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

特許第4769191号
(P4769191)

(45)発行日　平成23年9月7日(2011.9.7)　(24)登録日　平成23年6月24日(2011.6.24)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 545A
G06F 12/00 520P
G06F 12/00 518A

請求項の数 13　(全 55 頁)

(21)出願番号　特願2006-536567(P2006-536567)
(86)(22)出願日　平成16年7月22日(2004.7.22)
(65)公表番号　特表2007-509409(P2007-509409A)
(43)公表日　平成19年4月12日(2007.4.12)
(86)国際出願番号　PCT/US2004/023642
(87)国際公開番号　WO2005/045738
(87)国際公開日　平成17年5月19日(2005.5.19)
審査請求日　平成19年7月18日(2007.7.18)
(31)優先権主張番号　10/693,658
(32)優先日　平成15年10月24日(2003.10.24)
(33)優先権主張国　米国(US)

前置審査

(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　110001243
特許業務法人　谷・阿部特許事務所
(74)復代理人　100115624
弁理士　濱中　淳宏
(74)復代理人　100136490
弁理士　中西　英一

最終頁に続く

(54)【発明の名称】ネットワーク上のトランザクション処理ファイルオペレーションのための方法およびシステム

(57)【特許請求の範囲】
【請求項１】
ローカルデバイス上にあり、前記ローカルデバイスに結合されたリモートデバイス上に存在するファイルに対するファイルオペレーションを実行する要求を受信し、前記要求からのトランザクションを検索するよう動作可能なリダイレクタであって、前記リダイレクタは、前記ファイルに関連付けられたファイルシステム識別子（Ｆｉｄ）およびバージョン識別子を含む、前記リモートデバイスからのファイル情報を含む前記ファイルオペレーション結果を受信し、コミットされていないトランザクションに関連付けられた既存のＦＣＢを前記要求に対して使用することができるかどうかを判定し、前記ファイルについて実行されている、前記リダイレクタにより要求されていなかったファイルオペレーションに応答して、前記リモートデバイスが前記ローカルデバイスへシグナリングすべきことを、前記要求中に選択的に示すよう構成されたリダイレクタと、
前記トランザクションをＡＰＩを呼び出して検索し、サーバメッセージブロック（ＳＭＢ）プロトコルに基づいたプロトコルを使用して送信するためにトランザクション情報をフォーマットすることによって、マーシャリングするトランザクションマネージャであって、前記リダイレクタが前記マーシャリングされたトランザクション、および、前記リモートデバイスへの前記要求されたファイルオペレーションを送信して、前記ファイルについて実行されている、前記リダイレクタにより要求されていなかったファイルオペレーションを読み出すことを可能とするトランザクションマネージャと
を備えたことを特徴とするシステム。

【請求項２】
前記トランザクションマネージャは、ファイルシステムに一体化されていないことを特徴とする請求項１に記載のシステム。
【請求項３】
前記リダイレクタは、前記ファイルに関連付けられた前記Ｆｉｄ情報および前記バージョン識別子情報を含む、前記ファイルに関連付けられたファイル制御ブロック（ＦＣＢ）を選択的に作成するよう動作可能であることを特徴とする請求項１に記載のシステム。
【請求項４】
前記リダイレクタは、既存のＦＣＢを前記要求に対して使用することができるかどうかについて判定する際に、前記要求についてのパスネームおよびトランザクションコンテキストを前記既存のＦＣＢに関連付けられたパスネームと比較するよう動作可能であることを特徴とする請求項１に記載のシステム。
【請求項５】
前記トランザクションマネージャは、カーネルレベルトランザクションマネージャであることを特徴とする請求項１に記載のシステム。
【請求項６】
前記プロトコルは、非トランザクション処理リモートファイルオペレーションをサポートすることを特徴とする請求項１に記載のシステム。
【請求項７】
ローカルデバイス上にあって、結合しているネットワークを経由して、前記ローカルデバイス中に存在するファイルについてトランザクション処理ファイルオペレーションに対するものである要求をリモートデバイスから受信し、前記ファイルに関連付けられたファイルシステム識別子（Ｆｉｄ）およびバージョン識別子を前記リモートデバイスへ提供し、サーバメッセージブロック（ＳＭＢ）プロトコルに基づいたプロトコルを使用して、前記要求と共に前記トランザクションを送信するよう動作可能なサーバコンポーネントと、
前記ローカルデバイスにあって、前記リモートデバイスへの前記トランザクション処理ファイルオペレーションに関連したトランザクションメッセージを通信するよう動作可能なトランザクションマネージャと、
前記ローカルデバイスにあって、前記サーバコンポーネントおよび前記トランザクションマネージャに応答して、前記ファイルについて前記要求されたトランザクション処理ファイルオペレーションを実行するよう動作可能であって、前記リモートデバイスのためにトランザクション処理ファイルオペレーション結果を生成するファイルシステムであって、前記ファイルについて実行されている、前記リモートデバイスにより要求されていなかったファイルオペレーションに応答して、選択的に、前記サーバコンポーネントに前記リモートデバイスへシグナリングさせるよう動作可能なファイルシステムと
を備えたこと特徴とするシステム。
【請求項８】
トランザクション処理ファイルオペレーションに対する要求を受信するステップと、
対応するファイルが、ローカルデバイスまたは前記ローカルデバイスに結合されたリモートデバイスにあるかどうかを決定するステップと、
前記オペレーションに対するトランザクションを検索するステップと、
前記要求が前記リモートデバイス上のファイルオペレーションに対するものである場合に、前記要求に対する名前を提供するステップと、
前記ファイルに関連付けられた前記Ｆｉｄおよび前記バージョン識別子情報を含む、前記ファイルに関連付けられたファイル制御ブロック（ＦＣＢ）を選択的に作成するステップであって、既存のＦＣＢがコミットされていないトランザクションに関連付けられているかどうかを判定するステップと、
前記ファイルについて実行されている、前記ローカルデバイス以外のデバイスにより要求されていたファイルオペレーションに応答して、前記リモートデバイスが前記ローカルデバイスへシグナリングすべきことを前記要求中に選択的に示すステップと、

前記トランザクションをＡＰＩを呼び出して検索し、サーバメッセージブロック（ＳＭＢ）プロトコルに基づいたプロトコルを使用して送信するためにトランザクション情報をフォーマットすることによって、マーシャリングするステップと、

前記ファイルが前記リモートデバイス上にある場合に、前記マーシャリングされたトランザクションを前記要求と共にネットワーク上で前記リモートデバイスへ送信するステップと、

前記ファイルについて実行されている、前記ローカルデバイス以外のデバイスにより要求されていたファイルオペレーションから生じる情報を、前記リモートデバイスから受信するステップと

を備えることを特徴とする方法。

【請求項９】

前記リモートデバイスから受信される前記情報は、ファイル識別子（Ｆｉｄ）およびバージョン識別子を含むことを特徴とする請求項８に記載の方法。

【請求項１０】

既存のＦＣＢが前記要求に対して使用することができるかどうかを判定する前記ステップは、前記要求に対するパスネームおよびトランザクションコンテキストを前記既存のＦＣＢに関連付けられたパスネームと比較するステップをさらに含むことを特徴とする請求項８に記載の方法。

【請求項１１】

前記方法は、カーネルモードで実行されることを特徴とする請求項８に記載の方法。

【請求項１２】

前記プロトコルは、非トランザクション処理リモートファイルオペレーションをサポートすることを特徴とする請求項８に記載の方法。

【請求項１３】

請求項８乃至１２いずれかの方法を実行するコンピュータ実行可能命令をストアしたコンピュター読取り可能記録媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

以下に説明する様々な実施形態は、一般にネットワーク通信に関し、より詳細には、それだけには限定されないが、ネットワーク上のトランザクション処理ファイルオペレーションを可能にするための方法およびシステムに関する。

【背景技術】

【０００２】

トランザクションは、長い間、データベースおよびトランザクション処理システムによって提供されてきている。トランザクションは、いくつかのオペレーションを１つの原始的オペレーションに、すなわち個々のオペレーションの結果がともに有効か、または共倒れとなるというオペレーションのグループに一緒にグループ分けすることによって、アプリケーションプログラマにとって望ましい簡略化された失敗モデルを提供する。

【発明の開示】

【発明が解決しようとする課題】

【０００３】

もし、ただ１つのオペレーションが失敗する場合には、そのグループ中におけるすべてのオペレーションの効果は、そのトランザクションに関連するオペレーション数にかかわらず、「元に戻され（ｕｎｄｏｎｅ）」、あるいはロールバックされる。オペレーションの間のこの連帯（ｓｏｌｉｄａｒｉｔｙ）は、失敗の数がいくつの場合においても適用され、結局のところ各トランザクション処理システムは、これらのオペレーションのすべてが適用されている、あるいは、これらのオペレーションのいずれも適用されていないという２つの状態の内の、どちらか一方の状態に至る。

【課題を解決するための手段】

【０００４】
記載される様々な実施形態の態様によれば、ネットワーク上でファイルオペレーションをトランザクション処理する方法およびシステムが提供される。一態様によれば、コンピューティングプラットフォーム（すなわち、クライアント）は、ネットワークを介して、別のコンピューティングプラットフォーム（すなわち、サーバ）上のファイルに遠隔でアクセスすることができる。この態様によれば、これらクライアントおよびサーバはそれぞれ、ＴＭ（ｔｒａｎｓａｃｔｉｏｎ　ｍａｎａｇｅｒトランザクションマネージャ）およびＦＳ（ｆｉｌｅ　ｓｙｓｔｅｍファイルシステム）を含んでいる。このクライアントは、ＲＤＲ（ｒｅｄｉｒｅｃｔｏｒリダイレクタ）も含んでいるが、このサーバは、ＳＲＶ（ｓｅｒｖｅｒ　ｃｏｍｐｏｎｅｎｔサーバコンポーネント）を含んでいる。

【０００５】
動作中においては、ＲＤＲは、リモートトランザクション処理ファイルオペレーション（ｒｅｍｏｔｅ　ｔｒａｎｓａｃｔｅｄ　ｆｉｌｅ　ｏｐｅｒａｔｉｏｎ）の要求を受信する。この要求に応答して、ＲＤＲはこの要求からのトランザクションについて調べ、（例えば、一実施形態におけるこのＴＭによって）サーバへ送信するためにこのトランザクションをマーシャリングする。次いでＲＤＲは、ネットワーク上でこのトランザクション情報（例えば、一実施形態においてはマーシャリングブロブ（ｍａｒｓｈａｌ　ｂｌｏｂ））をサーバに送信する。ＳＲＶはトランザクション情報を受信し、次いでサーバのＴＭおよびＦＳは、トランザクション情報を使用してファイルオペレーションを実施する。次いで、サーバはネットワークを介してファイルオペレーションの結果をクライアントに返送する。

【０００６】
他の態様においては、ＲＤＲは、１つのファイルに対して複数のリモートファイルオペレーショントランザクションがオープンであるようにすることができる。ＲＤＲがトランザクション処理リモートファイルオペレーションを求める新しい要求を受信するとき、ＲＤＲは、リモートファイルの「ダーティ（ｄｉｒｔｙ）」バージョン（すなわち、書き込まれているバージョン）が、クライアント上で知られているかどうかを判定する。次いで、ＲＤＲはファイルについて元のバージョンの代わりに、新しい要求に対するダーティバージョンを使用する。一部の実施形態においては、ＲＤＲは、１つのトランザクション処理されたライトオペレーションが所与のファイルについて一時にオープンであることを可能にするだけである。

【０００７】
さらに他の態様においては、ＲＤＲは、トランザクション処理されたリモートファイルオペレーションを求める新しい要求は、クライアント上ですでに知られているファイル情報を使用することができるかどうかを判定する。この同じファイル情報を使用することができる場合に、ＲＤＲは、このファイル情報のもう１つのコピーを記憶する代わりにこの同じファイル情報を使用する。

【０００８】
さらに他の態様においては、ＲＤＲは、所与のリモートファイルについてのトランザクションに便宜的ロックを関連づけることができる。一実施形態においては、このロックは、ローカルサーバがファイルにアクセスすることを妨げないが、サーバに、ロックが解除されているというメッセージをクライアントに対して送信させる。次いで、トランザクション処理リモートファイルオペレーションを求める新しい要求が、クライアント中にすでにキャッシュされているファイル情報を使用することができるかどうかを判定するに際し、ＲＤＲは所与のファイルについてロックが解除されたかどうかを検査することができる。

【０００９】
非限定的で非網羅的な実施形態について、以下の図を参照しながら説明する。別に指定をしていない限り、様々な図面全体にわたって、同一の参照番号は同一の要素を示している。

【発明を実施するための最良の形態】
【００１０】
実施例のネットワーク環境
前述のように、トランザクションは、データベースおよびトランザクション処理システムにおいて使用されてきているが、以下の実施形態においては、トランザクションは、リモートファイルオペレーションのために使用される。図１は、クライアントがネットワーク１０１を介してクライアント上でファイルオペレーションをトランザクション処理することができるシステム１００を示している。図１の実施例のネットワーク環境においては、クライアント装置と呼ぶこともできる複数のクライアントコンピューティングデバイス１０５、１１０、１１５、および１２０が、ネットワーク１０１を介して少なくとも１つのサーバ装置１２５に結合されている。ネットワーク１０１は、有線ネットワークおよび／または無線ネットワークを含むことができる様々な従来のネットワークトポロジおよびタイプのいずれをも表すことが意図されている。ネットワーク１０１はさらに、パブリックプロトコルおよび／または独自開発プロトコルを含めて、様々な従来のネットワークプロトコルのうちのいずれをも利用することができる。ネットワーク１０１は、例えばインターネット、ならびに場合によっては１つまたは複数のＬＡＮ（ｌｏｃａｌ ａｒｅａ ｎｅｔｗｏｒｋローカルエリアネットワーク）、ＷＡＮ（ｗｉｄｅ ａｒｅａ ｎｅｔｗｏｒｋワイドエリアネットワーク）などの少なくとも一部分を含むことができる。
【００１１】
クライアント装置１０５は、それだけには限定されないが、デスクトップＰＣ（ｐｅｒｓｏｎａｌ ｃｏｍｐｕｔｅｒパーソナルコンピュータ）、ワークステーション、メインフレームコンピュータ、インターネットアプライアンス、およびゲーミングコンソール（ｇａｍｉｎｇ ｃｏｎｓｏｌｅ）を含めて様々な従来のコンピューティングデバイスのうちのいずれも含むことができる。さらに、ネットワーク１０１に関連するクライアント装置は、有線リンクおよび／または無線リンクによってネットワーク１０１と情報をやりとりすることができるＰＤＡ（ｐｅｒｓｏｎａｌ ｄｉｇｉｔａｌ ａｓｓｉｓｔａｎｔ携帯型個人情報端末）１１０、ラップトップコンピュータ１１５、およびセルラ電話１２０などを含むことができる。さらにまた、１つまたは複数のクライアント装置１０５、１１０、１１５、および１２０は、これら同じタイプのデバイス、または代わりの異なるタイプのデバイスを含むことができる。
【００１２】
サーバ装置１２５は、コンピューティングデバイス１０５、１１０、１１５および１２０に対して、様々なデータおよび／または機能のうちのいずれも提供することができる。このデータは、パブリックに利用可能であり、あるいは代わりに制限されることもあり、例えばあるユーザだけに制限され、または適切な料金が支払われる場合にしか利用可能でないなどのこともある。
【００１３】
サーバ装置１２５は、ネットワークサーバ、およびアプリケーションサーバ、またはその両者による組合せのうちの少なくとも１つである。サーバ装置１２５は、コンテンツのソースとなる任意のデバイスであり、クライアント装置１０５、１１０、１１５および１２０は、このようなコンテンツを受信する任意のデバイスを含んでいる。したがって、ピアツーピアネットワークにおいて、このコンテンツのソースとなるこのデバイスは、サーバ装置と呼ばれ、このコンテンツを受信するデバイスは、クライアント装置と呼ばれる。両方のタイプの装置は、本明細書中で説明される実施形態の例に従って、オペレーティングシステムおよびアプリケーションを含めて、ソフトウェアプログラムをロードし実行することが可能である。さらに、データおよび機能は、クライアント装置１０５、１１０、１１５および１２０の間で共有することができる。すなわち、サーバ装置１２５だけが、各クライアント装置に対するデータおよび／または機能のソースであるというわけではない。
【００１４】

データソース１３０または１３５において、オペレーティングシステムおよびアプリケーションを含めてソフトウェアプログラムが、実行するためにサーバ装置１２５またはクライアント装置１０５、１１０、１１５、および１２０のうちのどれか１つのために作成され、かつ／またはそのどれか１つに対して提供される。一貫性をもたせるために、以下の説明では、当技術分野において知られているように、単独または組合せのいずれでも、少なくともソフトウェアプログラム、オペレーティングシステム、およびアプリケーションのどれかを包含する「アプリケーション」と呼ばれている。さらに、これらのアプリケーションは、データソース１３０からのものとしてオフラインで、または、データソース１３５からのものとしてオンラインで、サーバ装置１２５に対して配布される。さらにまた、これらのアプリケーションは、一般にサーバ装置１２５またはデータソース１３５からオンラインでクライアント装置１０５、１１０、１１５、および１２０に対して配布される。これらのオフラインによる配布のための手段および方法についても同様に知られている。

【００１５】

以下で説明する様々な実施形態によれば、共に装置１０５、１１０、１１５、１２０、１２５における、またはそれらの中にあるデータおよび機能の少なくとも１つの配布は、トランザクションとして実装することができる。より詳細には、トランザクションは、図１の実施例のように、デバイス１０５、１１０、１１５、１２０、１２５のうちの１つの内部においてまたはネットワーク環境の中において、１つの原子的オペレーションとして同期してまたは非同期で実行されるオペレーションのグループである。このネットワーク上で実施されるトランザクション処理リモートファイルオペレーションの実施例について、図２～図７とともに以下にさらに詳細に説明する。

【００１６】

トランザクション処理リモートファイルオペレーション

図２は、図１におけるシステム１００中の（例えば、デバイス１０５、１１０、１１５、１２０、１２５から選択される）２つのデバイスのコンポーネントを示している。これらは、トランザクション処理リモートファイルオペレーションのためのクライアント２０２およびサーバ２０４として動作する。この実施形態においては、クライアント２０２もサーバ２０４も共に、あるバージョンのＭｉｃｒｏｓｏｆｔ（登録商標）Ｗｉｎｄｏｗｓ（登録商標）オペレーティングシステムを使用している。他の実施形態においては、異なるオペレーティングシステムを使用することができる。

