上で）「ｒｅａｄ　Ｃｈａｎｎｅｌ＝１、ＤａｔａＬｅｎｇｔｈ＝１ｋ、Ｒｅｍａｉｎｉｎｇ＝７ｋ」を発行することもできる。

【０１３４】

ＷＲＩＴＥ

開いたファイルにデータを書き込む。

【０１３５】

形式

【０１３６】

【表２０】

```
typedef struct _SMB2_REQ_WRITE
{
    USHORT StructureSize;              // = sizeof(SMB2_REQ_WRITE)
    USHORT DataOffset;                 // Offset to data from header

    ULONG  Length;                     // Length of data being written
    UINT64 Offset;                     // File offset of the write
    UINT64 FileId;                     // Identifier for the file being
                                       // written to

    ULONG  Channel;                    // If non-zero, the channel where
                                       // the remaining data should be sent
    ULONG  Remaining;                   // Number of bytes to be sent on
channel

    USHORT WriteChannelInfoOffset;     // If channel != 0, information
                                       // about the channel
    USHORT WriteChannelInfoLength;     // we wish to write data to.

    ULONG  Flags;

    UCHAR Buffer[1];                   // WriteChannelInfo
} SMB2_REQ_WRITE, *PSMB2_REQ_WRITE;

typedef struct _SMB2_RESP_WRITE
{
    USHORT StructureSize;              // = sizeof(SMB2_RESP_WRITE)
    USHORT Reserved;

    ULONG  Count;                      // How much of the data was written

    ULONG  Remaining;                  // How many bytes we can receive on
                                       // the channel
    USHORT WriteChannelInfoOffset;     // If channel != 0, optional
                                       // information about the channel we
    USHORT WriteChannelInfoLength;     // wish to write data to.

    UCHAR  Buffer[1];                   // WriteChannelInfo
} SMB2_RESP_WRITE, *PSMB2_RESP_WRITE;
```

【０１３７】

クライアントは、（ＦｉｌｅＩｄで識別される）ファイル、オフセット、書込みの長さ

を記入し、データを添付する。サーバ性能を助けるために、データが最初の折衝応答で返されるときパッディングされることが推奨される。また、クライアントは、サーバを最適化させるために、どれほど追加のデータをサーバに書き込むか示すこともできる。応答時に、サーバは、どれほどが書き込まれたか示し、さらに予期している量を返す。

【０１３８】

書込みで指定されたチャネルが、コマンドがもたらされたチャネルでない場合、クライアントは、別のチャネルでデータをストリーミングするよう求めている。一例が、チャネル０上で、「Ｃｈａｎｎｅｌ＝１、Ｌｅｎｇｔｈ＝０、Ｒｅｍａｉｎｉｎｇ＝６４ｋ」と共に受け取られる書込みであろう。クライアントは、チャネル１で６４ｋの書込みをストリーミングするよう求めている。サーバは、この書込みを可能にするために、「Ｃｏｕｎｔ＝０、Ｒｅｍａｉｎｉｎｇ＝６４ｋ」として応答するはずである。応答は、このチャネル上でデータが送られ、確認された後でもたらされる第２の応答のＡｓｙｎｃＩｄを含む。その場合、チャネル１上でストリーミングされる次の６４ｋバイトがそのデータになるはずである（ヘッダなし）。完了時に、サーバは、操作の成功／失敗を示すためにチャネル０上でＳＭＢ２＿ＲＥＳＰ＿ＷＲＩＴＥを送り、ＡｓｙｎｃＩｄ情報を使って第２の応答を送る。ただし、特定のチャネルが固有の肯定応答を可能にする場合はこの限りではなく、それは、そのチャネル自体で行われる。

【０１３９】

ＢＲＥＡＫ＿ＯＰＬＯＣＫ

ファイルに対して講じられる便宜的ロックの解除を要求し、確認するのに使用される。

【０１４０】

形式

【０１４１】

【表２１】

```
typedef struct _SMB2_REQ_BREAK_OPLOCK
{
    USHORT StructureSize;          // = sizeof(SMB2_REQ_BREAK_OPLOCK)
    UCHAR  OplockLevel;            // Level to break to.  (Level2 or None)
    UCHAR  Reserved;

    ULONG  Reserved2;             // Timeout in seconds

    UINT64 FileId;           // Identifier of the file being locked/unlocked
} SMB2_REQ_BREAK_OPLOCK, *PSMB2_REQ_BREAK_OPLOCK;

typedef struct _SMB2_RESP_BREAK_OPLOCK
{
    USHORT StructureSize;            // = sizeof(SMB2_RESP_LOCK)
    UCHAR  OplockLevel;             // Level broken to  (<= level requested)
    UCHAR  Reserved;

    ULONG Reserved2;
    UINT64 FileId;
} SMB2_RESP_BREAK_OPLOCK, *PSMB2_RESP_BREAK_OPLOCK;
```