【００１７】

この実施形態においては、クライアント２０２は、アプリケーション２１２、Ｉ／Ｏ（ｉｎｐｕｔ／ｏｕｔｐｕｔ入出力）マネージャ２１４、ファイルシステム（ＦＳ）２１６、リダイレクタセレクタ（ｒｅｄｉｒｅｃｔｏｒ　ｓｅｌｅｃｔｏｒ）２１８、トランザクションマネージャ（ＴＭ）２２２、およびリダイレクタ（ＲＤＲ）２２０を含んでいる。この実施形態において、サーバ２０４は、サーバコンポーネント（ＳＲＶ）２３４、Ｉ／Ｏマネージャ２１４Ａ、ＦＳ２１６ＡおよびＴＭ２２２Ａを含んでいる。この実施形態において、クライアント２０２およびサーバ２０４は、ネットワーク１００（図１）を介して互いに情報をやりとりすることができる。一部の実施形態においては、これらのコンポーネントは、ソフトウェアにより実装される。

【００１８】

この「Ｗｉｎｄｏｗｓ（登録商標）」の実施形態において、Ｉ／Ｏマネージャ２１４および２１４Ａ、ならびにＦＳ２１６および２１６Ａは、ＮＴＦＳ（ＮＴ　ｆｉｌｅ　ｓｙｓｔｅｍＮＴファイルシステム）によって実装され、リダイレクタセレクタ２１８は、ＭＵＰ（ｍｕｌｔｉｐｌｅ　ＵＮＣ　ｐｒｏｖｉｄｅｒ複数ＵＮＣプロバイダ）によって実装され、ここでＵＮＣは、Ｕｎｉｆｏｒｍ　Ｎａｍｉｎｇ　Ｃｏｎｖｅｎｔｉｏｎ（汎用名前付け規則）の頭字語である。したがって、リダイレクタセレクタ２１８は、本明細書中においてＭＵＰ２１８とも呼ばれる。さらに、Ｍｉｃｒｏｓｏｆｔ（登録商標）Ｗｉｎｄｏｗｓ（登録商標）オペレーティングシステムのＲＤＲおよびＳＲＶが、（追加機能と

ともに）それぞれＲＤＲ２２０およびＳＲＶ２３４を実装する。このＭｉｃｒｏｓｏｆｔ（登録商標）Ｗｉｎｄｏｗｓ（登録商標）オペレーティングシステムのＲＤＲおよびＳＲＶに対する追加機能の一例については、以下で説明する。さらにまた、この実施形態において、ＴＭ２２２およびＴＭ２２２Ａは、この実施形態の例においてはカーネルレベルトランザクションマネージャとして実装され、以下にさらに詳細に説明する。他の実施例においては、異なるＩ／Ｏマネージャ、ファイルシステム、リダイレクタセレクタ、ＴＭ、および／またはＲＤＲを使用することもできる。

【００１９】

図３は、クライアント２０２とサーバ２０４の間のトランザクション処理リモートファイルオペレーション（図２）についての実施例の処理フローを示している。図２および図３を参照すると、トランザクション処理リモートファイルオペレーションは、一実施形態によれば、以下のように実施される。

【００２０】

ブロック３０２において、ＲＤＲ２２０は、サーバ２０４中に存在するファイルに対するトランザクション処理ファイルオペレーションに対する要求を受信する。典型的なファイルオペレーションは、新しいファイルを作成する工程、ファイルを読み取る工程、ファイルを書き込む工程、ファイルをコピーする工程、ファイルの名前を変更する工程などを含んでいる。この実施形態においては、トランザクション処理ファイルオペレーションを求める要求は、アプリケーション２１２によって生成さる。このアプリケーションは、図２に示すようにユーザレベルのアプリケーションである。この要求は、トランザクションコンテキストについてのフィールドを含む構造を使用する。この要求は、Ｉ／Ｏマネージャ２１４によって受信され、このＩ／Ｏマネージャは、この要求がローカルファイルに対するものか、それともリモートファイルに対するものかを判定する。この実施形態においては、Ｉ／Ｏマネージャ２１４は、このＭｉｃｒｏｓｏｆｔ（登録商標）Ｗｉｎｄｏｗｓ（登録商標）オペレーティングシステムの標準コンポーネントである。例えば、アプリケーション２１２は、（＼＼ｓｅｒｖｅｒ＼ｓｈａｒｅ＼ｓｕｂｄｉｒｅｃｔｏｒｙ＼ｆｉｌｅｎａｍｅの形式の）このＵＮＣネームを用いて、Ｉ／Ｏマネージャ２１４への呼び出しを介して、この要求を行うことができる。次いでＩ／Ｏマネージャ２１４は、この要求をＭＵＰ２１８に渡す。トランザクションについては複数のハンドルが存在し、所与のファイルについては複数のトランザクションが存在することができる。他方、この要求がこのクライアント上のファイルに対するものである場合には、Ｉ／Ｏマネージャ２１４は、標準的な方法によって、この要求をＮＴＦＳ２１６に渡すはずである。

【００２１】

次いでＭＵＰ２１８は、この要求を実施するために必要とされるリダイレクタを見つける。このケースにおいては、このリダイレクタは、ＲＤＲ２２０である。この実施形態において、ＭＵＰ２１８は、このＭｉｃｒｏｓｏｆｔ（登録商標）Ｗｉｎｄｏｗｓ（登録商標）オペレーティングシステムの標準コンポーネントである。この実施形態においては、ＲＤＲ２２０は、このＭｉｃｒｏｓｏｆｔ（登録商標）Ｗｉｎｄｏｗｓ（登録商標）オペレーティングシステムＲＤＲの１つのバージョンであり、追加部分を用いることにより、このＲＤＲは、ＴＭ２２２と対話してトランザクションを実施することができる。この追加部分は例えば、要求からトランザクション処理ファイルオペレーションについてのトランザクションコンテキストを検索し、トランザクション処理ファイルオペレーションについてのＦＣＢを割り当て、ネットワーク上でトランザクションをリモートデバイスに送信し、（ファイル識別子およびバージョン識別子を含めて）これらのトランザクション処理ファイルオペレーションについての応答を受信し、ＴＭ２２２の指示の下にトランザクションオペレーションを実施し、ＲＤＲ２２０がトランザクションの状態に関して通知され続けることができるようにするためにＴＭ２２２とともにリソースマネージャとして積極的に参加する（エンリスト：ｅｎｌｉｓｔ）機能を含んでいる。一部の実施形態においては、ＲＤＲ２２０は、同時係属の同一出願人による「Transactional File System」という名称の２０００年３月３０日に出願の、米国特許出願第０９／５３９，２３３号明細書

および米国出願番号［整理番号第ＭＳ１－１７８１ＵＳ］に説明されているように実装される。リソースマネージャとして積極的に参加すること（ｅｎｌｉｓｔｉｎｇ）については以下で説明する。ＲＤＲ２２０は、このトランザクションをバッファリングするためのリソース、キャッシュマップ、ＦＣＢ（ｆｉｌｅ　ｃｏｎｔｒｏｌ　ｂｌｏｃｋファイル制御ブロック）、ＦＯＢＸ（ｆｉｌｅ　ｏｂｊｅｃｔ　ｅｘｔｅｎｓｉｏｎファイルオブジェクト拡張部分）、およびこのトランザクションおよび要求を処理するために必要な他の構造を含んでいる。

【００２２】

ブロック３０４において、ＲＤＲ２２０は、ＴＭ２２２からこのトランザクションを検索し、このトランザクションをサーバ２０４に送信するためにマーシャリングする。一実施形態においては、ＲＤＲ２２０は、ＴＭ２２２によって表示される（その実施形態について以下で説明している）ＡＰＩを呼び出すことによってこのトランザクションを検索し、トランザクションをサポートするように拡張されている１つのバージョンのＳＭＢプロトコルを使用して送信するためにこのトランザクション情報（例えば、マーシャリングブロブ）をフォーマットすることによってこのトランザクションをマーシャリングする。例示の一実施形態のＳＭＢ拡張部分については、表１～表３とともに以下で要約している。ブロック３０６において、ＲＤＲ２２０は、矢印２３６で示すようにこのトランザクションおよびこの要求をサーバ２０４に対して送信する。ブロック３０８において、ＲＤＲ２２０は、サーバ２０４からこのファイルオペレーションの結果を受信する。例えば、サーバ２０４は、前述のファイル識別子およびバージョン識別子を含むこの要求に対する応答を送信する。この実施形態においては、ＳＲＶ２３４は、このＭｉｃｒｏｓｏｆｔ（登録商標）Ｗｉｎｄｏｗｓ（登録商標）オペレーティングシステムＳＲＶの１つのバージョンである。追加機能部分を用いることにより、このＳＲＶは、トランザクション処理リモートファイルオペレーション中にファイル識別子およびバージョン識別子をクライアントに対して渡すことを含め、ネットワーク上でクライアントと対話してＳＭＢに対する拡張部分を使用してトランザクションを実施することができる。

【００２３】

【表１】

拡張部分	記述
新しいコマンド: NT_TRANSACT_CREATE2を追加する	マーシャリングされたトランザクションを獲得し、2つの構造REQ_CREATE_WITH_EXTRA_OPTIONSおよびRESP_CREATE_WITH_EXTRA_OPTIONSをネットワーク上で送信する。これら2つの構造は、それぞれ表2および表3で定義され、SMB構造: REQ_CREATE_WITH_SD_OR_EAおよびRESP_EXTENDED_CREATE_WITH_SD_OR_EAの拡張部分である。
新しい機能ビット: CAP_TXFを追加する	CAP_TXFは、サーバによってセットまたはリセットされてこのサーバがトランザクションをサポートするかどうかを示す。CAP_TXFは、SMBネゴシエーション応答（SMB Negotiate Response）の一部分である。この実施形態においては、CAP_TXFはサーバサポートトランザクションを示す0x20000として定義される。
新しいフラグ: SMB_FIND_TRANSACTED_OPERATIONをSMBFIND要求に追加する。このFIND要求（REQ_FIND_FIRST2）構造は、表4で定義され、その応答構造は表5で定義される。	SMB_FIND_TRANSACTED_OPERATIONは、トランザクションが使用されていることを示す。この実施形態においては、ファインドオペレーション（find operation）がハンドルベースではなくてパスベース（path based）であるので、このフラグが使用される。この実施形態においては、このフラグは、0x20として定義される。このフラグが設定される場合に、トランザクション情報は、FINDコマンドおよびECHOコマンドの終わりに送信される。
ECHOコマンドを拡張してトランザクション状態変化の通知を送信する。この要求／応答構造は、表6および7で定義される。	SMBECHOコマンドを拡張してトランザクションオペレーションのプリプリペア状態、プリペア状態、コミット状態、ロールバック状態の通知をサーバからクライアントへと提供する。

表１

【００２４】

【表２】

ＲＥＱ＿＿ＣＲＥＡＴＥ＿＿ＷＩＴＨ＿＿ＥＸＴＲＡ＿＿ＯＰＴＩＯＮＳ

フィールド	内容の記述
_ULONG(Flags)	フラグNT_CREATE_xxxの作成
_ULONG(RootDirectoryFid)	相対的なオープンについてのオプション的ディレクトリ
ACCESS_MASK DesiredAccess	望ましいアクセス（NTフォーマット）
LARGE_INTEGER AllocationSize	バイト単位の　初期割り当てサイズ
_ULONG(FileAttributes)	ファイル属性
_ULONG(ShareAccess)	共用アクセス
_ULONG(CreateDisopsition)	ファイルが存在するか否かを獲得するアクション
_ULONG(CreateOptions)	新しいファイルを作成するためのオプション
_ULONG(SecurityDescriptorLength)	バイト単位のSD長
_ULONG(EaLength)	バイト単位のEA（extended attribute拡張属性）長
_ULONG(NameLength)	キャラクタによる名前の長さ
_ULONG(ImpersonationLevel)	セキュリティQOS（Quality of Serviceサービス品質）情報
_UCHAR SecurityFlags	セキュリティQOS情報
_ULONG(TransactionLength)	バイト単位のマーシャリングされたトランザクションコンテキスト長
_ULONG(EfsStreamLength)	バイト単位の$EFS（encrypted file system暗号化ファイルシステム）ストリーム長
UCHAR Buffer[1]	データバッファ中でDWORD（4バイト）境界に位置合わせされた、ファイルネーム用のバッファ
UCHAR Name[]	(NULで終了していない）ファイルネーム

表２

【００２５】

【表３－１】

RESP__CREATE__WITH__EXTRA__OPTIONS

フィールド	内容の記述
UCHAR OplockLevel	許可された便宜的ロックレベル
UCHAR ExtendedResponse	拡張された応答について1にセットする
_USHORA(Fid)	ファイルID
_ULONG(CreateAction)	獲得されたアクション
_ULONG(EaErrorOffset)	EAエラーのオフセット
TIME CreationTime	ファイルが作成された時刻
TIME LastAccessTime	ファイルがアクセスされた時刻
TIME LastWriteTime	ファイルが最後に書き込まれた時刻
TIME ChangeTime	ファイルが最後に変更された時刻
_ULONG(FileAttributes)	ファイル属性
LARGE_INTEGER AllocationSize	割り当てられたバイト数
LARGE_INTEGER EndOfFile	ファイルオフセットの終了
_USHORT(FileType)	ファイルのファイルタイプ
_USHORT(DeviceState)	IPCデバイス（例えば、パイプ）の状態
BOOLEAN Directory	この構造がディレクトリである場合に真（TRUE）
UCHAR VolumeGuid[16]	ボリュームGUID（Globally Unique IDグローバルに一意のID）
UCHAR FileId[8]	ファイルID
_ULONG(MaximalAccessRights)	セッション所有者に対するアクセス権

【００２６】

【表３－２】

フィールド	内容の記述
_ULONG(GuestMaximalAccessRights)	ゲストに対する最大限のアクセス権
LARGE_INTEGER FilesystemFid;	同じパスネーム（pathname）を有する異なるファイルを区別するサーバ上のNTFSFid。トランザクション（TxF）を使用しながら、同じパスネームで、2つの異なるファイルを示すことができる。
_ULONG(VersionNum);	オープンされているファイルのTxFバージョン#

表３

【００２７】

【表４】

ＲＥＱ＿ＦＩＮＤ＿ＦＩＲＳＴ２

フィールド	内容の記述
_USHORT(SearchAttributes)	検索属性
_USHORT(SearchCount)	戻すべき最大エントリ数
_USHORT(Flags)	追加情報：設定ビット ０－この要求の後に検索をクローズ（close）する １－末尾に到達した場合に、検索をクローズする ２－レジュームキーを戻す
_USHORT(InformationLevel)	情報レベル
_ULONG(SearchStorageType)	検索ストレージタイプ
UCHAR Buffer[1]	ファイルネーム

表４

【００２８】

【表５】

Ｒｓｐ＿ｆｉｎｄ＿ｆｉｒｓｔ２

フィールド	内容の記述
_USHORT(Sid)	ハンドルを検索する
_USHORT(SearchCount)	戻されるエントリ数
_USHORT(EndOfSearch)	最後のエントリは戻されたか？
_USHORT(EaErrorOffset)	EAエラーの場合、EAリストにオフセットする
_ULONG(SearchStorageType)	検索ストレージタイプ
_USHORT(LastNameOffset)	サーバが検索を再開する必要がある場合、最後のエントリのファイルネームまでデータにオフセットし、そうでない場合には、0とする

表５

【００２９】

【表６】

ＲＥＱ＿ＥＣＨＯ

フィールド	内容の記述
UCHAR WordCount	パラメータワードのカウント = 1
_USHORT(SearchCount)	戻されたエントリ数
_USHORT(EndOfSearch)	最後のエントリは戻されたか?
_USHORT(EaErrorOffset)	EAエラーの場合、EAリストにオフセットする
_USHORT(LastNameOffset)	サーバが検索を再開する必要がある場合、最後のエントリのファイルネームまでデータにオフセットし、そうでない場合には、0とする

表６

【００３０】

【表７】

Ｒｓｐ＿ｅｃｈｏ

フィールド	内容の記述
UCHAR WordCount	パラメータワードのカウント = 1
_USHORT(SequenceNumber)	このエコーのシーケンス番号
_USHORT(ByteCount)	データバイトのカウント; min = 4
UCHAR Buffer[1]	エコーされたデータ

表７

【００３１】

図３Ａは、一実施形態におけるブロック３０２（図３）をさらに詳細に示している。ブロック３１２において、ＲＤＲ２２０は、この要求されたファイルオペレーションについてのトランザクションコンテキストを検索する。トランザクション処理リモートファイルをオープンする際に、ＲＤＲ２２０は、トランザクションがこの要求にすでに関連づけられているかどうかを判定する。例えば、一実施形態において、トランザクションはそれをスレッド（ｔｈｒｅａｄ）にアタッチすることによって要求に関連付けられるが、他の実施形態においては、異なる方法を使用してトランザクションを要求に関連づけることができる。一実施形態においては、ＲＤＲ２２０は、この要求がそれに関連づけられたトランザクションハンドルを有するかどうかを確かめることによってこのオペレーションを実施する。関連づけられたトランザクションハンドルを有する場合には、この要求は、その既存のトランザクションに加え（ｊｏｉｎ）られる。有していない場合には、ＲＤＲ２２０は、非トランザクション処理要求に対する標準的な方法でこの要求を処理する。

【００３２】

次いで、ＲＤＲ２２０は、この要求にＦＣＢを割り当てる。前述のように、複数の要求を有する複数のトランザクションが、所与のファイルをオープンすることもある。したがって、ブロック３０２（図３）の一実施形態においては、ＲＤＲ２２０が既存のＦＣＢをこの要求に対しても使用できるかどうかについて判定するブロック３１４が、実行される。この実施形態において、ＲＤＲ２２０は、この要求のファイル（すなわち、パスネーム）およびこの要求を作るスレッドに関連付けられたこのトランザクションコンテキストが、既存のＦＣＢのこれらとマッチングするかどうかを確認する。例えば、第２の要求の処理中に、この同じファイルの２つの書込みオペレーションが、同じトランザクション中で要求されたとする場合には、この第１の要求のために既にＦＣＢが存在しているはずである。両方のオペレーションが共に書込みオペレーションであるので、この同じＦＣＢを両者に対して使用することができる。

【００３３】

ブロック３１４において、ＲＤＲ２２０が、ＦＣＢが同じトランザクションコンテキストならびに同じファイル（すなわち、パスネーム）および同じバージョンと共に存在すると判定する場合には、ブロック３１６において、既存のＦＣＢがこの要求に対して使用される。一部の実施形態において、ＲＤＲ２２０は、最新バージョンを有するＦＣＢを使用することになる。例えば、ファイルの読取りオペレーションがこの同じファイルのコミットされていない書込みオペレーションに続く場合、ＲＤＲ２２０は、この書込みオペレーションが現在使用しているファイルのこのバージョンを使用することになる。このアプローチによって、キャッシングのさらに効率的な使用が可能になる。

【００３４】

しかし、ブロック３１４において、この要求に対する既存のＦＣＢを使用することができない場合には、ブロック３１８においてＲＤＲは、この要求に対して新しいＦＣＢを作成する。代替的な実施形態においては、新しいＦＣＢが、要求ごとに作成される。

【００３５】

図４は、この同じファイルに対するコミットされていない複数のトランザクション処理要求の一例を示している。図４に示すように、オペレーション４０１は、ファイルの読取りオペレーションを求める要求に対応する。すなわち、このファイルオペレーションは、「読取りのためにオープンする」オペレーションである。オペレーション４０１は、ハンドルＨ１と、このハンドルに関連付けられたトランザクションＴ１を有する。ＲＤＲ２２０（図２）が要求するこのファイルのバージョンは、バージョンＡとして示される。これがこのファイルに対しての最初のコミットされていないトランザクションであると想定すると、バージョンＡが、サーバ２０４（図２）から読み出され、クライアント２０２（図２）中にキャッシュされる。

【００３６】

後の時刻において、オペレーション４０２がこの同じファイルに対して要求される。この例においては、オペレーション４０２も読取りオペレーションであり、ハンドルＨ２およびトランザクションＴ２を有する。このトランザクションは、オペレーション４０１のトランザクションとは異なるので、ＲＤＲ２２０は、この場合にもサーバ２０４からこのファイルのバージョンＡが読み出される。

【００３７】

この例においては、次いでオペレーション４０３が、オペレーション４０２と同じトランザクションにおいてこの同じファイルに対して要求される。したがって、オペレーション４０３はハンドルＨ３を有し、トランザクションＴ２に結合（ｊｏｉｎ）される。しかし、この例においては、オペレーション４０３は書込みオペレーションであり、したがってＲＤＲ２２０は、このファイルのバージョンＢをローカルに記憶している（例えば、キャッシュしている）。バージョンＢは、時として「ダーティバージョン」と呼ばれることがある。

【００３８】

次いでオペレーション４０４が、オペレーション４０２および４０３と同じトランザク

ションにおいて、この同じファイルに対して要求される。したがって、オペレーション４０４はハンドルＨ４を有し、これもまたトランザクションＴ２に結合される。この例においては、オペレーション４０４は読取りオペレーションである。この実施形態においては、ブロック３１４（図３Ａ）の結果により、ＲＤＲ２２０は、オペレーション４０４に対するバージョンＢを記憶しており、おそらくはキャッシュしていることになる。

【００３９】

次いで、オペレーション４０５が、異なるトランザクションにおいてこの同じファイルに対して要求される。したがって、オペレーション４０５はハンドルＨ５を有し、新しいトランザクションＴ３に関連付けられる。このトランザクションは、以前のオペレーションのトランザクションとは異なるので、一実施形態において、ＲＤＲ２２０は、この場合にもこのファイルのバージョンＡをサーバ２０４から読み出す。他の実施形態においては、ＲＤＲ２２０はサーバ２０４（図２）を調べなくても、依然としてバージョンＡが現行バージョンであることを認識しており、この「ローカル」バージョンＡを使用する。例えば、この代替的な実施形態では、便宜的ロックを使用してサーバ２０４中に存在するこのファイルのどのようなより新しいバージョンについても知ることができるようになる。すなわち、ＲＤＲ２２０は、サーバ２０４上のファイルに対する書込みを妨げることのない便宜的ロックを、このファイルに関連付けることができるが、サーバ２０４が、このロックが解除されたことをＲＤＲ２２０にシグナリングするようにする。このようなケースにおいては、次いでＲＤＲ２２０は、バージョンＡはもはや最新のバージョンではないことを知るはずである。さらに他の実施形態においては、ＲＤＲ２２０は、サーバ２０４に照会してこのファイルの最新バージョンを決定し、次いでこの最新のバージョンに関連付けられた既存のＦＣＢを再使用することができる。

【００４０】

次いで、オペレーション４０６において、トランザクションＴ２がコミット（ｃｏｍｍｉｔ）される。これは、サーバ２０４上でバージョンを変更する効果を有する。サーバ２０４上に記憶されたこの新しいバージョンは、バージョンＣとして示される。前述したように、ＲＤＲ２２０は、すべてのリモートトランザクション中にリソースマネージャとして積極的に参加（エンリスト）するので、ＲＤＲ２２０は、サーバ２０４がこのファイルの新しいバージョンを有することを学習する。

【００４１】

次いで、オペレーション４０１と同じトランザクションにおいてこの同じファイルに対して、オペレーション４０７が要求される。したがって、オペレーション４０７はハンドルＨ６を有し、トランザクションＴ１に結合される。しかし、ＲＤＲ２２０はサーバ２０４上のこのファイルのバージョンＣを知っているので、ＲＤＲ２２０は、このオペレーションに対するバージョンＣを記憶しており、おそらくキャッシュしている。一部の実施形態においては、ＲＤＲ２２０は、サーバ２０４からバージョンＣを読み出す。

【００４２】

同様に、オペレーション４０５と同じトランザクションによってこの同じファイルに対して、オペレーション４０８が要求されるとき、オペレーション４０８はハンドルＨ７を有し、トランザクションＴ３に結合される。この場合にも、ＲＤＲ２２０がサーバ２０４上のこのファイルのバージョンＣについて知っているので、ＲＤＲ２２０は、このオペレーションに対するバージョンＣを記憶しており、おそらくキャッシュしている。

【００４３】

図５は、一実施形態に従って、クライアント２０２（図２）からのキャッシュされたデータがどのようにしてサーバ２０４（図２）にフラッシュされるの（すなわち、永続的に記憶される）かを示している。図２および図５を参照すると、クライアント２０２は、一実施形態によれば、以下で説明するようにサーバ２０４にデータをフラッシュする。

【００４４】

ブロック５０２において、このデータを生成するアプリケーションは、呼出しを行うか、あるいはこのトランザクションをコミットする要求を発行する。この呼出しまたは要求

は、ＴＭ２２２に渡される。それに応答して、ＴＭ２２２は、（この実施例のトランザクションマネージャに関して以下で説明する）プリプリペア通知（Ｐｒｅ－ｐｒｅｐａｒｅ　Ｎｏｔｉｆｉｃａｔｉｏｎ）を生成する。

【００４５】

この実施形態において、ＲＤＲ２２０は、ブロック５０４に示すようにＴＭ２２２からこのプリプリペア通知を受信する。それに応答してＲＤＲ２２０は、このネットワークを介してこのデータをＳＲＶ２３４に対してフラッシュする。ＳＲＶ２３４は、次にこのデータをＮＴＦＳ２１６Ａに渡す。これらのオペレーションは、ブロック５０６によって示される。一部の実施形態においては、サーバ２０４のＴＭ２２２Ａは、いつこのプリプリペアオペレーションが完了するかについてＲＤＲ２２０にシグナリングする。ブロック５０４およびブロック５０６は、（実施例のトランザクションマネージャに関して以下で説明する）プリペア（Ｐｒｅｐａｒｅ）オペレーションが実施されるよりも前に、サーバ２０４上に書き込まれるべきクライアント２０２からのデータが、サーバ２０４上に存在していることを確実にする。

【００４６】

ブロック５０８において、ＲＤＲ２２０は、（実施例のトランザクションマネージャに関して以下で説明する）プリペア通知をＴＭ２２２から受信する。一実施形態においては、ＲＤＲ２２０は、このプリペア通知に応答してプリペア通知メッセージをサーバ２０４に送信し、このプリペア通知は、ＴＭ２２２Ａに渡される。次にＴＭ２２２Ａは、このプリペア通知をＮＴＦＳ２１６Ａに渡す。これらのオペレーションは、ブロック５１０および５１２によって示される。このプリペア通知は、クライアント２０２およびサーバ２０４が、このデータをコミットするか、またはロールバックすることができるような方法で、このデータを記憶するようにする。一部の実施形態においては、サーバ２０４のＴＭ２２２Ａは、ＲＤＲ２２０へ、いつこのプリペアオペレーションが完了するかをシグナリングする。次いで、このデータは、ブロック５１４で示すように、標準の２相コミットオペレーション（例えば、このトランザクションをコミットし、または、中止するオペレーション）を使用して処理される。

【００４７】

以上では、トランザクションマネージメントは、別々のＴＭコンポーネント（すなわち、ＴＭ２２２およびＴＭ２２２Ａ）を使用して実施されるものとして説明しているが、他の実施形態においては、このトランザクションマネージメントインフラストラクチャは、このファイルシステムインフラストラクチャに一体化することができる。さらに、このような一体化された実施形態においては、これらのトランザクションメッセージ（例えば、以下で説明するようにプリプリペア（ＰｒｅＰｒｅｐａｒｅ）、プリペア（Ｐｒｅｐａｒｅ）、コミット（Ｃｏｍｍｉｔ）、アボート（Ａｂｏｒｔ）など）は、この伝送チャネル上をファイルメッセージと共に流れる。

【００４８】

実施例のトランザクションマネージャ

図６は、一実施形態による、トランザクションを実施する際に使用されるコンポーネントを示している。特定のトランザクションを構成するオペレーションのグループは、少なくとも当業者には「原子性（ａｔｏｍｉｃｉｔｙ）」、「一貫性（ｃｏｎｓｉｓｔｅｎｃｙ）」、「隔離性（ｉｓｏｌａｔｉｏｎ）」、および「永続性（ｄｕｒａｂｉｌｉｔｙ）」を含む頭字語「ＡＣＩＤ」として知られている特性を、全体として有している。より詳細には、トランザクションの各オペレーションからもたらされるデータの更新は、すべて不変であるか、あるいはどれも不変でなく（原子性）、トランザクションは、基礎をなすデータを一貫した状態にし（一貫性）、トランザクション更新の効果は、全体のトランザクションが不変となるまでは他の同時に実行されるオペレーションから見えず（隔離性）、またトランザクションに対する結果が決定された後は、この結果は決して変化しないことが保証される（永続性）。

【００４９】

図６のカーネルレベルのトランザクションマネージメントの実施例は、複数の装置に関与する分散トランザクションの実施例を対象としており、トランザクションに期待される「ＡＣＩＤ」特性を保持している。さらに、図６の実施例は、カーネルオブジェクトを参照しているが、この実施例は、決してカーネルオブジェクトによって実装されるトランザクションだけに限定されるものではない。より詳細には、本明細書中で説明するトランザクションは、カーネルオブジェクト以外のオブジェクトによっても実装することができ、また異なるタイプのトランザクションマネージャによっても実装することができる。

【００５０】

図６において、クライアントアプリケーション６００に対応するトランザクションは、少なくとも第１のデバイス上のトランザクションマネージャ６０５、ならびに第２のデバイス上のクライアントアプリケーション６００Ｂおよびトランザクションマネージャ６３５を利用している。クライアントアプリケーション６００Ｂは、クライアントアプリケーション６００に関連付けられている。互いに情報をやりとりするトランザクションマネージャ６０５および６３５は、全体のトランザクションおよびリソースについての状態情報を保持し、さらにクライアントアプリケーションと関連するＲＭ（ｒｅｓｏｕｒｃｅ ｍａｎａｇｅｒリソースマネージャ）の間で相互作用またはプロトコルを調整することができるカーネルオブジェクトの集合体とすることができる。

【００５１】

図６の実施例におけるＲＭ６２５およびＲＭ６３０を含めてリソースマネージャは、データを永続的状態に記憶することができる少なくとも１つの基礎をなすリソースの状態を保持する。このようなリソースの非排他的な実施例は、データベースおよびメッセージキュー（ｍｅｓｓａｇｅ ｑｕｅｕｅ）を含んでいる。図６の実施形態の例の第１のデバイスにおいて、ＲＭ６２５はリソース６２７に対応し、ＲＭ６３０はリソース６３２に対応し、第２のデバイスにおいて、ＲＭ６５５はリソース６５７に対応する。

【００５２】

図６に示すように、第１のデバイス上のトランザクションマネージャ６０５は、以下のカーネルオブジェクト：ＴＸ（ｔｒａｎｓａｃｔｉｏｎ ｏｂｊｅｃｔトランザクションオブジェクト）６１０、ＲＭＯ（ｒｅｓｏｕｒｃｅ ｍａｎａｇｅｒ ｏｂｊｅｃｔリソースマネージャオブジェクト）６１５、およびＥＮ（ｅｎｌｉｓｔｍｅｔ ｏｂｊｅｃｔエンリストメントオブジェクト）６２０を含んでおり、第２のデバイス上のトランザクションマネージャ６３５は、以下のカーネルオブジェクト：ＴＸ６４０、ＲＭＯ６４５、およびＥＮ６５０を含んでいる。ＴＸは特定のトランザクションを表し、このトランザクションに参加するプロセスによってオープンにすることができる。

【００５３】

ＲＭＯは、特定のトランザクションに参加するリソースを表す。トランザクションへのＲＭＯによる参加は、２相コミットメッセージを受信することを含んでいる。さらに、ＲＭＯは永続的であり、その結果、この対応するトランザクションマネージャは、どのトランザクションの結果が対応するＲＭに伝送されるかを知っている。代わりに、ＲＭＯが過渡的なこともあり、したがって、失敗を超えて永続的なＲＭＯを管理することなく、クライアントアプリケーションがトランザクション通知のストリームに加入することができるようになる。

【００５４】

ＥＮは、トランザクションとリソースマネージャとの間の関係を表す。リソースマネージャは、ＥＮ上にエンリストメントを作成することによってＥＮがトランザクションに参加することになることを示す。ＲＭＯが、特定のトランザクションに関して（ＰｒｅｐａｒｅやＣｏｍｍｉｔなどの）オペレーションを実施するように要求されているときには、ＲＭＯは、ＥＮを使用して参加を示す。リソースマネージャは、特定のトランザクションに関して複数のＥＮを有することもできる。

【００５５】

以下のように、カーネル環境に対して、トランザクションが正常にすべての適切なファ

イルをアップデートするようにするために実装される２相コミットプロトコルが、図６および図７の実施例を参照しながら説明される。特に、クライアントアプリケーション６００が、第１のデバイス上のトランザクションマネージャ６０５に対応するカーネルオブジェクトをオープンし、ＳＲＶ２３４（図２）が、第２のデバイス上のトランザクションマネージャ６３５に対応するカーネルオブジェクトをオープンした後に、「プリペア（ｐｒｅｐａｒｅ）」フェーズ７００が開始され、７０５において、このトランザクション中の各ＲＭには、対応するトランザクションマネージャからの「プリペア」命令が送信される。このように警告を受けて、７１０において、ＲＭはリソースデータを永続的状態にすることによってプリペアを行い、その結果、各リソース中のこのデータは、「コミットされ」または「ロールバックされる」ことが可能になる。プリペアが行われると、７１５において、ＲＭはこの対応するトランザクションマネージャのＴＸに対して確認メッセージを送信する。

【００５６】

「コミット」フェーズ７２０は、このトランザクションが解決すると実施され、それによって７２５においてこのトランザクションマネージャのＴＸは、各関連付けられたＲＭに対して「コミットされ」または「アボート／ロールバックされた」トランザクション結果を送信する。次いで、ＲＭは、この結果を関連付けられたログに記録し、この基礎となるリソースデータは、このトランザクション結果に従ってコミットされ、あるいはロールバックされる。代替的な実施形態においては、このトランザクションについてのデータが永続的でない揮発性のエンリストメントを可能とすることができ、したがってこのデータは、ログ記録もされず、また復元もされない。

【００５７】

カーネルレベル上のトランザクションマネージメントは、それだけには限定されないがＭｉｃｒｏｓｏｆｔ（登録商標）Ｗｉｎ３２（登録商標）アプリケーションプログラミングインターフェースおよびＭｉｃｒｏｓｏｆｔ（登録商標）Ｗｉｎｄｏｗｓ（登録商標）オペレーティングシステムを含めてシステムアーキテクチャに適用可能なＡＰＩ（ａｐｐｌｉｃａｔｉｏｎ ｐｒｏｇｒａｍ ｉｎｔｅｒｆａｃｅアプリケーションプログラムインターフェース）を利用することによって実装することができる。本明細書中で説明しているＡＰＩは、ハンドルベースのインターフェースを介して表示され、「ハンドル」は、ＡＰＩを対象とするオブジェクトを参照する。さらに、非同期式のオペレーションが明示的に要求されない限り、各カーネルオブジェクト上、特にＴＸおよびＲＭＯ上のオペレーションは、同期式である。さらにまた、トランザクションの異なる実施形態に対応するオペレーションは、本明細書中で説明している１つまたは複数のＡＰＩの様々な組合せによって実装することができる。すなわち、一部の実施形態では、本明細書中で説明しているＡＰＩのすべてを使用することができるが、他の実施形態では、それらの様々な組合せを使用することができる。

【００５８】

ＴＸカーネルオブジェクト上でオペレーションを実装するＡＰＩ、およびこのＡＰＩの機能の対応する記述が、以下に提供される（この関連するルーチンのさらに詳細な記述も、以下にさらに提供される）。

【００５９】

・ＰｒｅｐｒｅｐａｒｅＥｎｌｉｓｔｍｅｎｔ：「フェーズ０」の処理としても知られており、ＴＸがすべての関連するＲＭに対してプリプリペアメッセージを送信することを要求する。