【０１４２】

別のユーザが、既存のロックの解除を必要とするやり方で、クライアントが便宜的ロックを保持しているファイルへのアクセスを要求すると、ＳＲＶは、そのクライアントに、ＳＭＢ２＿ＲＥＳＰ＿ＢＲＥＡＫ＿ＯＰＬＯＣＫを送る。次いで、クライアントは、そのｏｐｌｏｃｋを解除するために所与のファイルについてＲＥＱ＿ＢＲＥＡＫ＿ＯＰＬＯＣＫを送るはずであり、ＳＲＶは、この場合もやはり、それに応答してこれを確認する。

【０１４３】

ＬＯＣＫ

バイト範囲ロックを要求するのに使用され、便宜的ロックを要求する（また、ロックが解除されたときにクライアントに知らせる）のにも使用される。

【０１４４】

形式

【０１４５】

【表２２】

```
#define SMB2_LOCKFLAG_SHARED_LOCK        0x01
#define SMB2_LOCKFLAG_EXCLUSIVE_LOCK     0x02
#define SMB2_LOCKFLAG_UNLOCK             0x04
#define SMB2_LOCKFLAG_FAIL_IMMEDIATELY   0x10

typedef struct _SMB2_LOCK
{
    UINT64 Offset;
    UINT64 Length;
    ULONG Flags;
    ULONG Reserved;
} SMB2_LOCK, *PSMB2_LOCK;

typedef struct _SMB2_REQ_LOCK
{
    USHORT StructureSize;                // = sizeof(SMB2_REQ_LOCK)
    UCHAR  LockCount;

    ULONG  Reserved;

    UINT64 FileId;                       // Identifier of the file being
                                         // locked/unlocked

    SMB2_LOCK Locks[1];                  // Array of size (LockCount)
} SMB2_REQ_LOCK, *PSMB2_REQ_LOCK;

typedef struct _SMB2_RESP_LOCK
{
    USHORT StructureSize;                // = sizeof(SMB2_RESP_LOCK)
    USHORT Reserved;
} SMB2_RESP_LOCK, *PSMB2_RESP_LOCK;
```

【０１４６】

Ｌｏｃｋ（ロック）要求の構文は、ＳＭＢ１ロック要求に類似している。クライアントは、ＦｉｌｅＩｄと、ロックしようとするオフセットおよび長さを示す１つまたは複数のＳＭＢ＿ＬＯＣＫ構造を指定する。これらのロック構造のすべてはロックまたはアンロックでなければならない。しかしながら、単一のバッチロック操作（ｂａｔｃｈｅｄ　ｌｏｃｋ　ｏｐｅｒａｔｉｏｎ）において、共有された排他的ロック要求を混在させることができる。ロックバッチ処理（ｌｏｃｋ　ｂａｔｃｈｉｎｇ）の最も一般的な用途は、バッチｏｐｌｏｃｋ解除の一部として一連のロックを請求することであり、それらロックすべてが成功することが保証されているときに最も有用である。

【０１４７】

正常な戻りは、クライアントが要求されたバイト範囲ロックを獲得した（または解除した）ことをクライアントに示す。失敗の場合には、バイト範囲ロックは許可されていない。

【０１４８】

ＥＣＨＯ

Ｅｃｈｏは、クライアントによって、サーバが、所与の時点において依然として稼動しているかどうか判定するのに使用される。このコマンドを受け取ると、サーバは、単に、それを反転させ、成功を返す。

【０１４９】

形式

【０１５０】

【表２３】

```
typedef struct _SMB2_REQ_ECHO {
    USHORT StructureSize;
    USHORT Reserved;
} SMB2_REQ_ECHO, *PSMB2_REQ_ECHO;

typedef struct _SMB2_RESP_ECHO {
    USHORT StructureSize;
    USHORT Reserved;
} SMB2_RESP_ECHO, *PSMB2_RESP_ECHO;
```

【０１５１】

サーバは、パケットに応答して、適正に動作していることを示す。クライアントにサーバを「ｐｉｎｇ」させるのに使用される。

【０１５２】

ＣＡＮＣＥＬ

クライアントによって、送信済み操作の取消しを要求するのに使用される。

【０１５３】

形式

【０１５４】

【表２４】

```
typedef struct _SMB2_REQ_CANCEL {
    USHORT StructureSize;
    USHORT Reserved;
} SMB2_REQ_CANCEL, *PSMB2_REQ_CANCEL;
```