【００６０】

・ＰｒｅｐａｒｅＥｎｌｉｓｔｍｅｎｔ：ＴＸがすべてのエンリストされたＲＭに対してプリペア要求を送信することを要求する。

【００６１】

・ＣｒｅａｔｅＴｒａｎｓａｃｔｉｏｎ：新しいＴＸをオープンする。

【００６２】

・ＯｐｅｎＴｒａｎｓａｃｔｉｏｎ：既存のＴＸをオープンする。
【００６３】
・ＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ：ＴＸがコミットされることを要求する。
【００６４】
・ＲｏｌｌｂａｃｋＴｒａｎｓａｃｔｉｏｎ：ＴＸがこのトランザクションをアボートまたはロールバックすることを要求する。
【００６５】
・ＳａｖｅｐｏｉｎｔＴｒａｎｓａｃｔｉｏｎ：ＴＸがこのトランザクションの状態を保存することを要求する。
【００６６】
・ＧｅｔＴｒａｎｓａｃｔｉｏｎＩｎｆｏ：このＴＸについての情報を検索する。
【００６７】
・ＳｅｔＴｒａｎｓａｃｔｉｏｎＩｎｆｏ：このＴＸについての情報を設定する。
【００６８】
ＲＭＯカーネルオブジェクト上でオペレーションを実装するために利用されるＡＰＩ、およびこのＡＰＩの機能の対応する記述が、以下に提供される（この関連するルーチンのさらに詳細な記述も、以下にさらに提供される）。
【００６９】
ＣｒｅａｔｅＲｅｓｏｕｒｃｅＭａｎａｇｅｒ：リソースを表す新しいＲＭＯを作成する。
【００７０】
ＯｐｅｎＲｅｓｏｕｒｃｅＭａｎａｇｅｒ：既存のＲＭＯをオープンする。
【００７１】
ＤｅｓｔｒｏｙＲｅｓｏｕｒｃｅＭａｎａｇｅｒ：ＲＭＯを破壊し、したがってＲＭＯを非永続的なものにする。
【００７２】
ＧｅｔＲｅｓｏｕｒｃｅＭａｎａｇｅｒＩｎｆｏ：ＲＭＯについての情報を検索する。
【００７３】
ＳｅｔＲｅｓｏｕｒｃｅＭａｎａｇｅｒＩｎｆｏ：ＲＭＯについての情報を設定する。
【００７４】
ＣｒｅａｔｅＥｎｌｉｓｔｍｅｎｔ：ＲＭＯをトランザクションに参加させ、関連した通知を検索する。
【００７５】
ＧｅｔＮｏｔｉｆｉｃａｔｉｏｎＲｅｓｏｕｒｃｅＭａｎａｇｅｒ：使用可能なＲＭ通知を照会し戻す。
【００７６】
トランザクションに参加した後、ＲＭＯカーネルオブジェクトによってＴＸカーネルオブジェクト上でオペレーションを実装するために利用されるＡＰＩ、およびこのＡＰＩの機能の対応する記述が、以下に提供される（この関連するルーチンのさらに詳細な記述も、以下にさらに提供される）。
【００７７】
・ＰｒｅＰｒｅｐａｒｅＣｏｍｐｌｅｔｅ：対応するトランザクションマネージャが要求するときに、ＲＭがプリプリペアを完了していることを示す。
【００７８】
・ＰｒｅｐａｒｅＣｏｍｐｌｅｔｅ：対応するトランザクションマネージャが要求するときに、ＲＭがトランザクションのプリペアを完了していることを示す。
【００７９】
・ＲｏｌｌｂａｃｋＣｏｍｐｌｅｔｅ：対応するトランザクションマネージャが要求す

るときに、ＲＭが実行されたトランザクション作業のロールバックを完了していることを示す。

【００８０】

・ＣｏｍｍｉｔＣｏｍｐｌｅｔｅ：対応するトランザクションマネージャが要求するときに、ＲＭがトランザクション作業のコミットを完了していることを示す。

【００８１】

残念ながら、トランザクションマネージメントを実装するために利用されるＴＸ、ＲＭＯ、およびＥＮに関連するＡＰＩは、様々なセキュリティ攻撃に対して１つまたは複数のカーネルオブジェクトをさらすこともある。例えば、悪意のあるまたは不正なＲＭが、トランザクション中にそれ自体をエンリストしてファンクション呼出しに決して応答しないことによってサービス拒否攻撃を引き起こし、あるいは、代わりにトランザクションアボートを強制することができる。したがって、図６をまた参照すると、さらなる例示の実施例は、安全なカーネルレベルの分散トランザクションを対象としている。

【００８２】

図６の実施形態の一例はさらに、各カーネルオブジェクトの少なくとも１つに対してＡＣＬ（ａｃｃｅｓｓ　ｃｏｎｔｒｏｌ　ｌｉｓｔアクセスコントロールリスト）を含むことができるセキュリティデスクリプタ（ｓｅｃｕｒｉｔｙ　ｄｅｓｃｒｉｐｔｏｒ）を適用することにより、ぜい弱なカーネルオブジェクトに対するセキュリティソリューションを提供する。

【００８３】

第１のデバイスにおいては、ＡＣＬ６６０はＴＸ６１０に適用され、ＡＣＬ６６５はＲＭＯ６１５に適用され、ＡＣＬ６７０はＥＮ６２０に適用される。第２のデバイスにおいては、ＡＣＬ６７５はＴＸ６４０に適用され、ＡＣＬ６８０はＲＭＯ６４５に適用され、ＡＣＬ６８５はＥＮ６５０に適用される。

【００８４】

ＡＣＬは、特定のユーザまたはユーザグループが特定のオブジェクト上で行使することを許可または拒否される「権利（ｒｉｇｈｔ）」を定義する。より詳細には、図８の実施例ＡＣＬ８１０に示すように、カーネルオブジェクトに適用またはアタッチされるＡＣＬは、対応するＳＩＤ（ｓｅｃｕｒｉｔｙ　ｉｄｅｎｔｉｆｉｅｒセキュリティ識別子）および対応する１組の権利を含む少なくともＡＣＥ（ａｃｃｅｓｓ　ｃｏｎｔｒｏｌ　ｅｎｔｒｙアクセス制御エントリ）を含んでいる。図８のＡＣＥエントリ１～１２は、それぞれ対応するＳＩＤ１～１２、および対応するＲＩＧＨＴ１～１２を含んでいる。

【００８５】

ＳＩＤ１～１２は、このＡＣＬが適用されるカーネルオブジェクト上でオペレーション、または一連のオペレーションを実装しようと試みることができるユーザまたはユーザグループを識別する。ＲＩＧＨＴ１～１２は、このＳＩＤによって識別されるユーザまたはユーザグループがこの各カーネルオブジェクト上で実施することができるオペレーションまたは一連のオペレーションを指定し、さらにこの識別されたユーザまたはユーザグループへのこのようなオペレーションまたは複数のオペレーションのアクセス可能性を指定する。すなわち、ＲＩＧＨＴ１～１２は、識別されたユーザまたはユーザグループが、指定されたオペレーションを実施することが許可されること、あるいは、この指定されたユーザまたはユーザグループが、指定されたオペレーションを実施することが禁止されることを示すことができる。

【００８６】

以下は、ＴＸに対して適用されるＡＣＬにおいてＲＩＧＨＴ１～１２が指定することができるオペレーションの例のリストであり、このオペレーションの機能の記述が後に続いている。ＲＩＧＨＴ１～１２はさらに、この対応するＳＩＤによって識別されるユーザまたはユーザグループに対してＴＸ上でこのオペレーションが許可または拒否されることを指定する。

【００８７】

・ＴＲＡＮＳＡＣＴＩＯＮ＿ＱＵＥＲＹ＿ＩＮＦＯＲＭＡＴＩＯＮ：ＴＸについての情報を獲得する。

【００８８】

・ＴＲＡＮＳＡＣＴＩＯＮ＿ＳＥＴ＿ＩＮＦＯＲＭＡＴＩＯＮ：ＴＸについての情報を設定する。

【００８９】

・ＴＲＡＮＳＡＣＴＩＯＮ＿ＥＮＬＩＳＴ：このトランザクション中のＴＸ上にエンリストする。

【００９０】

・ＴＲＡＮＳＡＣＴＩＯＮ＿ＣＯＭＭＩＴ：ＴＸに関連するすべてのデータアップデートを永続的にする。

【００９１】

・ＴＲＡＮＳＡＣＴＩＯＮ＿ＲＯＬＬＢＡＣＫ：ＴＸ上でこのオペレーションをアボート、すなわちロールバックする。

【００９２】

・ＴＲＡＮＳＡＣＴＩＯＮ＿ＰＲＯＰＡＧＡＴＥ：ＴＸから別のオブジェクトへとデータを伝送する。

【００９３】

・ＴＲＡＮＳＡＣＴＩＯＮ＿ＳＡＶＥＰＯＩＮＴ：このトランザクションの最新ポイントを保存する。

【００９４】

・ＴＲＡＮＳＡＣＴＩＯＮ＿ＭＡＲＳＨＡＬ：このトランザクションに関するデータを別のデバイスへと伝送する。

【００９５】

以下は、ＲＭＯに対して適用されるＡＣＬにおいてＲＩＧＨＴ１～１２が指定することができるオペレーションの例のリストであり、このオペレーションの機能の記述が後に続いている。ＲＩＧＨＴ１～１２はさらに、この対応するＳＩＤによって識別されるユーザまたはユーザグループに対してＲＭＯ上でこのオペレーションが許可または拒否されることを指定する。

【００９６】

・ＲＥＳＯＵＲＣＥＭＡＮＡＧＥＲ＿ＱＵＥＲＹ＿ＩＮＦＯＲＭＡＴＩＯＮ：ＲＭＯについての情報を獲得する。

【００９７】

・ＲＥＳＯＵＲＣＥＭＡＮＡＧＥＲ＿ＳＥＴ＿ＩＮＦＯＲＭＡＴＩＯＮ：ＲＭＯについての情報を設定する。

【００９８】

・ＲＥＳＯＵＲＣＥＭＡＮＡＧＥＲ＿ＲＥＣＯＶＥＲ：トランザクションの失敗の瞬間におけるトランザクションの状態を決定する。

【００９９】

・ＲＥＳＯＵＲＣＥＭＡＮＡＧＥＲ＿ＥＮＬＩＳＴ：トランザクションにおいてＲＭＯをエンリストする。

【０１００】

・ＲＥＳＯＵＲＣＥＭＡＮＡＧＥＲ＿ＧＥＴ＿ＮＯＴＩＦＩＣＡＴＩＯＮ：トランザクションマネージャからのトランザクションの解決に応じて通知を受信する。

【０１０１】

・ＲＥＳＯＵＲＣＥＭＡＮＡＧＥＲ＿ＲＥＧＩＳＴＥＲ＿ＰＲＯＴＯＣＯＬ：このトランザクションにおいてＲＭＯがサポートするプロトコルを登録する。

【０１０２】

・ＲＥＳＯＵＲＣＥＭＡＮＡＧＥＲ＿ＣＯＭＰＬＥＴＥ＿ＰＲＯＰＡＧＡＴＩＯＮ：トランザクションの解決に従ってリソースを設定する。

【０１０３】
以下は、ＥＮに対して適用されるＡＣＬにおいてＲＩＧＨＴ１～１２が指定することができるオペレーションの例のリストであり、このオペレーションの機能の記述が後に続いている。ＲＩＧＨＴ１～１２はさらに、この対応するＳＩＤによって識別されるユーザまたはユーザグループに対してＥＮ上でこのオペレーションが許可または拒否されることを指定する。
【０１０４】
・ＥＮＬＩＳＴＭＥＮＴ＿ＱＵＥＲＹ＿ＩＮＦＯＲＭＡＴＩＯＮ：ＥＮについての情報を獲得する。
【０１０５】
・ＥＮＬＩＳＴＭＥＮＴ＿ＳＥＴ＿ＩＮＦＯＲＭＡＴＩＯＮ：ＥＮについての情報を設定する。
【０１０６】
・ＥＮＬＩＳＴＭＥＮＴ＿ＲＥＣＯＶＥＲ：トランザクションの失敗の瞬間におけるエンリストメントの状態を決定する。
【０１０７】
・ＥＮＬＩＳＴＭＥＮＴ＿ＲＥＦＥＲＥＮＣＥ：エンリストメントキーを取得し、参照（または逆参照）する。
【０１０８】
・ＥＮＬＩＳＴＭＥＮＴ＿ＳＵＢＯＲＤＩＮＡＴＥ＿ＲＩＧＨＴＳ：このトランザクションをロールバックし、通知に応答する。
【０１０９】
・ＥＮＬＩＳＴＭＥＮＴ＿ＳＵＰＥＲＩＯＲ＿ＲＩＧＨＴＳ：トランザクション中でプリプリペアオペレーション、プリペアオペレーション、または上位ロールバックオペレーションを開始するなど、上位トランザクションマネージャが実施するはずのオペレーションを実施する。
【０１１０】
したがって、各カーネルオブジェクトＴＸ、ＲＭＯ、およびＥＮはそれぞれそれに適用されるＡＣＬを有することができる。このように、ＡＰＩがこれらのカーネルオブジェクトのそれぞれ１つの上にあるオペレーションを開始しようと試みるときに、このＡＰＩが起源とするユーザまたはユーザグループに対して、このオペレーションが許可または拒否されるかどうかを判定して、このＡＣＬに従う必要がある。
【０１１１】
より詳細には、オペレーションを実施するためのハンドルがオープンされるとき、このＡＰＩに対応するユーザまたはユーザグループはこのＡＣＬ中のＳＩＤに対して検査され、許可されたオペレーションのリストが生成され、このＡＰＩによって指定されるこのオペレーションは所与のハンドル上のこのＳＩＤに対して許可されたオペレーションに対して検査される。
【０１１２】
カーネルオブジェクトの間のトランザクションマネージメントをセキュリティ保護し、セキュリティパラメータを強要する代替的な実施形態は、このＭｉｃｒｏｓｏｆｔ（登録商標）Ｗｉｎｄｏｗｓ（登録商標）オペレーティングシステム用のセキュリティモデルによるトランザクションに参加することができるカーネルオブジェクトにセキュリティデスクリプタを適用する工程を含んでいる。
【０１１３】
上述のように、これらのＡＰＩは、ハンドルベースのインターフェースとして表示され、このインターフェースを利用してこのセキュリティモデルを実装する。以上でリストアップしたＴＸカーネルオブジェクト上でオペレーションを実装するこれらのＡＰＩのより詳細な記述が、以下に含まれている。これらの記述は、このルーチン、対応する引数、およびリターン値の記述を含んでいる。

【０１１４】
PreprepareEnlistment
（IN PHANDLE TransactionHandle；
IN PHANDLE ResourceManagerHandle）。
【０１１５】
・このルーチンは、すべての関連するＲＭに対してプリプリペア要求（Ｐｒｅ－Ｐｒｅｐａｒｅ ｒｅｑｕｅｓｔ）を発行することによってトランザクションが「プリプリペアされる」ことを要求する。プリプリペアは、このトランザクションが、下流のＲＭがもはや変更を受け入れることができないというプリペア状態（Ｐｒｅｐａｒｅｄ ｓｔａｔｅ）に入る以前に、キャッシュに似た特性を有するＲＭにそのキャッシュを、可能であれば他のＲＭに対してフラッシュする機会を許可する。
【０１１６】
・このルーチンが呼び出されず、トランザクション参加者がフェーズ０の処理を要求している場合、プリペア（Ｐｒｅｐａｒｅ）が受信されるときに要求されるように、プリプリペア要求が発行される。しかし、ＰｒｅｐｒｅｐａｒｅＥｎｌｉｓｔｍｅｎｔが存在しない場合には、キャッシュに似たＲＭを含む一部の構成では、分散シナリオにおいて、不必要なトランザクションロールバックが引き起こされることもある。
【０１１７】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：プリプリペアすべきトランザクションを示すハンドルを与える。
【０１１８】
ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅ：このトランザクションをプリプリペアしている上位－ＴＭ／ＣＲＭに対してハンドルを与える。この上位－ＴＭ／ＣＲＭだけが、このトランザクション上でＰｒｅｐａｒｅＥｎｌｉｓｔｍｅｎｔ、ＳｕｐｅｒｉｏｒＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ、およびＳｕｐｅｒｉｏｒＲｏｌｌｂａｃｋＴｒａｎｓａｃｔｉｏｎを呼び出すことが可能になる。
【０１１９】
・リターン値：
【０１２０】
【表８】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES
STATUS_TM_TOO_LATE

【０１２１】
PrepareEnlistment
（IN PHANDLE TransactionHandle、
IN PHANDLE ResourceManagerHandle）；
・このルーチンは、すべてのその関連付けられたＲｅｓｏｕｒｃｅＭａｎａｇｅｒに対してプリペア要求（Ｐｒｅｐａｒｅ ｒｅｑｕｅｓｔ）を発行することによって、トランザクションが「プリペアされる」ことを要求する。この要求は、２相コミットプロトコルを開始する。
【０１２２】
・ＰｒｅｐａｒｅＥｎｌｉｓｔｍｅｎｔを発行するトランザクション参加者は、トラン

ザクションオブジェクトをシステムまたはアプリケーションがクラッシュしても生き残ることになる永続的状態にする。このような参加者は、どのようなタイプの失敗の後にもこのトランザクション上でリカバリ（ｒｅｃｏｖｅｒｙ）を実施して、結果を送付する。この要件を満たすのに失敗すると、リソースのリーク、ならびに矛盾したトランザクションの結果がもたらされることもある。

【０１２３】

・引数：

ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：プリペアすべきトランザクションについてのハンドルを与える。

【０１２４】

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅ：このトランザクションをプリペアしているＴＭに対して、ハンドルを与える。このトランザクションが（ＰｒｅｐｒｅｐａｒｅＥｎｌｉｓｔｍｅｎｔに対するコールを介して）プリプリペアされている場合には、ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅは、ＰｒｅｐｒｅｐａｒｅＥｎｌｉｓｔｍｅｎｔに対する呼出し中で使用された上位－ＴＭ／ＣＲＭとマッチングする。さらに、このＡＰＩをコールする上位－ＴＭ／ＣＲＭだけが、このトランザクション上でＳｕｐｅｒｉｏｒＣｏｍｍｉｔｔｒａｎｓａｃｔｉｏｎおよびＳｕｐｅｒｉｏｒＲｏｌｌｂａｃｋＴｒａｎｓａｃｔｉｏｎを呼び出すことが許可されることになる。

【０１２５】

・リターン値：

【０１２６】

【表９】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES
STATUS_TM_TOO_LATE
STATUS_RM_NOT_RECOVERABLE

【０１２７】

```
CreateTransaction
   (OUT PHANDLE              TransactionHandle、
    IN ULONG                 DesiredAccess
OPTIONAL；
    IN POBJECT_ATTRIBUTES    ObjectAttributes
OPTIONAL；
    IN ULONG                 CreateOptions
OPTIONAL；
    IN PHANDLE               ResourceManagerHandle
OPTIONAL；
    IN NOTIFICATION_MASK     NotificationMask
OPTIONAL；
    IN LPVOID                TransactionKey
OPTIONAL）。
```

【０１２８】

・このルーチンは、新しいトランザクションオブジェクトを作成し、この新しいオブジ

ェクトにハンドルを戻す。
【０１２９】
・代わりに（ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅパラメータが指定される場合）、このルーチンは、それが正常に作成された後に、このトランザクション上で「ジョイン（Ｊｏｉｎ）」オペレーションを実施する。
【０１３０】
クライアントは、ＣｌｏｓｅＨａｎｄｌｅＡＰＩを使用してこのトランザクションハンドルをクローズする。この最後のトランザクションハンドルが、いずれかがこのトランザクション上でＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎを呼び出すことなくクローズする場合には、このトランザクションは、暗黙的にロールバックされる。
【０１３１】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：この新しいトランザクションに対するハンドルを受信することになるロケーションにポインタを与える。
【０１３２】
ＤｅｓｉｒｅｄＡｃｃｅｓｓ：この所望のアクセスレベルを指定するマスクを与える。この有効なアクセスマスクの選択肢には、以下のものがある。：
ＳＹＮＣＨＲＯＮＩＺＥ　このハンドル上で同期オペレーションを実施することができる。
【０１３３】
ＴＲＡＮＳＡＣＴＩＯＮ＿ＣＯＭＭＩＴ　このハンドルを使用してトランザクションをコミットすることができる。
【０１３４】
ＴＲＡＮＳＡＣＴＩＯＮ＿ＰＲＥＰＡＲＥ　このハンドルを使用してトランザクションをコミットすることができる。
【０１３５】
ＴＲＡＮＳＡＣＴＩＯＮ＿ＲＯＬＬＢＡＣＫ　このハンドルを使用してトランザクションをアボートすることができる。
【０１３６】
ＴＲＡＮＳＡＣＴＩＯＮ＿ＳＡＶＥＰＯＩＮＴ　このハンドルを使用してこのトランザクションについてのセーブポイントを作成することができる。
【０１３７】
ＴＲＡＮＳＡＣＴＩＯＮ＿ＪＯＩＮ　このハンドルを使用してこのトランザクションをＲＭとして加えることができる。
【０１３８】
ＴＲＡＮＳＡＣＴＩＯＮ＿ＲＥＡＤ＿ＡＴＴＲＩＢＵＴＥＳ　トランザクションに関連する属性を読み取ることができる。
【０１３９】
ＴＲＡＮＳＡＣＴＩＯＮ＿ＷＲＩＴＥ＿ＡＴＴＲＩＢＵＴＥＳ　トランザクションに関連する属性を書き込むことができる。
【０１４０】
ＯｂｊｅｃｔＡｔｔｒｉｂｕｔｅｓ：オプションのオブジェクト属性構造に、ポインタを与える。
【０１４１】
ＣｒｅａｔｅＯｐｔｉｏｎｓ　オプションのトランザクションフラグを与える。有効な作成フラグ選択肢は以下を含んでいる。
【０１４２】
ＴＲＡＮＳＡＣＴＩＯＮ＿ＣＲＥＡＴＥ＿ＰＲＥＳＵＭＥＤ＿ＮＯＴＨＩＮＧ「何も想定しない（ｐｒｅｓｕｍｅｄ　ｎｏｔｈｉｎｇ）」トランザクションを作成する。

【０１４３】
ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅ：指定されたトランザクションについての通知を受信するＲｅｓｏｕｒｃｅＭａｎａｇｅｒにハンドルを与える。
【０１４４】
ＮｏｔｉｆｉｃａｔｉｏｎＭａｓｋ：このＲｅｓｏｕｒｃｅＭａｎａｇｅｒがこのトランザクションに関して受信したい通知を指定する。
【０１４５】
ＴｒａｎｓａｃｔｉｏｎＫｅｙ：このＲＭがこのトランザクションについての任意の通知とともに受信をしたい不明瞭なポインタ値を指定する。このＲＭは、このポインタ値を使用して通知をこのＲＭのアドレス空間中のあるオブジェクトに関連づけることができ、それによって通知が行われるたびにルックアップを実行する必要をなくすることができる。
【０１４６】
・リターン値：
【０１４７】
【表１０】

STATUS_SUCCESS
STATUS_INVALID_PARAMETER
STATUS_OBJECT_NAME_COLLISION
STATUS_OBJECT_NAME_INVALID
STATUS_PRIVILEGE_NOT_HELD
STATUS_INSUFFICIENT_RESOURCES

【０１４８】

```
OpenTransaction
   (OUT PHANDLE              TransactionHandle、
    IN ACCESS_MASK           DesiredAccess、
    IN POBJECT_ATTRIBUTES    ObjectAttributes、
    IN PHANDLE               ResourceManagerHandle
optional、
    IN NOTIFICATION_MASK     NotificationMask
optional、
    IN LPVOID                TransactionKey
optional）；
```

・このルーチンは、既存のトランザクションオブジェクトを調べ、ハンドルをこのトランザクションに戻す。この呼出し元は、ＯｂｊｅｃｔＡｔｔｒｉｂｕｔｅｓのＯｂｊｅｃｔＮａｍｅフィールド中においてＧＵＩＤのストリング表現を指定する。
【０１４９】
・代わりに（このＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅパラメータが指定される場合）、このルーチンはまた、それがオープンされた後にこのトランザクション上で「ジョイン」オペレーションを実施する。
【０１５０】
・クライアントは、ＣｌｏｓｅＨａｎｄｌｅＡＰＩを使用してこのトランザクションハンドルをクローズする。この最後のトランザクションハンドルが、どれかがこのトランザクション上でＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎをコールすることなくクローズする場合には、このトランザクションは、暗黙的にロールバックされたトランザクションである

。
【０１５１】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：このオープンオペレーションが成功する場合に、このトランザクションに対するハンドルを受信することになるロケーションにポインタを与える。
【０１５２】
ＤｅｓｉｒｅｄＡｃｃｅｓｓ：この所望のアクセスレベルを指定するマスクを与える。
【０１５３】
ＯｂｊｅｃｔＡｔｔｒｉｂｕｔｅｓ：オプションのオブジェクト属性構造にポインタを与える。
【０１５４】
ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅ：この指定されたトランザクションについての通知を受信するＲｅｓｏｕｒｃｅＭａｎａｇｅｒにハンドルを与える。
【０１５５】
ＮｏｔｉｆｉｃａｔｉｏｎＭａｓｋ：このＲｅｓｏｕｒｃｅＭａｎａｇｅｒがこのトランザクションに関して受信することができる通知を指定する。
【０１５６】
ＴｒａｎｓａｃｔｉｏｎＫｅｙ：このＲＭがこのトランザクションについての任意の通知とともに受信したい不明瞭なポインタ値をオプション的に指定する。このＲＭは、このポインタ値を使用して通知をこのＲＭのアドレス空間中のあるオブジェクトに関連づけることができ、それによって通知が行われるたびにルックアップを実行する必要をなくすることができる。
【０１５７】
・リターン値：
【０１５８】
【表１１】

STATUS_SUCCESS
STATUS_INVALID_PARAMETER
STATUS_OBJECT_NAME_INVALID
STATUS_OBJECT_NAME_NOT_FOUND
STATUS_OBJECT_PATH_SYNTAX_BAD
STATUS_PRIVILEGE_NOT_HELD
STATUS_INSUFFICIENT_RESOURCES

【０１５９】
```
CommitTransaction
   (IN PHANDLE    TransactionHandle
    IN ULONG      CommitOptions    Optional) ;
```
・このルーチンは、ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅに関連するこのトランザクションがコミットされることを要求する。オープンまたは作成されている任意のトランザクションハンドルは、Ｔｒａｎｓａｃｔｉｏｎ＿Ｃｏｍｍｉｔ　Ｄｅｓｉｒｅｄ　Ａｃｃｅｓｓを用いてコミットすることができる。トランザクションの作成者だけがそれをコミットすることが可能であることを提示する制約は、存在していないからである。

【０１６０】
・問題のトランザクションが、以前にＰｒｅｐａｒｅＥｎｌｉｓｔｍｅｎｔ要求を発行していない場合に、２相コミットプロトコルは、すべてのエンリストされたＲＭ上で開始することができる。この呼出しは、このクライアントによって発行されている１相コミット要求と考えることができる。
【０１６１】
・このトランザクションが以前にＰｒｅｐａｒｅＥｎｌｉｓｔｍｅｎｔを介してプリペアされている場合には、このルーティング（ｒｏｕｔｉｎｇ）は、呼び出されない。ＰｒｅｐａｒｅＥｎｌｉｓｔｍｅｎｔを呼び出したＲＭだけが、このルーチンＳｕｐｅｒｉｏｒＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎを使用して、このトランザクション状態を解決することができる。
【０１６２】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：コミットすべきトランザクションを示すハンドルを与える。
【０１６３】
ＣｏｍｍｉｔＯｐｔｉｏｎｓ：ＣＯＭＭＩＴ＿ＲＥＴＡＩＮＩＮＧトランザクションがコミットされることになる。
【０１６４】
・リターン値：
【０１６５】
【表１２】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES
STATUS_TM_TRANSACTION_ABORTED

【０１６６】

```
RollbackTransaction
   （IN PHANDLE           TransactionHandle、
    IN SAVEPOINT          SavePoint         Optional、
    IN ROLLBACK_REASON    RollbackReason    Optional）；
```

・このルーチンは、ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅに関連するこのトランザクションをロールバックすることを要求する。このオプションのＳａｖｅＰｏｉｎｔが指定され、有効なセーブポイントである場合、このロールバックは、部分的なロールバックでもよい。ＮＵＬＬのＳａｖｅＰｏｉｎｔ引数は、このトランザクションが完全にロールバックまたはアボートされるべきことを示す。オプションのＲｏｌｌｂａｃｋＲｅａｓｏｎ構造を与えることができる。これは、トランザクションオブジェクト中に保持されることになり、ＧｅｔＩｎｆｏｒｍａｔｉｏｎＴｒａｎｓａｃｔｉｏｎに対する呼出しを介して関心のあるトランザクション参加者は検索することができる。
【０１６７】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：ロールバックすべきトランザクションを示すハンドルを与える。
【０１６８】
ＳａｖｅＰｏｉｎｔ：どこまでトランザクションの状態をロールバックすべきかを示

すＳａｖｅＰｏｉｎｔネームを与える。
【０１６９】
ＲｏｌｌｂａｃｋＲｅａｓｏｎ：ロールバックリーズン（ｒｏｌｌｂａｃｋ　ｒｅａｓｏｎ）を与える。
【０１７０】
・リターン値：
【０１７１】
【表１３】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES
STATUS_TM_TRANSACTION_COMMITTED

【０１７２】
SavePointTransaction
（IN PHANDLE TransactionHandle、
IN ULONG SavePointFlags Optional、
OUT LPSAVEPOINT SavePoint）；
・このルーチンは、「セーブポイント」がＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅに関連付けられたトランザクションについて生成されることを要求する。このセーブポイントは、後続のロールバック要求に対するターゲットとして使用される。
【０１７３】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：セーブポイントを確立すべき対象のトランザクションを示すハンドルを与える。
【０１７４】
ＳａｖｅＰｏｉｎｔＦｌａｇｓ：このセーブポイントの生成に影響を及ぼす１組のフラグをオプション的に与える。
【０１７５】
ＳａｖｅＰｏｉｎｔ：セーブポイント識別子が記憶されるロケーションにポインタを与える。
【０１７６】
・リターン値：
【０１７７】
【表１４】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES
STATUS_TM_TRANSACTION_COMMITTED
STATUS_TM_TRANSACTION_ABORTED

【０１７８】
QueryInformationTransaction
(IN HANDLE TransactionHandle、
IN TRANSACTION_INFORMATION_CLASS
TransactionInformationClass、
OUT PVOID
TransactionInformation、
IN ULONG
TransactionInformationLength、
OUT PULONG ReturnLength Optional);
・このルーチンは、ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅによって表されるトランザクションオブジェクトについての要求された情報を戻す。
【０１７９】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：情報が要求されているトランザクションを示すハンドルを与える。
【０１８０】
ＴｒａｎｓａｃｔｉｏｎＩｎｆｏｒｍａｔｉｏｎＣｌａｓｓ：このトランザクションオブジェクトについてのどのクラスの情報が要求されているかを示す。
【０１８１】
ＴｒａｎｓａｃｔｉｏｎＩｎｆｏｒｍａｔｉｏｎ：要求されているこのトランザクション情報が記憶されるバッファに対してポインタを与える。
【０１８２】
ＴｒａｎｓａｃｔｉｏｎＩｎｆｏｒｍａｔｉｏｎＬｅｎｇｔｈ：ＴｒａｎｓａｃｔｉｏｎＩｎｆｏｒｍａｔｉｏｎによってポイントされるバッファの長さを示す。
【０１８３】
ＲｅｔｕｒｎＬｅｎｇｔｈ：このＴｒａｎｓａｃｔｉｏｎＩｎｆｏｒｍａｔｉｏｎバッファに書き込まれる情報の長さを受信することになるロケーションにポインタを与える。
【０１８４】
・リターン値：
【０１８５】
【表１５】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES
STATUS_INVALID_INFO_CLASS
STATUS_INFO_LENGTH_MISMATCH

【０１８６】
SetInformationTransaction
(IN HANDLE TransactionHandle、
IN TRANSACTION_INFORMATION_CLASS
TransactionInformationClass、 IN PVOID
TransactionInformation、

IN ULONG
TransactionInformationLength）；
・このルーチンはＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅによって表されるこのトランザクションオブジェクトについてこの要求された情報を設定する。
【０１８７】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：どの情報が修正されることになるかをこのトランザクションに示すハンドルを与える。
【０１８８】
ＴｒａｎｓａｃｔｉｏｎＩｎｆｏｒｍａｔｉｏｎＣｌａｓｓ：このトランザクションオブジェクトについてのどのクラスの情報が要求されているかを示す。
【０１８９】
ＴｒａｎｓａｃｔｉｏｎＩｎｆｏｒｍａｔｉｏｎ：要求されるこのトランザクション情報が記憶されるバッファにポインタを与える。
【０１９０】
ＴｒａｎｓａｃｔｉｏｎＩｎｆｏｒｍａｔｉｏｎＬｅｎｇｔｈ：ＴｒａｎｓａｃｔｉｏｎＩｎｆｏｒｍａｔｉｏｎによってポイントされるバッファの長さを示す。
【０１９１】
ＲｅｔｕｒｎＬｅｎｇｔｈ：このＴｒａｎｓａｃｔｉｏｎＩｎｆｏｒｍａｔｉｏｎバッファに書き込まれる情報の長さを受信することになるロケーションにポインタを与える。
【０１９２】
・リターン値：
【０１９３】
【表１６】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES
STATUS_INVALID_INFO_CLASS
STATUS_INFO_LENGTH_MISMATCH

【０１９４】
以下は、以上でリストアップした、ＲＭＯカーネルオブジェクト上でオペレーションを実装するＡＰＩのより詳細な記述を含んでいる。これらの記述は、このルーチン、対応する引数、およびリターン値の記述を含んでいる。
【０１９５】

```
CreateResourceMnager
   (OUT PHANDLE                  ResourceManagerHandle、
    IN ACCESS_MASK               DesiredAccess
    Optional、
    IN POBJECT_ATTRIBUTES        ObjectAttributes、
    IN ULONG                     CreateOptions
    Optional、
    IN RM_NOTIFICATION_ROUTINE   NotificationRoutine
```

Optional）；
・このルーチンは、リソースを表す新しいＲｅｓｏｕｒｃｅＭａｎａｇｅｒオブジェクトを作成する。
【０１９６】
・ＲｅｓｏｕｒｃｅＭａｎａｇｅｒオブジェクトは、このＲＭが参加したという、トランザクションに関するＴＭ通知についてのエンドポイントとしての役割も果たす。ＲＭは、ＧｅｔＮｏｔｉｆｉｃａｔｉｏｎＲｅｓｏｕｒｃｅＭａｎａｇｅｒを呼び出すことによってこれらの通知を要求する。
【０１９７】
・ＲｅｓｏｕｒｃｅＭａｎａｇｅｒは、通常は永続的なオブジェクトであり、すなわちこのオブジェクトは、（システムまたはＲＭの）あらゆる失敗の後に再オープンされ、リカバリを実施する必要がある。ＲｅｓｏｕｒｃｅＭａｎａｇｅｒオブジェクトの過渡的なバージョンは、このオプションＲＥＳＯＵＲＣＥＭＡＮＡＧＥＲ＿ＮＯ＿ＲＥＣＯＶＥＲＹを指定することによって作成することができる。過渡的なＲＭは、リカバリを実施するように義務づけられておらず、また許可されてもいない。このリカバリ不可能なＲＭオプションにより、この十分複雑なログ記録をとるプリペアを実装する必要なしに、またリカバリを実施することもなしに、アプリケーションまたはＲＭはトランザクションの進行（例えば、ＰＲＥＰＲＥＰＡＲＥ、ＰＲＥＰＡＲＥ、ＣＯＭＭＩＴ）についての通知を受信できるようになる。
【０１９８】
・引数：
ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅ：この新しいＲｅｓｏｕｒｃｅＭａｎａｇｅｒに対するハンドルを受信することになるロケーションにポインタを与える。
【０１９９】
ＤｅｓｉｒｅｄＡｃｃｅｓｓ：所望のアクセスレベルを指定するマスクを与える。有効なアクセスマスクの選択肢を以下に示す。：
ＳＹＮＣＨＲＯＮＩＺＥ：ハンドル上でオペレーションを同期させる。
【０２００】
ＲＥＳＯＵＲＣＥ　ＭＡＮＡＧＥＲ＿ＤＥＳＴＲＯＹ：このリソースマネージャを破壊する。
【０２０１】
ＲＥＳＯＵＲＣＥ　ＭＡＮＡＧＥＲ＿ＲＥＡＤ＿ＡＴＴＲＩＢＵＴＥＳ：リソースマネージャに関連する属性を読み取る。
【０２０２】
ＲＥＳＯＵＲＣＥ　ＭＡＮＡＧＥＲ＿ＷＲＩＴＥ＿ＡＴＴＲＩＢＵＴＥＳ：リソースマネージャに関連する属性を書き込む。
【０２０３】
ＯｂｊｅｃｔＡｔｔｒｉｂｕｔｅｓ：この新しいＲＭオブジェクトについての属性を指定する。この属性は、ＲＭネームを含んでいる。
【０２０４】
ＣｒｅａｔｅＯｐｔｉｏｎｓ：この作成されたオブジェクトについてのオプションを指定する。
【０２０５】
ＲＥＳＯＵＣＥＭＡＮＡＧＥＲ＿ＮＯ＿ＲＥＣＯＶＥＲＹ：ＲｅｓｏｕｒｃｅＭａｎａｇｅｒオブジェクトは、非永続的であり、リカバリを実施しない。
【０２０６】
ＲＥＳＯＵＣＥＭＡＮＡＧＥＲ＿ＣＯＭＭＵＮＩＣＡＴＩＯＮ：ＲｅｓｏｕｒｃｅＭａｎａｇｅｒは、他のコンピュータに対してどのように通信を行うかを知っている。ＲｅｓｏｕｒｃｅＭａｎａｇｅｒを使用してトランザクションをマーシャリングまたはアンマーシャリングする（ｕｎｍａｒｓｈａｌｌ）ことができる。