【０１５５】

Ｃａｎｃｅｌ（取消し）に応答はないが、結果として、コマンド自体が正常に完了され、またはＳＴＡＴＵＳ＿ＣＡＮＣＥＬＬＥＤで失敗することになるはずであり、これは可能な限り早く行われるはずである。送られる操作は、それが取消しコマンドのＭｅｓｓａｇｅＩｄを共有することになるため識別される。これは、サーバに送信されたＭｅｓｓａｇｅＩｄがすでに以前に使用されている場合の一事例である。応答がＡｓｙｎｃＩｄと共にもたらされた場合、それはヘッダにあるはずであり、サーバ側でコマンドを探し出すのに使用される。

【０１５６】

ＩＯＣＴＬ

ＩＯＣＴＬは、ネットワークを介してＤｅｖｉｃｅ　Ｃｏｎｔｒｏｌ（デバイス制御）またはＦｉｌｅ　Ｓｙｓｔｅｍ　Ｃｏｎｔｒｏｌ（ファイルシステム制御）コマンドを発行するのに使用される。

【０１５７】

形式

【０１５８】

【表２５】

```
// Request
typedef struct _SMB2_REQ_IOCTL
{
    USHORT StructureSize;              // = sizeof(SMB2_REQ_TRANSACT)
    USHORT Reserved;

    ULONG  CtlCode;

    UINT64 FileId;

    ULONG  InputOffset;                // Bytes for input buffer
    ULONG  InputCount;                 // Count of parameter bytes in this
                                       // message
    ULONG  MaxInputResponse;           // Max bytes server can return for
                                       // response parameters

    ULONG  OutputOffset;               // Data bytes location
    ULONG  OutputCount;                // Count of data bytes in this
                                       // message
    ULONG  MaxOutputResponse;          // Max bytes server can return for
                                       // response data

    ULONG  Flags;
    ULONG  Reserved2;

    UCHAR  Buffer[1];                  // Parameters[], Data[]
} SMB2_REQ_IOCTL, *PSMB2_REQ_IOCTL;

// Response
typedef struct _SMB2_RESP_IOCTL
{
    USHORT StructureSize;              // = sizeof(SMB2_RESP_TRANSACT)
    USHORT Reserved;

    ULONG  CtlCode;

    UINT64 FileId;

    ULONG  InputOffset;                // Bytes for input buffer
    ULONG  InputCount;                 // Count of parameter bytes in this
                                       // message

    ULONG  OutputOffset;               // Data bytes location
    ULONG  OutputCount;                // Count of data bytes in this
                                       // message

    ULONG  Flags;
    ULONG  Reserved2;

    UCHAR  Buffer[1];                  // Parameters[], Data[]
} SMB2_RESP_IOCTL, *PSMB2_RESP_IOCTL;} SMB2_RESP_TRANSACT,
*PSMB2_RESP_TRANSACT;
```

【０１５９】

ＩＯＣＴＬは、ネットワークを介して汎用のファイルシステムまたはデバイス制御コマンドを発行するのに使用される。これは、制御コードのＭＥＴＨＯＤに基づいて入力および出力バッファをパックし、ネットワークを介してそれらを送る。次いで、サーバ側は、それらを再パッケージし、ファイルオブジェクトに対するＦＳＣＴＬ／ＩＯＣＴＬを発行する。その結果は同様にパックされ、ステータスコードと共にユーザに返される。許容可能なＦＳＣＴＬ／ＩＯＣＴＬコードのセットは、ＳＲＶによっても、基礎をなすファイルシステムによっても制限され得る（必ずしもすべてがリモートで有効であるとは限らない）。

【０１６０】

バッファされた、また直接の要求では、要求時には入力だけが有効であり、出力は応答

時に送られる。どちらの要求でも、入力も出力も両方向に送られることはない。

【０１６１】

ＱＵＥＲＹ　ＤＩＲＥＣＴＯＲＹ

クライアントがネットワークを介して開いたディレクトリハンドル上でディレクトリ列挙を問い合わせることを可能にする。

【０１６２】

形式

【０１６３】

【表２６】

```
//
// QUERY_DIRECTORY Command
//
#define SMB2_REOPEN     0x10

// Request
typedef struct _SMB2_REQ_QUERY_DIRECTORY
{
    USHORT StructureSize;               // =
                                        // sizeof(SMB2_REQ_QUERY_DIRECTORY)
    UCHAR  FileInformationClass;
    UCHAR  Flags;                       // SL_/SMB2_ flags
    ULONG  FileIndex;

    UINT64 FileId;

    USHORT FileNameOffset;
    USHORT FileNameLength;
    ULONG  OutputBufferLength;

    UCHAR  Buffer[1];                   // FileName parameter
} SMB2_REQ_QUERY_DIRECTORY, *PSMB2_REQ_QUERY_DIRECTORY;

// Response
typedef struct _SMB2_RESP_QUERY_DIRECTORY
{
    USHORT StructureSize;               // =
sizeof(SMB2_RESP_QUERY_DIRECTORY)
    USHORT OutputBufferOffset;
    ULONG  OutputBufferLength;

    UCHAR  Buffer[1];                   // Response data
} SMB2_RESP_QUERY_DIRECTORY, *PSMB2_RESP_QUERY_DIRECTORY;
```

【０１６４】

ＱｕｅｒｙＤｉｒｅｃｔｏｒｙ呼び出しは、既存のＮＴセマンティクスに非常にマッチする。呼び出し側は、ＩｎｆｏＣｌａｓｓ、開いているディレクトリのＦｉｌｅＩｄ、（ワイルドカード／ファイルサーチパラメータまたは既存のサーチでの再開名（ｒｅｓｕｍｅ　ｎａｍｅ）を指定する）ファイル名部分、および呼び出しに関連付けられた任意の有効なＳＬ＿フラグを提供し、ＳＲＶは最大でＯｕｔｐｕｔＢｕｆｆｅｒＬｅｎｇｔｈまでのバッファを返す。

【０１６５】

また、ＱｕｅｒｙＤｉｒｅｃｔｏｒｙフラグ構造に含まれ得る新しいフラグ（ＳＭＢ２＿ＲＥＯＰＥＮ）もある。このフラグは、ＳＬ＿ＲＥＳＴＡＲＴ＿ＳＣＡＮフラグのより強力なバージョンである。後者は、指定されたサーチが変化していない場合にスキャンを再開することのみを許容する（すなわち、＊．＊またはｔ＊サーチを再開する）。前者は、サーバに、指定されたサーチが変化している場合にスキャンを再開するよう命じる。このフラグを使用するために、呼び出し側は、呼び出し全体にわたる排他的使用、および（変更通知などの）未処理の操作がないことを保証しなければならない。サーバは、この操作を実行するための適切な手段を講じ、それにはサーバ側で基礎をなすディレクトリハン

ドルを閉じ、再開することが関与し得る。これは、クライアントからは見えない。

【０１６６】

ＣＨＡＮＧＥ　ＮＯＴＩＦＹ

この潜在的に長期間続く操作は、クライアントがディレクトリに関する変更通知のために登録することを可能にする。

【０１６７】

形式

【０１６８】

【表２７】

```
//
// CHANGE_NOTIFY Command
//

// Request
typedef struct _SMB2_REQ_CHANGE_NOTIFY
{
    USHORT StructureSize;               // =
                                        // sizeof(SMB2_REQ_CHANGE_NOTIFY)
    USHORT Flags;                       // SL_WATCH_TREE?
    ULONG  OutputBufferLength;

    UINT64 FileId;

    ULONG  CompletionFilter;
    ULONG  Reserved;
} SMB2_REQ_CHANGE_NOTIFY, *PSMB2_REQ_CHANGE_NOTIFY;

// Response
typedef struct _SMB2_RESP_CHANGE_NOTIFY
{
    USHORT StructureSize;              // =
                                       // sizeof(SMB2_RESP_CHANGE_NOTIFY)
    USHORT OutputBufferOffset;
    ULONG  OutputBufferLength;

    UCHAR  Buffer[1];                   // Notify data
} SMB2_RESP_CHANGE_NOTIFY, *PSMB2_RESP_CHANGE_NOTIFY;
```

【０１６９】

呼び出し側は、呼び出し側がどの変更を対象とするか指定するＣｏｍｐｌｅｔｉｏｎＦｉｌｔｅｒと共にディレクトリのＦｉｌｅＩｄを送る。また、呼び出し側は、再帰的通知操作を示すＳＬ＿ＷＡＴＣＨ＿ＴＲＥＥフラグも送ることができる。この操作は、無限の期間保留することができるため、ほとんどの場合「ａｓｙｎｃ」挙動を呼び出す。また、同じハンドルに関するこれ以上のどんな変更も、ローカルファイルシステム挙動の場合と同様に、最初の変更が完了するのを待って保留することにも留意されたい。

【０１７０】

ＱＵＥＲＹ　ＩＮＦＯ

クライアントがリモートシステムからの情報を問い合わせることを可能にする。現在、これは、ファイル情報、ファイルシステム情報、セキュリティ情報、またはクォータ情報（ｑｕｏｔａ　ｉｎｆｏｒｍａｔｉｏｎ）を問い合わせるのに使用され得る。