【０２０７】
ＲＥＳＯＵＲＣＥＭＡＮＡＧＥＲ＿ＣＬＵＳＴＥＲ＿ＲＥＣＯＶＥＲＹ：ＲｅｓｏｕｒｃｅＭａｎａｇｅｒは、失敗している可能性のあるファイルをログ記録した結果をどのようにして読み取り、このクラスタ中の他のノードに送付するべきかを知っている。ＲｅｓｏｕｒｃｅＭａｎａｇｅｒを使用してクラスタ中のトランザクションをリカバリすることができる。
【０２０８】
ＮｏｔｉｆｉｃａｔｉｏｎＲｏｕｔｉｎｅ：通知がこのＲｅｓｏｕｒｃｅＭａｎａｇｅｒにとって使用可能であるときに呼び出すべき通知ルーチンを指定する。
【０２０９】
・リターン値：
【０２１０】
【表１７】

STATUS_SUCCESS
STATUS_INVALID_PARAMETER
STATUS_OBJECT_NAME_COLLISION
STATUS_OBJECT_NAME_INVALID
STATUS_PRIVILEGE_NOT_HELD
STATUS_INSUFFICIENT_RESOURCES

【０２１１】

```
OpenResourceManager
   (OUT PHANDLE                 ResourceManagerHandle、
    IN ACCESS_MASK              DesiredAccess
    Optional、
    IN POBJECT_ATTRIBUTES       ObjectAttributes、
    IN ULONG                    OpenOptions
    Optional、
    IN RM_NOTIFICATION_ROUTINE      NotificationRoutine、
    Optional）。
```

【０２１２】
・このルーチンは、名前によって既存のＲｅｓｏｕｒｃｅＭａｎａｇｅｒオブジェクトをオープンする。ターゲットのＲｅｓｏｕｒｃｅＭａｎａｇｅｒオブジェクトが永続的であるが、現在オープンされていない場合、このオブジェクトは最初に「リカバリする」状態にあり、リカバリされる必要がある。リカバリが完了した後に、ＲｅｃｏｖｅｒｙＣｏｍｐｌｅｔｅＲｅｓｏｕｒｃｅＭａｎａｇｅｒを呼び出す必要がある。
【０２１３】
・引数：
ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅ：既存のＲｅｓｏｕｒｃｅＭａｎａｇｅｒオブジェクトに対するハンドルを受信することになるロケーションにポインタを与える。
【０２１４】
ＤｅｓｉｒｅｄＡｃｃｅｓｓ：このオブジェクトに対する所望のアクセスを指定するマスクを与える。
【０２１５】
ＯｂｊｅｃｔＡｔｔｒｉｂｕｔｅｓ：この新しいＲＭオブジェクトについての属性を

指定する。
【０２１６】
　ＯｐｅｎＯｐｔｉｏｎｓ：このオブジェクトについてのオプションを指定する。有効なオプションは、以下を含んでいる。
【０２１７】
　ＲＥＳＯＵＲＣＥ＿ＭＡＮＡＧＥＲ＿ＤＥＴＡＩＬＥＤ＿ＲＥＣＯＶＥＲＹ＿ＮＯＴＩＦＩＣＡＴＩＯＮＳ：このリソースマネージャは、通常のリカバリ通知の代わりに、（通信エンドポイントについての追加の情報を伴う）詳細なリカバリ通知を受信する。
【０２１８】
　ＮｏｔｉｆｉｃａｔｉｏｎＲｏｕｔｉｎｅ：通知が、このＲｅｓｏｕｒｃｅＭａｎａｇｅｒにとって使用可能であるときに、呼び出されることになる通知ルーチンを指定する。
【０２１９】
・リターン値：
【０２２０】
【表１８】

STATUS_SUCCESS
STATUS_INVALID_PARAMETER
STATUS_OBJECT_NAME_INVALID
STATUS_OBJECT_NAME_NOT_FOUND
STATUS_OBJECT_PATH_SYNTAX_BAD
STATUS_PRIVILEGE_NOT_HELD
STATUS_INSUFFICIENT_RESOURCES.

【０２２１】
DestroyResourceManager
　（IN PHANDLE　ResourceManagerHandle）；
・このルーチンは、ＲｅｓｏｕｒｃｅＭａｎａｇｅｒオブジェクトを破壊し、それがもはや永続的でないようにする。
【０２２２】
・引数：
　ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅ：破壊すべきＲｅｓｏｕｒｃｅＭａｎａｇｅｒオブジェクトを指示するハンドルを与える。
【０２２３】
・リターン値：
【０２２４】

【表１９】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_TM_NEEDS_RECOVERY.

【０２２５】

```
QueryInformationResourceManager
   (IN HANDLE                          ResourceManagerHandle、
    IN RESOURCEMANAGER_INFORMATION_CLASS
    ResourceManagerInformationClass、
    OUT PVOID                          ResourceManagerInformation、
    IN ULONG
    ResourceManagerInformationLength、
    OUT PULONG                          ReturnLength Optional)。
```

【０２２６】

・このルーチンは、ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅによって表されるＲＭＯについてのこの要求された情報を戻す。

【０２２７】

・引数：

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅ：情報が要求されているこのＲｅｓｏｕｒｃｅＭａｎａｇｅｒを示すハンドルを与える。

【０２２８】

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＩｎｆｏｒｍａｔｉｏｎＣｌａｓｓ：このＲｅｓｏｕｒｃｅＭａｎａｇｅｒオブジェクトについてのどのクラスの情報が要求されているかを示す。

【０２２９】

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＩｎｆｏｒｍａｔｉｏｎ：要求されるこのＲｅｓｏｕｒｃｅＭａｎａｇｅｒ情報が記憶されることになるバッファにポインタを与える。

【０２３０】

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＩｎｆｏｒｍａｔｉｏｎＬｅｎｇｔｈ：ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＩｎｆｏｒｍａｔｉｏｎによってポイントされるバッファの長さを示す。

【０２３１】

ＲｅｔｕｒｎＬｅｎｇｔｈ：ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＩｎｆｏｒｍａｔｉｏｎバッファに書き込まれる情報の長さを受信するロケーションにポインタを与える。

【０２３２】

・リターン値：

【０２３３】

【表２０】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_INVALID_INFO_CLASS

STATUS_INFO_LENGTH_MISMATCH

【０２３４】

SetInformationResourceManager
(IN HANDLE ResourceManagerHandle、
IN RESOURCEMANAGER_INFORMATION_CLASS
ResourceManagerInformationClass、
IN PVOID ResourceManagerInformation、
IN ULONG ResourceManagerInformationLength) ;

・このルーチンはＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅによって表されるＲＭＯについてのこの要求された情報を設定する。

【０２３５】

・引数：

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅ：情報が修正されているこのＲｅｓｏｕｒｃｅＭａｎａｇｅｒを示すハンドルを与える。

【０２３６】

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＩｎｆｏｒｍａｔｉｏｎＣｌａｓｓ：このＲｅｓｏｕｒｃｅＭａｎａｇｅｒオブジェクトについてのどのクラスの情報が要求されているかを示す。

【０２３７】

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＩｎｆｏｒｍａｔｉｏｎ：要求されるこのＲｅｓｏｕｒｃｅＭａｎａｇｅｒ情報が記憶されるバッファにポインタを与える。

【０２３８】

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＩｎｆｏｒｍａｔｉｏｎＬｅｎｇｔｈ：ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＩｎｆｏｒｍａｔｉｏｎによってポイントされるバッファの長さを示す。

【０２３９】

・リターン値：

【０２４０】

【表２１】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_INVALID_INFO_CLASS

STATUS_INFO_LENGTH_MISMATCH.

【０２４１】

```
CreateEnlistment
   (IN PHANDLE              ResourceManagerHandle、
    IN PHANDLE              TransactionHandle、
    IN NOTIFICATION_MASK    NotificationMask、
    IN LPVOID               TransactionKey Optional);
```

・このルーチンは、ＲＭＯを特定のトランザクションに結合（ｊｏｉｎｔ）させ、それに関連した通知を受信する。

【０２４２】

・このＣｒｅａｔｅＥｎｌｉｓｔｍｅｎｔ呼出しは、べき等（ｉｄｅｍｐｏｔｅｎｔ）であり、ＲＭはこのルーチンを複数回呼び出して、そのＮｏｔｉｆｉｃａｔｉｏｎＭａｓｋまたはＴｒａｎｓａｃｔｉｏｎＫｅｙを変更することができる。ＣｒｅａｔｅＥｎｌｉｓｔｍｅｎｔに対する後続の呼出しは、このトランザクション上で新しいエンリストメントを作成せずに通知マスクおよびトランザクションキーを置換する。

【０２４３】

・ＮｏｔｉｆｉｃａｔｉｏｎＭａｓｋを使用して通知が複数回受信されることを要求することができる。例えば、ＰＲＥＰＲＥＰＡＲＥ通知を受信するＲＭは、ＪｏｉｎＴｒａｎｓａｃｔｉｏｎを呼び出し、ＰＲＥＰＲＥＰＡＲＥフラグを指定することによって別の通知を要求することができる。したがって、ＲＭは、複数のＰＲＥＰＲＥＰＡＲＥ要求を受信することができる。このような要求は拒絶されることもあり、このトランザクションが、この要求された通知が受信されているはずのポイントを以前に進んでいる場合には、これは望ましいこともある。例えば、あるＲＭが既にＰＡＲＥに通知されているときに、ＰＲＥＰＲＥＰＡＲＥを要求することは認められない。

【０２４４】

・引数：

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅ：ＲＭにハンドルを供給してこの指定されたトランザクションについての通知を受信する。

【０２４５】

ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：このＲＭが参加することを望むトランザクションに対してハンドルを与える。

【０２４６】

ＮｏｔｉｆｉｃａｔｉｏｎＭａｓｋ：ＲＭがこのトランザクションに関して受信したいと思う通知を指定する。有効なマスクは、以下のようになっており、これらは一緒にＯＲをとることもできる。

【０２４７】

ＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＹ＿ＭＡＳＫ＿ＲＭ：ＲＭ（ＰＲＥＰＡＲＥ、ＣＯＭＭＩＴ、ＲＯＬＬＢＡＣＫ、ＳＡＶＥＰＯＩＮＴ）が望む一般の通知。

【０２４８】

ＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＹ＿ＭＡＳＫ＿ＣＲＭ：ＣＲＭまたは上位ＴＭ（ＰｒｅＰｒｅｐａｒｅ＿Ｃｏｍｐｌｅｔｅ、ＰｒｅｐａｒｅＣｏｍｐｌｅｔｅ、ＣｏｍｍｉｔＣｏｍｐｌｅｔｅ、ＲｏｌｌｂａｃｋＣｏｍｐｌｅｔｅ、ＳａｖｅｂａｃｋＣｏｍｐｌｅｔｅ）が望む一般の通知。

【０２４９】

ＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＹ＿ＰＲＥＰＲＥＰＡＲＥ：ＰｒｅＰｒｅｐａｒｅに対する通知。

【０２５０】

ＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＹ＿ＰＲＥＰＡＲＥ：ＰＲＥＰＡＲＥに対する通知。

【０２５１】

ＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＹ＿ＣＯＭＭＩＴ：ＣＯＭＭＩＴに対する通知。

【０２５２】

ＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＹ＿ＲＯＬＬＢＡＣＫ：ＲＯＬＬＢＡＣＫに対する通知。

【０２５３】

ＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＹ＿ＰＲＥＰＲＥＰＡＲＥ＿ＣＯＭＰＬＥＴＥ：ＰＲＥＰＲＥＰＡＲＥが完了しているという通知。

【０２５４】

ＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＹ＿ＰＲＥＰＡＲＥ＿ＣＯＭＰＬＥＴＥ：ＰＲＥＰＡＲＥが完了しているという通知。

【０２５５】

ＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＹ＿ＣＯＭＭＩＴ＿ＣＯＭＰＬＥＴＥ：ＣＯＭＭＩＴが完了しているという通知。

【０２５６】

ＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＹ＿ＲＯＬＬＢＡＣＫ＿ＣＯＭＰＬＥＴＥ：ＲＯＬＬＢＡＣＫが完了しているという通知。

【０２５７】

ＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＹ＿ＳＡＶＥＰＯＩＮＴ＿ＣＯＭＰＬＥＴＥ：ＳＡＶＥＰＯＩＮＴが完了しているという通知。

【０２５８】

ＴｒａｎｓａｃｔｉｏｎＫｅｙ：ＲＭがこのトランザクションについての任意の通知とともに受信したいと思う不明瞭なポインタ値を指定する。このＲＭは、このポインタ値を使用して通知をこのＲＭアドレス空間中のあるオブジェクトに関連づけ、それによって通知が行われるたびにルックアップを実施する必要をなくすることができる。

【０２５９】

・リターン値：

【０２６０】

【表２２】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_PARAMETER

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_TM_TOO_LATE.

【０２６１】

```
GetNotificationResourceManager
   (IN PHANDLE                    ResourceManagerHandle、
    IN PTRANSACTION_NOTIFICATION  TransactionNotification、
    IN PLARGE_INTEGER             Timeout Optional);
```

・どれかが使用可能な場合、このルーチンは、ＲＭ通知を照会し、戻す。

【０２６２】

・引数：

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒＨａｎｄｌｅ：通知を戻す対象のＲｅｓｏｕｒｃｅＭａｎａｇｅｒを示すハンドルを与える。

【０２６３】

ＴｒａｎｓａｃｔｉｏｎＮｏｔｉｆｉｃａｔｉｏｎ：この第１の使用可能な通知で満たすべきＴＲＡＮＳＡＣＴＩＯＮ＿ＮＯＴＩＦＩＣＡＴＩＯＮ構造にポインタを与える。

【０２６４】

Ｔｉｍｅｏｕｔ：使用可能になる通知を待ちながら、それ以降、呼出し側はブロックしたいと考える時刻を与える。このタイムアウトが満了するときに何も使用可能でない場合には、この呼出し側は、ＳＴＡＴＵＳ＿ＴＩＭＥＯＵＴと共に戻る。

【０２６５】

・リターン値：

【０２６６】

【表２３】

STATUS_SUCCESS

STATUS_TIMEOUT

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES.

【０２６７】

以上でリストアップした、トランザクションに参加した後にＲＭＯカーネルオブジェクトによってＴＸカーネル上でオペレーションを実装するこれらのＡＰＩのより詳細な記述が、以下に含まれている。この記述は、このルーチン、対応する引数、およびリターン値を含んでいる。

【０２６８】

```
PrePrepareComplete
   (IN PHANDLE    EnlistmentHandle) ;
```

・このルーチンは、このＫＴＭが要求するときに、ＲＭがトランザクションのプリプリペア処理（別名「フェーズ０」）を完了していることを示す。

【０２６９】

・引数：

ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：このプリプリペアオペレーションが完了しているトランザクションを示すハンドルを与える。

【０２７０】

・リターン値：

【０２７１】

【表２４】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES
STATUS_TM_NOT_REQUESTED

【０２７２】

PrepareComplete

（IN PHANDLE　EnlistmentHandle）；

・このルーチンは、ＫＴＭが要求するときに、このＲＭがトランザクションのプリペアを完了していることを示す。

【０２７３】

・引数：

ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：このプリペアオペレーションが完了しているトランザクションを示すハンドルを与える。

【０２７４】

・リターン値：

【０２７５】

【表２５】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES
STATUS_TM_NOT_REQUESTED

【０２７６】

RollbackComplete

（IN PHANDLE　EnlistmentHandle）；

・このルーチンは、要求されたように、ＲＭがトランザクションによって実施される作業のロールバックを正常に完了していることを示す。ＲＭが、要求されたようにこのトランザクションを正常にロールバックすることができない場合には、ＲＭは、ＲｏｌｌｂａｃｋＴｒａｎｓａｃｔｉｏｎを介して完全なロールバックを求める要求を発行すべきである。

【０２７７】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：このロールバックオペレーションが完了しているトランザクションを示すハンドルを与える。
【０２７８】
・リターン値：
【０２７９】
【表２６】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES
STATUS_TM_NOT_REQUESTED

【０２８０】
CommitComplete
（IN PHANDLE EnlistmentHandle）；
・このルーチンは、要求されたように、ＲＭが、トランザクションによって実施される作業のコミットを完了していることを示す。
【０２８１】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：このコミットオペレーションが完了しているトランザクションを示すハンドルを与える。
【０２８２】
・リターン値：
【０２８３】
【表２７】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES
STATUS_TM_NOT_REQUESTED.