【０１７１】

形式

【０１７２】

【表２８】

```
//
// QUERY_INFO
//
#define SMB2_0_INFO_FILE            0x01
#define SMB2_0_INFO_FILESYSTEM      0x02
#define SMB2_0_INFO_SECURITY        0x03
#define SMB2_0_INFO_QUOTA           0x04

typedef struct _SMB2_QUERY_QUOTA_INFO {
    UCHAR ReturnSingleEntry;        // Indicates that only a single entry
                                    // should be returned rather than
                                    // filling the buffer with as
                                    // many entries as possible.
    UCHAR RestartScan;              // Indicates whether the scan of the
                                    // quota information is to be
                                    // restarted from the beginning.
    USHORT Reserved;
    ULONG SidListLength;            // Supplies the length of the SID
                                    // list if present
    ULONG StartSidLength;           // Supplies optional SID that
                                    // indicates the returned information
                                    // is to start with an entry other
                                    // than the first.  This parameter is
                                    // ignored if a SidList is given
    ULONG StartSidOffset;           // Supplies the offset of Start Sid
                                    // in the buffer
} SMB2_QUERY_QUOTA_INFO, *PSMB2_QUERY_QUOTA_INFO;

// Request
typedef struct _SMB2_REQ_QUERY_INFO
{
    USHORT StructureSize;               // = sizeof(SMB2_REQ_QUERY_INFO)
    UCHAR  InfoType;                    // Determines info type
                                        //(SMB2_0_INFO_*)
    UCHAR  FileInfoClass;
    ULONG  OutputBufferLength;

    USHORT InputBufferOffset;           // Input buffer only valid on
                                        // Quota calls
    USHORT Reserved;
    ULONG  InputBufferLength;

    union {
        ULONG  SecurityInformation;     // For Query Security calls
        ULONG  EaIndex;                 // For QueryEA calls
    };
    ULONG  Flags;

    UINT64 FileId;

    UCHAR Buffer[1];
} SMB2_REQ_QUERY_INFO, *PSMB2_REQ_QUERY_INFO;

// Response
typedef struct _SMB2_RESP_QUERY_INFO
{
    USHORT StructureSize;               // = sizeof(SMB2_RESP_QUERY_INFO)
    USHORT OutputBufferOffset;
    ULONG  OutputBufferLength;

    UCHAR  Buffer[1];                   // File Info
} SMB2_RESP_QUERY_INFO, *PSMB2_RESP_QUERY_INFO;
```

【０１７３】

クライアントは、ＩｎｆｏＴｙｐｅで、これがファイル情報を求める要求か、ファイルシステム情報を求める要求か、セキュリティ情報を求める要求か、それともクォータ情報を求める要求か示すＳＭＢ２＿０＿ＩＮＦＯ＿＊オプションを指定する。ＦｉｌｅＩｄは

（ファイル情報またはセキュリティ情報での）当該のファイルを示す。ファイルが存在するボリュームは、ファイルシステム情報またはクォータ要求に使用される。

【０１７４】

下位情報レベルは、ＦｉｌｅＩｎｆｏＣｌａｓｓに記入され、問い合わせられる情報の種類に依存する。これは、ファイル情報問い合わせではＦＩＬＥ＿ＩＮＦＯＲＭＡＴＩＯＮ＿ＣＬＡＳＳになり、ファイルシステム情報ではＦＳ＿ＩＮＦＯＲＭＡＴＩＯＮ＿ＣＬＡＳＳになる。クォータおよびセキュリティについては０になる。

【０１７５】

入力バッファは、現在、クォータ要求だけに使用される。というのは、クォータ要求は、何が求められるか決定するために、入力時にＳＭＢ２＿ＱＵＥＲＹ＿ＱＵＯＴＡ＿ＩＮＦＯ構造を取るからである。その他の要求では、これは空になる。

【０１７６】

ＯｕｔｐｕｔＢｕｆｆｅｒＬｅｎｇｔｈは、ユーザに返す最大量のデータを指定する。

【０１７７】

ＳＥＴ　ＩＮＦＯ

クライアントがリモートシステムに関する情報を設定することを可能にする。現在、これは、ファイル情報、ファイルシステム情報、セキュリティ情報、またはクォータ情報を設定するのに使用され得る。

【０１７８】

形式

【０１７９】

【表２９】

```
//
// SET_INFO
//

// Request
typedef struct _SMB2_REQ_SET_INFO
{
    USHORT StructureSize;               // = sizeof(SMB2_REQ_SET_INFO)
    UCHAR  InfoType;
    UCHAR  FileInfoClass;
    ULONG  BufferLength;

    USHORT BufferOffset;
    USHORT Reserved;
    union {
        ULONG  Reserved2;
        ULONG  SecurityInformation;  // For SET_SECURITY calls
    };

    UINT64 FileId;

    UCHAR  Buffer[1];                   // File info
} SMB2_REQ_SET_INFO, *PSMB2_REQ_SET_INFO;

// Response
typedef struct _SMB2_RESP_SET_INFO
{
    USHORT StructureSize;               // = sizeof(SMB2_RESP_SET_INFO)
} SMB2_RESP_SET_INFO, *PSMB2_RESP_SET_INFO;
```

【０１８０】

設定される情報の種類および特定のクラスがＱＵＥＲＹ＿ＩＮＦＯについて前述したように、ＦｌａｇｓおよびＦｉｌｅＩｎｆｏＣｌａｓｓフィールドに設定される。提供される入力バッファは設定される情報であり、ＦｉｌｅＩｄはファイルを識別する。

【０１８１】

ＳｅｔＳｅｃｕｒｉｔｙ呼び出しでは、ＳｅｃｕｒｉｔｙＩｎｆｏｒｍａｔｉｏｎフィ

ールドが設定される情報を表す（すなわち、ＯＷＮＥＲ＿ＳＥＣＵＲＩＴＹ＿ＩＮＦＯＲＭＡＴＩＯＮなど）。

【０１８２】

結論

本発明には様々な変更および代替構成の余地があるが、本発明のいくつかの例示的実施形態を図面に示し、本明細書で詳細に説明している。しかしながら、本発明を開示の特定の形に限定する意図はなく、それとは反対に、本発明は、本発明の精神および範囲内に含まれるすべての変更、代替構造、および均等物をカバーするものであることを理解すべきである。