【０２８４】
さらに、伝搬ルーチン（ｐｒｏｐａｇａｔｉｏｎ ｒｏｕｔｉｎｅ）をこのカーネルオブジェクトについて提供することができる。このようなルーチンの実施例を以下に示す。
【０２８５】

```
RegisterProtocolAddressInformation
   (IN HANDLE         ResourceManager、
    IN PROTOCOL_ID    ProtocolId、
    IN ULONG          ProtocolInformationSize、
    IN PVOID          ProtocolInformation    Optional)。
```

【０２８６】
・このルーチンは、リソースマネージャを特定のプロトコルに対する通信リソースマネ

ージャとして登録する。このルーチンはまた、情報のブロブをこのプロトコルに関連づける。ただ１つのリソースマネージャしか、所与のマシン上のあるプロトコルに対して登録をすることができない。

【０２８７】

・引数：

ＲｅｓｏｕｒｃｅＭａｎａｇｅｒ：登録中のリソースマネージャにハンドルを与える。

【０２８８】

ＰｒｏｔｏｃｏｌＩｄ：このプロトコルを識別するＧＵＩＤ。

【０２８９】

ＰｒｏｔｏｃｏｌＩｎｆｏｒｍａｔｉｏｎＳｉｚｅ：ＰｒｏｔｏｃｏｌＩｎｆｏｒｍａｔｉｏｎのサイズ。

【０２９０】

ＰｒｏｔｏｃｏｌＩｎｆｏｒｍａｔｉｏｎ：このプロトコルに関連するオプションのブロブ。

【０２９１】

・リターン値：

【０２９２】

【表２８】

STATUS_SUCCESS
STATUS_INVALID_HANDLE

【０２９３】

```
MarshallTransaction
   (IN PHANDLE    TransactionHandle、
    IN ULONG          NumberOfProtocols、
    IN PPROTOCOL_ID   ProtocolArray、
    IN ULONG          BufferLength、
    IN PVOID          Buffer、
    OUT PULONG        BufferUsed Optional)。
```

【０２９４】

・このルーチンは、ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅに対応するこのトランザクションの表現がバッファ中に直列に配列されることを要求する。

【０２９５】

・引数：

ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：このコミットオペレーションが完了しているトランザクションを示すハンドルを与える。

【０２９６】

ＮｕｍｂｅｒＯｆＰｒｏｔｏｃｏｌｓ：このプロトコルアレイのサイズを示す。

【０２９７】

ＰｒｏｔｏｃｏｌＡｒｒａｙ：このトランザクションをマーシャリングするために使用することができるプロトコルを指定するＰＲＯＴＯＣＯＬ＿ＩＤ（ＧＵＩＤ）のアレイ。このアレイは、プリファランス（ｐｒｅｆｅｒｅｎｃｅ）によって順序づけすべきである（このアレイ中の第１のプロトコルは、優先のプロトコルであり、第２のプロトコルは、第２最優先のプロトコルであるなど）。

【０２９８】

ＢｕｆｆｅｒＬｅｎｇｔｈ：使用可能なバッファの長さを与える。

【０２９９】

Ｂｕｆｆｅｒ：このトランザクションの直列配列を記憶すべきバッファにポインタを

与える。
【０３００】
ＢｕｆｆｅｒＵｓｅｄ：バッファ中に書き込まれる実際のバイトを記憶すべきロケーションにポインタを与える。
【０３０１】
・リターン値：
【０３０２】
【表２９】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES

【０３０３】
```
GetProtocolAddressInformation
   (IN ULONG      AddressBufferSize、
    OUT PVOID     AddressBuffer、
    OUT PULONG    AddressBufferUsed Optional）。
```
【０３０４】
・このルーチンは、このマシン上ですべての登録済みのプロトコルを表現する情報が、ＡｄｄｒｅｓｓＢｕｆｆｅｒ中で、直列に配列されることを要求する。次いでこの情報を別のマシンに渡し、これをＰｕｓｈＴｒａｎｓａｃｔｉｏｎに対する引数として使用してＡｄｄｒｅｓｓＩｎｆｏｒｍａｔｉｏｎが生成されるマシンに対してトランザクションを転送（ｐｕｓｈ）することができる。
【０３０５】
・引数：
ＡｄｄｒｅｓｓＢｕｆｆｅｒＳｉｚｅ：使用可能なバッファの長さを与える。
【０３０６】
ＡｄｄｒｅｓｓＢｕｆｆｅｒ：使用可能なバッファの長さを与える。
【０３０７】
ＡｄｄｒｅｓｓＢｕｆｆｅｒＵｓｅｄ：このトランザクションの直列配列が記憶されるバッファのロケーションにポインタを与える。
【０３０８】
・リターン値：
【０３０９】
【表３０】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES

【０３１０】
```
PullTransaction
   (OUT PHANDLE            TransactionHandle、
```

IN ULONG NumberOfProtocols、
IN PCRM_PROTOCOL_ID ProtocolArray、
IN ULONG BufferLength、
IN PVOID Buffer）。

【０３１１】

・このルーチンは、バッファ中の直列配列によって表現されるトランザクションが、トランザクションマネージャにより使用可能になることを要求する。このトランザクションが、登録済みのリソースマネージャのうちの１つによって正常に伝搬された後に、この新しいトランザクションオブジェクトに対するハンドルが戻される。

【０３１２】

・引数：

ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：この新しいトランザクションを表すハンドルが記憶されるべきところにポインタを与える。

【０３１３】

ＮｕｍｂｅｒＯｆＰｒｏｔｏｃｏｌｓ：このプロトコルアレイのサイズを示す。

【０３１４】

ＰｒｏｔｏｃｏｌＡｒｒａｙ：このトランザクションをマーシャリングするために使用することができるプロトコルを指定するＰＲＯＴＯＣＯＬ＿ＩＤ（ＧＵＩＤ）のアレイ。このアレイは、プリファランスによって順序づけすべきである（このアレイ中の第１のプロトコルは、優先のプロトコルであり、第２のプロトコルは、第２最優先のプロトコルであるなど）。

【０３１５】

ＢｕｆｆｅｒＬｅｎｇｔｈ：使用可能なバッファの長さを与える。

【０３１６】

Ｂｕｆｆｅｒ：このトランザクションの直列配列が記憶されるバッファにポインタを与える。

【０３１７】

・リターン値：

【０３１８】

【表３１】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES

【０３１９】

PushTransaction
（IN HANDLE TransactionHandle、
IN ULONG NumberOfProtocols、
IN PCRM_PROTOCOL_ID ProtocolArray、
IN ULONG DestinationInfoLength、
IN PVOID DestinationInfo、
IN ULONG ResponseBufferLength、
OUT PVOID ResponseBuffer、
OUT PULONG ResponseBufferUsed Optional、
OUT PULONG PushCookie Optional)。

【０３２０】

・このルーチンは、このトランザクションが、プッシュスタイルの伝搬を使用して宛先マシンに伝搬されることを要求する。プロトコルは、ＰｒｏｔｏｃｏｌＡｒｒａｙ中にリストされる順序で、成功するまで使用されることになる。この宛先マシンへ伝搬する際にどのプロトコルでもうまくいかない場合には、このルーチンはｆａｉｌｕｒｅを戻すことになる。

【０３２１】

・引数：

ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：このリモートマシンに伝搬すべきこのトランザクションオブジェクトにポインタを与える。

【０３２２】

ＤｅｓｔｉｎａｔｉｏｎＩｎｆｏＬｅｎｇｔｈ：使用可能なＤｅｓｔｉｎａｔｉｏｎＩｎｆｏＬｅｎｇｔｈの長さを与える。

【０３２３】

ＤｅｓｔｉｎａｔｉｏｎＩｎｆｏ：この宛先についての「エンドポイント」情報が記憶されるバッファに対してポインタを与える。これは、この宛先マシン上のＧｅｔＰｒｏｔｏｃｏｌＡｄｄｒｅｓｓＩｎｆｏｒｍａｔｉｏｎに対する呼出から受信される出力とすることができる。

【０３２４】

ＲｅｓｐｏｎｓｅＢｕｆｆｅｒＬｅｎｇｔｈ：使用可能なＲｅｓｐｏｎｓｅＢｕｆｆｅｒの長さを与える。

【０３２５】

ＲｅｓｐｏｎｓｅＢｕｆｆｅｒ：このトランザクションの直列配列が記憶されるバッファにポインタを与える。

【０３２６】

ＰｕｓｈＣｏｏｋｉｅ：このプッシュ要求を表すクッキーが記憶されることになるバッファにポインタを与える。

【０３２７】

・リターン値：

【０３２８】

【表３２】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES

【０３２９】

```
GetPushTransactionBuffer
   (IN HANDLE     TransactionHandle、
    IN ULONG      PushCookie、
    IN ULONG      ResponseBufferLength、
    OUT PVOID     ResponseBuffer、
    OUT PULONG    ResponseBufferUsed Optional)。
```

【０３３０】

・ＰｕｓｈＴｒａｎｓａｃｔｉｏｎに対する初期呼出しが、ＳＴＡＴＵＳ＿ＢＵＦＦＥＲ＿ＴＯＯ＿ＳＭＡＬＬリターンコードを受信した場合に、この呼出しを使用してＰｕｓｈＴｒａｎｓａｃｔｉｏｎに対する呼出しの出力を検索する。この場合には、呼出し側は、ＧｅｔＰｕｓｈＴｒａｎｓａｃｔｉｏｎＢｕｆｆｅｒをコールし、十分なバッファサイ

ズで渡すことになる。
【０３３１】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：この新しいトランザクションを表すハンドルが記憶されるロケーションにポインタを与える。
【０３３２】
ＢｕｆｆｅｒＬｅｎｇｔｈ：使用可能なバッファの長さを与える。
【０３３３】
Ｂｕｆｆｅｒ：このトランザクションの直列配列が記憶されるバッファにポインタを与える。
【０３３４】
・リターン値：
【０３３５】
【表３３】

STATUS_SUCCESS
STATUS_ACCESS_DENIED
STATUS_INVALID_HANDLE
STATUS_INSUFFICIENT_RESOURCES

【０３３６】
```
PropagationComplete
   (IN HANDLE    EnlistmentHandle、
    IN ULONG     RequestCookie、
    IN ULONG     BufferLength、
    IN PVOID     Buffer）。
```
【０３３７】
・トランザクションの伝搬を正常に完了した後に、このルーチンは、ＣＲＭによって呼出される。
【０３３８】
・引数：
ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：この新しいトランザクションを表すハンドルを記憶すべきロケーションにポインタを与える。
【０３３９】
ＲｅｑｕｅｓｔＣｏｏｋｉｅ：どの要求が完了しているかを示すために、この元のＰＲＯＰＡＧＡＴＥ通知の引数中で受信されたこのＲｅｑｕｅｓｔＣｏｏｋｉｅを与える。
【０３４０】
ＢｕｆｆｅｒＬｅｎｇｔｈ：使用可能なこのバッファの長さを与える。
【０３４１】
Ｂｕｆｆｅｒ：このトランザクションの直列配列が記憶されるバッファにポインタを与える。
【０３４２】
・リターン値：
【０３４３】

【表３４】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

【０３４４】

PropagationFailed
（IN HANDLE ResourceManagerHandle、
IN ULONG RequestCookie、
IN STATUS PropStatus）。

【０３４５】

・ＣＲＭは、このルーチンを使用して要求されたときにこのトランザクションを伝搬することに失敗していることを示す。

【０３４６】

・引数：

ＴｒａｎｓａｃｔｉｏｎＨａｎｄｌｅ：この新しいトランザクションを表すハンドルを記憶すべきロケーションにポインタを与える。

【０３４７】

ＢｕｆｆｅｒＬｅｎｇｔｈ：使用可能なこのバッファの長さを与える。

【０３４８】

Ｂｕｆｆｅｒ：このトランザクションの直列配列が記憶されるバッファにポインタを与える。

【０３４９】

・リターン値：

【０３５０】

【表３５】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES.

【０３５１】

実施例のコンピューティング環境

図９は、一般的なコンピュータ環境９００を示しており、このコンピュータ環境を使用して本明細書中で説明している技術を実装することができる。このコンピュータ環境９００は、コンピューティング環境の一実施例にすぎず、コンピュータおよびネットワークアーキテクチャの利用範囲あるいは機能について、どのような限定も示唆するものではない。また、このコンピュータ環境９００は、この実施例のコンピュータ環境９００において示されたコンポーネントのうちのどの１つまたは組合せに関してもどのような依存性または要件を有するものとも解釈すべきではない。

【０３５２】

コンピュータ環境９００は、コンピュータ９０２の形態の汎用コンピューティングデバイスを含んでいる。コンピュータ９０２のコンポーネントは、それだけには限定されない

が、１つまたは複数のプロセッサまたは処理装置９０４と、システムメモリ９０６と、プロセッサ９０４を含めて様々なシステムコンポーネントをシステムメモリ９０６に結合するシステムバス９０８とを含むことが可能である。

【０３５３】

システムバス９０８は、メモリバスまたはメモリコントローラ、周辺バス、アクセラレーティドグラフィックスポート（ａｃｃｅｌｅｒａｔｅｄ ｇｒａｐｈｉｃｓ ｐｏｒｔ）、および様々なバスアーキテクチャのうちのどれかを使用したプロセッサバスまたはローカルバスを含めて、いくつかのタイプのバス構造のうちの１つまたは複数のいずれかを表す。実施例として、このようなアーキテクチャは、ＩＳＡバス、ＭＣＡバス、ＥＩＳＡ（Ｅｎｈａｎｃｅｄ ＩＳＡ拡張ＩＳＡ）バス、ＶＥＳＡ（Ｖｉｄｅｏ Ｅｌｅｃｔｒｏｎｉｃｓ Ｓｔａｎｄａｒｄ Ａｓｓｏｃｉａｔｉｏｎビデオエレクトロニクス規格協会）ローカルバス、メザニンバスとしても知られているＰＣＩバス、ＰＣＩエクスプレス（Ｅｘｐｒｅｓｓ）バス、ＵＳＢ、ＳＤ（Ｓｅｃｕｒｅ Ｄｉｇｉｔａｌセキュアデジタル）バス、またはＩＥＥＥ１３９４バス、すなわちファイヤワイヤ（ＦｉｒｅＷｉｒｅ）を含んでいる。

【０３５４】

コンピュータ９０２は、様々なコンピュータ読取り可能媒体を含むこともできる。このような媒体は、コンピュータ９０２によってアクセス可能な任意の入手可能な媒体とすることが可能であり、揮発性媒体も不揮発性媒体も、着脱可能媒体も着脱不能媒体も含んでいる。

【０３５５】

システムメモリ９０６は、ＲＡＭ（ｒａｎｄｏｍ ａｃｃｅｓｓ ｍｅｍｏｒｙランダムアクセスメモリ）９１０などの揮発性メモリ、および／またはＲＯＭ（ｒｅａｄ ｏｎｌｙ ｍｅｍｏｒｙ読取り専用メモリ）９１２やフラッシュＲＡＭなどの不揮発性メモリの形態のコンピュータ読取り可能媒体を含んでいる。起動中などにコンピュータ９０２内の要素間で情報を転送する助けをする基本ルーチンを含むＢＩＯＳ（Ｂａｓｉｃ ｉｎｐｕｔ／ｏｕｔｐｕｔ ｓｙｓｔｅｍ基本入出力システム）９１４は、ＲＯＭ９１２またはフラッシュＲＡＭに記憶される。ＲＡＭ９１０は、一般に処理装置９０４によって直ちにアクセス可能な、かつ／または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んでいる。

【０３５６】

コンピュータ９０２は、他の着脱可能／着脱不能の揮発性／不揮発性のコンピュータストレージ媒体を含むこともできる。実施例として、図９は、着脱不能の不揮発性の磁気媒体（図示せず）から読み取りそれに書き込むためのハードディスクドライブ９１６、着脱可能な不揮発性の磁気ディスク９２０（例えば、「フロッピー（登録商標）ディスク」）から読み取りそれに書き込むための磁気ディスクドライブ９１８、およびＣＤ－ＲＯＭ、ＤＶＤ－ＲＯＭ、他の光媒体など着脱可能な不揮発性の光ディスク９２４から読み取りまたはそれに書き込みあるいはその両方を行うための光ディスクドライブ９２２を示している。ハードディスクドライブ９１６、磁気ディスクドライブ９１８、および光ディスクドライブ９２２は、それぞれ１つまたは複数のデータ媒体インターフェース９２５によってシステムバス９０８に接続されている。代わりに、ハードディスクドライブ９１６、磁気ディスクドライブ９１８、および光ディスクドライブ９２２は、１つまたは複数のインターフェース（図示せず）によってシステムバス９０８に接続することもできる。

【０３５７】

これらのディスクドライブおよびこれらの関連するコンピュータ読取り可能媒体は、コンピュータ９０２用のコンピュータ読取り可能命令、データ構造、プログラムモジュール、および他のデータの不揮発性ストレージを提供する。この実施例では、ハードディスク９１６、着脱可能磁気ディスク９２０、および着脱可能光ディスク９２４が示されているが、磁気カセットまたは他の磁気ストレージデバイス、フラッシュメモリカード、ＣＤ－ＲＯＭ、ＤＶＤ（ｄｉｇｉｔａｌ ｖｅｒｓａｔｉｌｅ ｄｉｓｋデジタル多用途ディス

ク）または他の光ストレージ、ランダムアクセスメモリ（ＲＡＭ）、読取り専用メモリ（ＲＯＭ）、ＥＥＰＲＯＭ（ｅｌｅｃｔｒｉｃａｌｌｙ　ｅｒａｓａｂｌｅ　ｐｒｏｇｒａｍｍａｂｌｅ　ｒｅａｄ－ｏｎｌｙ　ｍｅｍｏｒｙ電気的消去可能プログラム可能読取り専用メモリ）など、コンピュータによってアクセス可能なデータを記憶することができる他のタイプのコンピュータ読取り可能媒体を利用してこの実施例のコンピューティングシステムおよびコンピューティング環境を実装することもできることが理解されよう。

【０３５８】

実施例としてオペレーティングシステム９２６、１つまたは複数のアプリケーションプログラム９２８、他のプログラムモジュール９３０、およびプログラムデータ９３２を含めて、任意数のプログラムモジュールをハードディスク９１６、磁気ディスク９２０、光ディスク９２４、ＲＯＭ９１２、および／またはＲＡＭ９１０に記憶することができる。このようなオペレーティングシステム９２６、１つまたは複数のアプリケーションプログラム９２８、他のプログラムモジュール９３０、およびプログラムデータ９３２（またはそれらの何らかの組合せ）のそれぞれは、前述の実施形態の例に従ってトランザクションを実行して、分散ファイルシステムをサポートする存在するコンポーネントのすべてまたは一部分を実装することができる。

【０３５９】

ユーザは、キーボード９３４やポインティングデバイス９３６（例えば、「マウス」）などの入力デバイスを介してコマンドおよび情報をコンピュータ９０２に入力することができる。他の入力デバイス９３８（詳細には示さず）は、マイクロフォン、ジョイスティック、ゲームパッド、サテライトディッシュ、シリアルポート、および／またはスキャナなどを含むことができる。これらおよび他の入力デバイスは、システムバス９０８に結合された入出力インターフェース９４０を介して処理装置９０４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）など他のインターフェースおよびバス構造によって接続することもできる。

【０３６０】

モニタ９４２または他のタイプのディスプレイデバイスも、ビデオアダプタ９４４などのインターフェースを介してシステムバス９０８に接続することができる。モニタ９４２に加えて、他の出力周辺デバイスには、Ｉ／Ｏインターフェース９４０を介してコンピュータ９０２に接続することができるスピーカ（図示せず）やプリンタ９４６などのコンポーネントも含めることが可能である。

【０３６１】

コンピュータ９０２は、リモートコンピューティングデバイス９４８など１台または複数台のリモートコンピュータに対する論理接続を使用してネットワーク環境で動作することが可能である。実施例として、リモートコンピューティングデバイス９４８は、ＰＣ、ポータブルコンピュータ、サーバ、ルータ、ネットワークコンピュータ、ピアデバイス、または他の共通ネットワークノードなどとすることが可能である。リモートコンピューティングデバイス９４８は、コンピュータ９０２に対して本明細書中で説明しているエレメントおよび特徴の多くまたはすべてを含むことができるポータブルコンピュータとして示されている。代わりに、コンピュータ９０２は、非ネットワーク環境でも同様に動作することができる。

【０３６２】

コンピュータ９０２とリモートコンピュータ９４８の間の論理接続は、ローカルエリアネットワーク（ＬＡＮ）９５０および一般的なワイドエリアネットワーク（ＷＡＮ）９５２として示されている。このようなネットワーキング環境はオフィス、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において一般的なものである。

【０３６３】

ＬＡＮネットワーキング環境中で実装されるとき、コンピュータ９０２は、ネットワークインターフェースまたはネットワークアダプタ９５４を介してローカルネットワーク９

５０に接続される。ＷＡＮネットワーキング環境中で実装されるときには、コンピュータ９０２は、一般にワイドネットワーク９５２上で通信を確立するためのモデム９５６または他の手段を含んでいる。コンピュータ９０２に内蔵または外付けにすることができるモデム９５６は、Ｉ／Ｏインターフェース９４０または他の適切なメカニズムを介してシステムバス９０８に接続することができる。ここに示すネットワーク接続は実施例であり、コンピュータ９０２と９４８との間で少なくとも１つの通信リンクを確立する他の手段を使用することもできる。

【０３６４】

コンピューティング環境９００と共に示すネットワーク環境などのネットワーク環境において、コンピュータ９０２に関連して示すプログラムモジュールまたはその一部分は、リモートメモリストレージデバイスに記憶することができる。実施例として、リモートアプリケーションプログラム９５８は、リモートコンピュータ９４８のメモリデバイス上に存在する。例示のために、アプリケーションまたはプログラム、およびこのオペレーティングシステムなど他の実行可能なプログラムコンポーネントは、本明細書中では個別のブロックとして示されているが、かかるプログラムおよびコンポーネントは、コンピューティングデバイス９０２の異なるストレージコンポーネント中にいろいろな時刻に存在し、このコンピュータの少なくとも１つのデータプロセッサによって実行されることが理解されよう。

【０３６５】

様々なモジュールおよび技法は、１台または複数台のコンピュータまたは他のデバイスによって実行される、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状況において本明細書中において説明することができる。一般に、プログラムモジュールは、特定のタスクを実施し、または特定の抽象データ型を実装するルーチン、プログラム、オブジェクト、コンポーネント、データ構造などを含んでいる。これらのプログラムモジュールなどは、仮想マシンや他のジャストインタイムコンパイル実行環境などの中で、ネイティブコード（ｎａｔｉｖｅ　ｃｏｄｅ）として実行し、またはダウンロードし実行することができる。一般に、プログラムモジュールの機能は、様々な実施形態において必要に応じて組み合わせ、または分散させることができる。

【０３６６】

これらのモジュールおよび技法の実装は、コンピュータ読取り可能媒体の何らかの形態上に記憶し、またはその形態に渡って伝送することができる。コンピュータ読取り可能媒体は、コンピュータによりアクセスすることが可能な任意の使用可能な媒体とすることが可能である。実施例として、限定するものではないが、コンピュータ読取り可能媒体は、「コンピュータストレージ媒体」と「通信媒体」を含むことができる。

【０３６７】

「コンピュータストレージ媒体」は、コンピュータ読取り可能命令、データ構造、プログラムモジュール、または他のデータなどの情報の記憶のための任意の方法または技術により実装される揮発性および不揮発性の、着脱可能および着脱不能の媒体を含んでいる。コンピュータストレージ媒体は、それだけには限定されないが、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリもしくは他のメモリ技術、ＣＤ－ＲＯＭ、デジタル多用途ディスク（ＤＶＤ）もしくは他の光ストレージ、磁気カセット、磁気テープ、磁気ディスクストレージ、もしくは他の磁気ストレージデバイス、または所望の情報を記憶するために使用することができコンピュータによりアクセスすることが可能な他の任意の媒体を、含んでいる。

【０３６８】

「通信媒体」は一般に、搬送波や他の搬送メカニズムなどの被変調データ信号の形で、コンピュータ読取り可能命令、データ構造、プログラムモジュール、または他のデータを実施する。通信媒体は、任意の情報配信媒体も含んでいる。用語「被変調データ信号」は、信号中の情報を符号化するようにしてその信号中の１つまたは複数のその特性が設定あるいは変更された信号を意味する。非限定的な実施例にすぎないが、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、並びに、音響、ＲＦ、赤外線、他の無線媒体などの無線媒体が含まれる。以上のうちの任意の組合せもまた、コンピュータ読取り可能媒体の範囲内に含まれる。

【０３６９】

本明細書全体を通して、「一実施形態（ｏｎｅ　ｅｍｂｏｄｉｍｅｎｔ）」、「一実施形態（ａｎ　ｅｍｂｏｄｉｍｅｎｔ）」または「実施形態の一例（ａｎ　ｅｘａｍｐｌｅ　ｅｍｂｏｄｉｍｅｎｔ）」と言及しているが、これらは、説明している特定の機能、構成、または特徴が本発明の少なくとも一実施形態中に含まれることを意味する。したがって、このようなフレーズの使用によって、ただ１つの実施形態を超えたものについて言及することができる。さらに、この説明される機能、構成、または特徴は、１つまたは複数の実施形態において適切な任意の方法で組み合わせることができる。

【０３７０】

しかし、本発明は、１つまたは複数の特定の細部によらなくても、あるいは、他の方法、リソース、材料などを用いても実行することができることは当業者に理解されるだろう。他の事例においては、本発明の態様を単にあいまいにしてしまうことがないように、よく知られている構成、リソース、またはオペレーションを詳細に示さず、または説明していない。

【０３７１】

本発明の実施形態の例および用途について例示し説明してきたが、本発明は上述の正確な構成およびリソースに限定されるものでないことを理解されたい。本明細書中に開示している本発明の方法およびシステムの構成、オペレーション、および細部において、請求している本発明の範囲を逸脱することなく、当業者にとって明らかな様々な修正、変更、および変形を行うことができる。

【図面の簡単な説明】

【０３７２】

【図１】トランザクション処理リモートファイルオペレーションを使用することができるシステムの一実施例を示す図である。

【図２】図１のシステムのクライアントおよびサーバのコンポーネントの一実施例を示す図である。

【図３】図２のクライアントとサーバの間のトランザクション処理リモートファイルオペレーションについての実施例の処理フローを示す図である。

【図３Ａ】図２のクライアントとサーバの間のトランザクション処理リモートファイルオペレーションについての実施例の処理フローを示す図である。

【図４】図２のクライアントおよびサーバによる複数のリモートファイルアクセスの一実施例を示す図である。

【図５】ネットワーク上のトランザクションの２相コミットを実施するための一実施例の処理フローを示す図である。

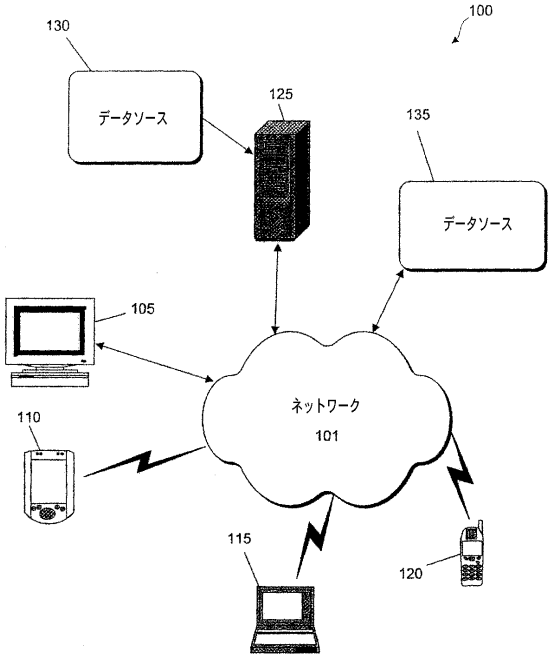
【図６】トランザクション管理を実装するためのコンポーネントの一実施例を示す図である。

【図７】カーネルレベルトランザクションについての一実施例処理フローを示す図である。

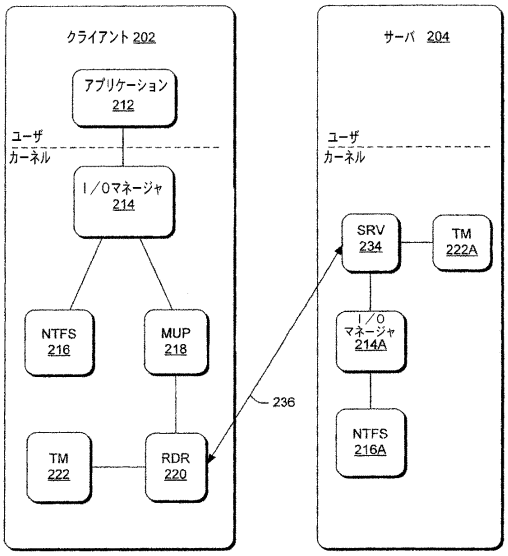
【図８】セキュリティフィーチャの一実施例を示す図である。

【図９】様々な実施形態による、本明細書中で説明される技法を実装するために使用することができる一般的なコンピュータ環境を示す図である

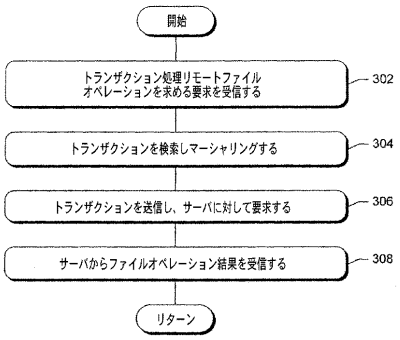
【図１】



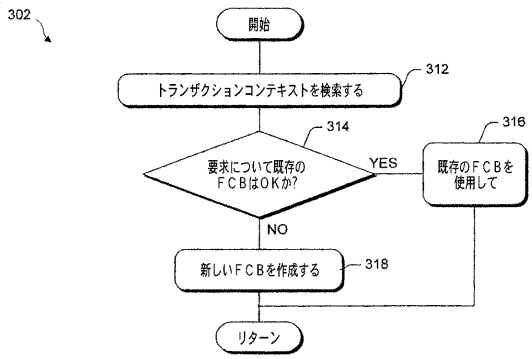
【図２】



【図３】



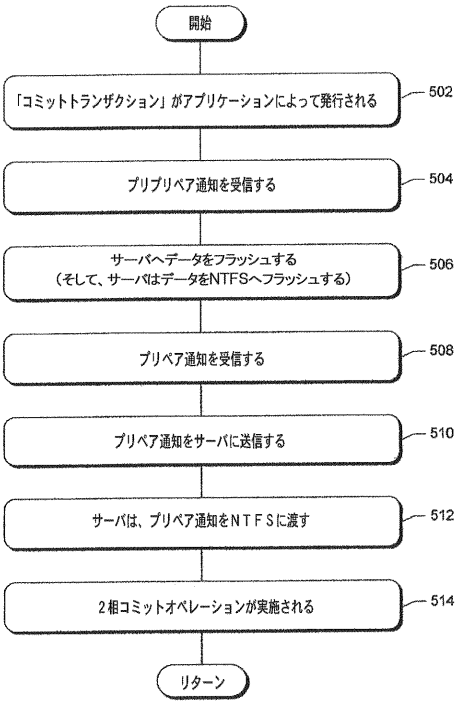
【図３Ａ】



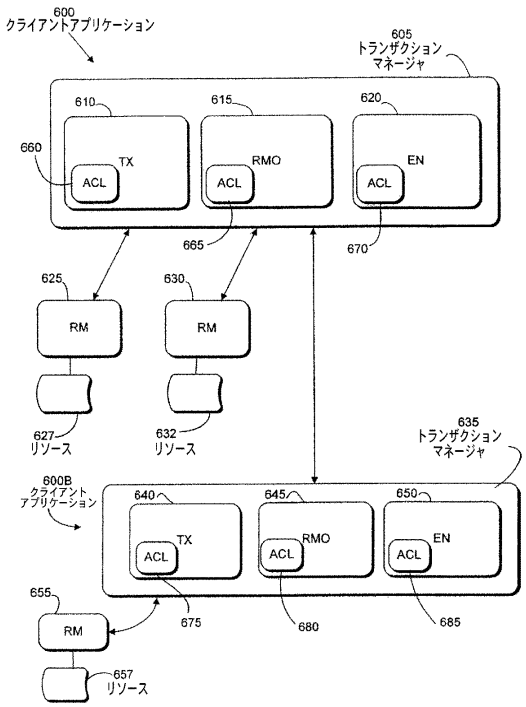
【図４】

	オペレーション	戻されるハンドル	トランザクション	見られるバージョン
401	読取りのためオープンする	H1	T1	A
402	読取りのためオープンする	H2	T2	A
403	書込みのためオープンする	H3	T2	B
404	読取りのためオープンする	H4	T2	B
405	読取りのためオープンする	H5	T3	A
406	T2をコミットする		T2	
407	読取りのためオープンする	H6	T1	C
408	読取りのためオープンする	H7	T3	C

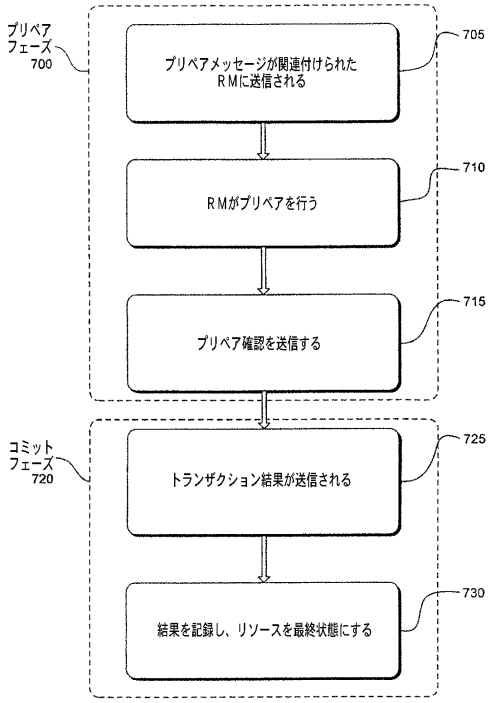
【図５】


開始
「コミットトランザクション」がアプリケーションによって発行される
502
プリプリペア通知を受信する
504
サーバへデータをフラッシュする
（そして、サーバはデータをNTFSへフラッシュする）
506
プリペア通知を受信する
508
プリペア通知をサーバに送信する
510
サーバは、プリペア通知をＮＴＦＳに渡す
512
2相コミットオペレーションが実施される
514
リターン

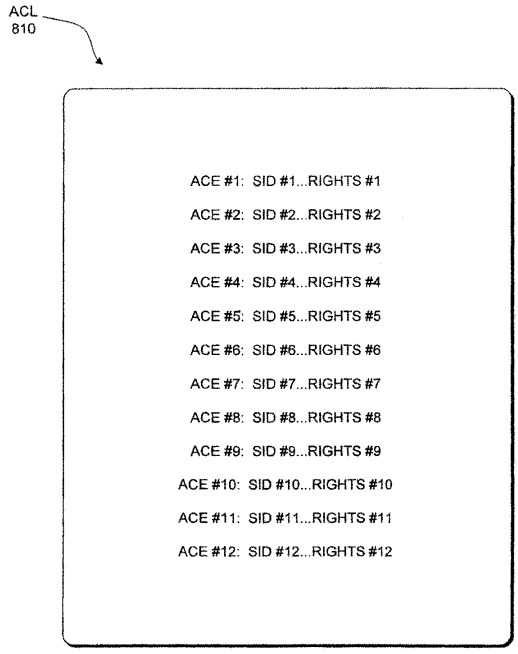
【図６】


600
クライアントアプリケーション
605
トランザクション
マネージャ
610
615
620
660
TX
RMO
EN
ACL
665
670
625
630
RM
627
リソース
632
635
600B
640
645
650
655
675
680
685
657

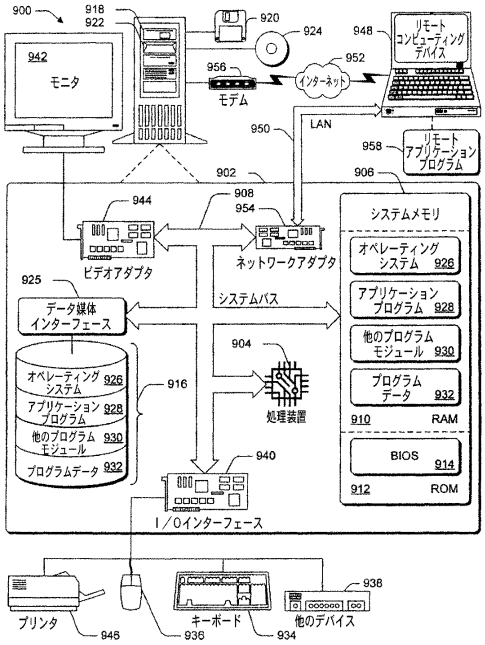
【図７】


プリペア
フェーズ
700
プリペアメッセージが関連付けられた
RMに送信される
705
RMがプリペアを行う
710
プリペア確認を送信する
715
コミット
フェーズ
720
トランザクション結果が送信される
725
結果を記録し、リソースを最終状態にする
730

【図８】


ACL
810
ACE #1: SID #1...RIGHTS #1
ACE #2: SID #2...RIGHTS #2
ACE #3: SID #3...RIGHTS #3
ACE #4: SID #4...RIGHTS #4
ACE #5: SID #5...RIGHTS #5
ACE #6: SID #6...RIGHTS #6
ACE #7: SID #7...RIGHTS #7
ACE #8: SID #8...RIGHTS #8
ACE #9: SID #9...RIGHTS #9
ACE #10: SID #10...RIGHTS #10
ACE #11: SID #11...RIGHTS #11
ACE #12: SID #12...RIGHTS #12

【図９】


900
918
922
920
924
948
リモート
コンピューティング
デバイス
942
モニタ
956
モデム
952
インターネット
950
LAN
958
リモート
アプリケーション
プログラム
902
906
908
954
944
システムメモリ
ビデオアダプタ
ネットワークアダプタ
オペレーティング
システム 926
925
データ媒体
インターフェース
システムバス
アプリケーション
プログラム 928
他のプログラム
モジュール 930
プログラム
データ 932
904
916
処理装置
910 RAM
オペレーティング
システム 926
アプリケーション
プログラム 928
他のプログラム
モジュール 930
プログラムデータ 932
940
BIOS 914
912 ROM
I／Oインターフェース
938
プリンタ
946
キーボード
936
他のデバイス
934

フロントページの続き

(72)発明者　プラディープ　ヤナ　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　シシール　パディカ
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　バラン　セシュ　ラマン
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
(72)発明者　スレンドラ　ヴェルマ
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　ジョン　カーギル
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　ヤコブ　ラクチュール
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　田川　泰宏

(56)参考文献　特表２００３－５３０６４６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 12/00