【図面の簡単な説明】

【０１８３】

【図１】本発明の様々な態様が組み込まれ得る汎用コンピューティング環境の一例を示す図である。

【図２】本発明の様々な態様による、クライアントがサーバと通信するネットワーク環境の一例を表すブロック図である。

【図３】本発明の様々な態様による、クライアントとサーバの間の折衝およびセッションセットアップの一例を表すタイミング図である。

【図４】本発明の様々な態様による、作成コンテキストを伴うｃｒｅａｔｅコマンドを含む様々なコマンドを表すタイミング図である。

【図５】本発明の様々な態様による、クライアントとサーバの間の複合要求および可能な応答を表すタイミング図である。

【図６】本発明の様々な態様による、複数のチャネルを介したクライアント／サーバ通信を表す図である。

【図７】本発明の様々な態様による、保護された接続の検証を表す図である。

【図８】本発明の様々な態様による、シンボリックリンクに基づく一例を使用する拡張された誤り戻り情報を表す図である。

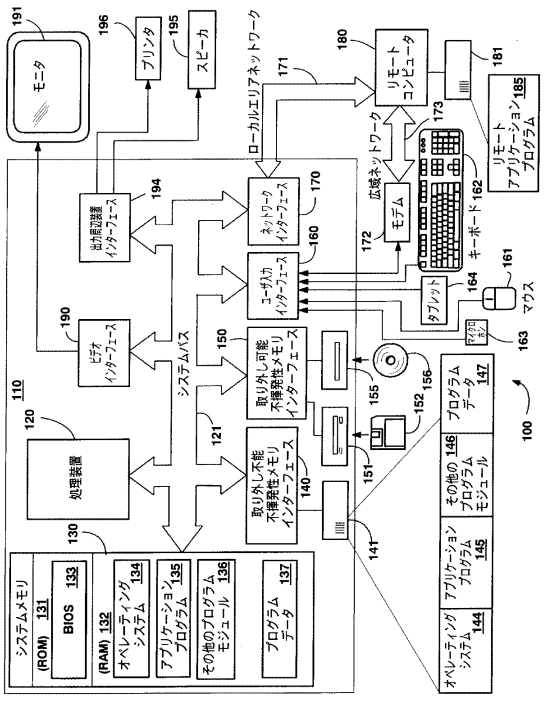
【符号の説明】

【０１８４】

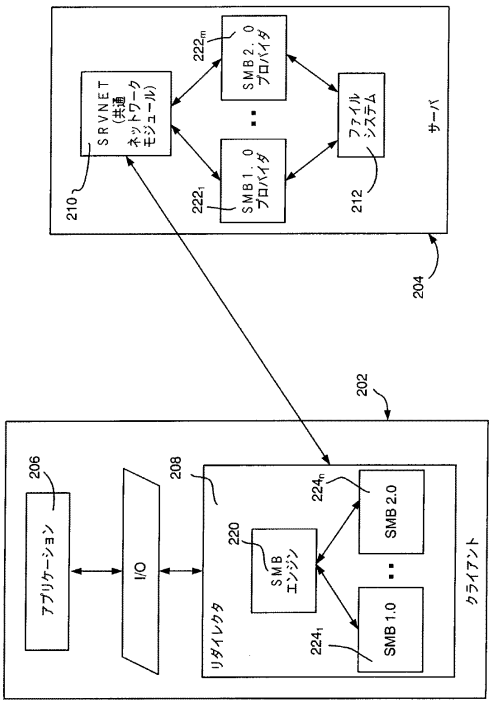
１２０　処理装置
１２１　システムバス
１３０　システムメモリ
１３４　オペレーティングシステム
１３５　アプリケーションプログラム
１３６　その他のプログラムモジュール
１３７　プログラムデータ
１４０　取り外し不能不揮発性メモリインターフェース
１４４　オペレーティングシステム
１４５　アプリケーションプログラム
１４６　その他のプログラムモジュール
１４７　プログラムデータ
１５０　取り外し可能不揮発性メモリインターフェース
１６０　ユーザ入力インターフェース
１６１　マウス
１６２　キーボード
１６３　マイクロホン
１６４　タブレット
１７０　ネットワークインターフェース
１７１　ローカルエリアネットワーク
１７２　モデム
１７３　広域ネットワーク

１８０　リモートコンピュータ
１８５　リモートアプリケーションプログラム
１９０　ビデオインターフェース
１９１　モニタ
１９４　出力周辺装置インターフェース
１９５　スピーカ
１９６　プリンタ
２０２　クライアント
２０４　サーバ
２０６　アプリケーション
２０８　リダイレクタ
２１０　ＳＲＶＮＥＴ（共通ネットワークモジュール）
２１２　ファイルシステム
２２０　ＳＭＢエンジン
２２２_{1}　ＳＭＢ１．０プロバイダ
２２２_{m}　ＳＭＢ２．０プロバイダ

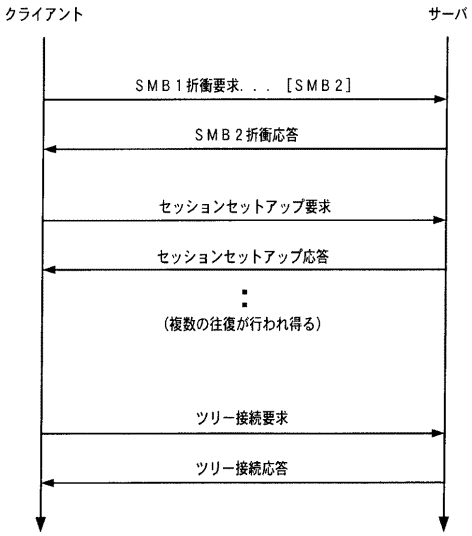
【図１】



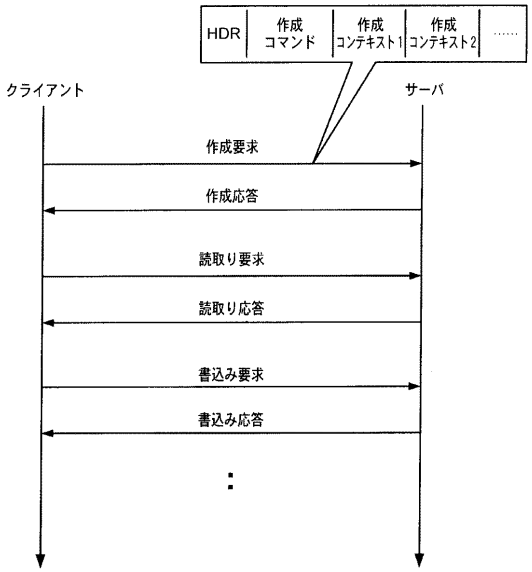
【図２】



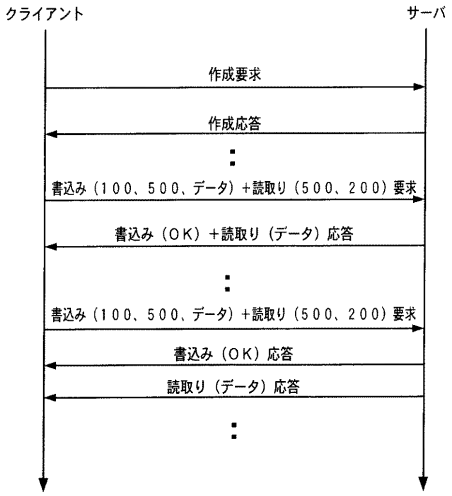
【図３】


クライアント
サーバ
ＳＭＢ１折衝要求．．．［ＳＭＢ２］
ＳＭＢ２折衝応答
セッションセットアップ要求
セッションセットアップ応答
（複数の往復が行われ得る）
ツリー接続要求
ツリー接続応答

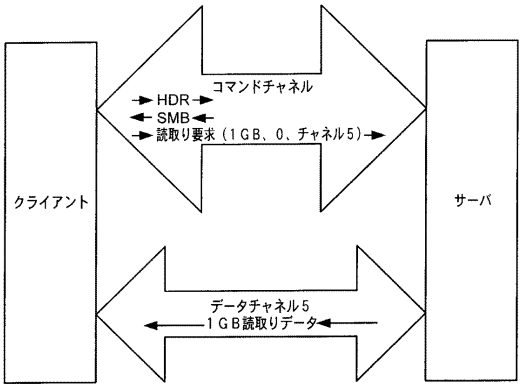
【図４】


HDR
作成コマンド
作成コンテキスト1
作成コンテキスト2
クライアント
サーバ
作成要求
作成応答
読取り要求
読取り応答
書込み要求
書込み応答

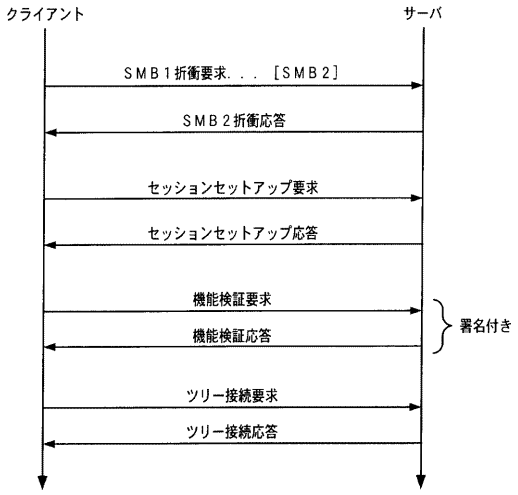
【図５】


クライアント
サーバ
作成要求
作成応答
書込み（１００、５００、データ）＋読取り（５００、２００）要求
書込み（ＯＫ）＋読取り（データ）応答
書込み（１００、５００、データ）＋読取り（５００、２００）要求
書込み（ＯＫ）応答
読取り（データ）応答

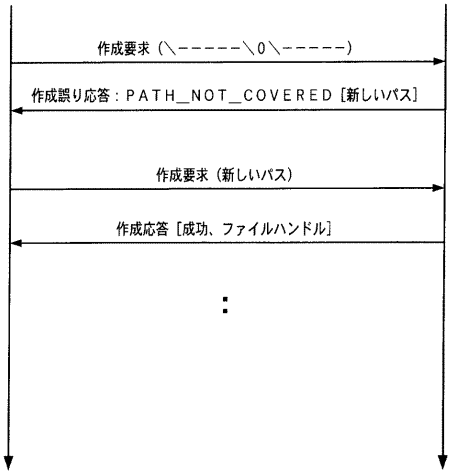
【図６】


コマンドチャネル
HDR
SMB
読取り要求（１ＧＢ、０、チャネル５）
クライアント
サーバ
データチャネル５
１ＧＢ読取りデータ

【図７】


クライアント
サーバ
ＳＭＢ１折衝要求．．．［ＳＭＢ２］
ＳＭＢ２折衝応答
セッションセットアップ要求
セッションセットアップ応答
機能検証要求
機能検証応答
署名付き
ツリー接続要求
ツリー接続応答

【図８】


作成要求（＼－－－－－＼０＼－－－－－）
作成誤り応答：ＰＡＴＨ＿ＮＯＴ＿ＣＯＶＥＲＥＤ［新しいパス］
作成要求（新しいパス）
作成応答［成功、ファイルハンドル］
：

フロントページの続き

(72)発明者　デイヴィッド　クルーズ
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内
(72)発明者　マシュー　ジョージ
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内
(72)発明者　プラディープ　ジュナナ　マダバラプ
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内
(72)発明者　スンダル　サブバラヤン
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

審査官　森谷　哲朗

(56)参考文献　特開平０３－０７４７４５（ＪＰ，Ａ）
特開昭６０－０１９３４１（ＪＰ，Ａ）
特開平０６－０７５８９０（ＪＰ，Ａ）
特開平０５－１４３４８８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０４Ｌ　２９／０６
Ｇ０６Ｆ　１３／００