(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4972082号
(P4972082)
(45)発行日　平成24年7月11日(2012.7.11)　(24)登録日　平成24年4月13日(2012.4.13)

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

請求項の数 4　(全 34 頁)

(21)出願番号　特願2008-504005 (P2008-504005)
(86)(22)出願日　平成17年7月28日(2005.7.28)
(65)公表番号　特表2008-535081 (P2008-535081A)
(43)公表日　平成20年8月28日(2008.8.28)
(86)国際出願番号　PCT/US2005/026858
(87)国際公開番号　WO2006/107318
(87)国際公開日　平成18年10月12日(2006.10.12)
審査請求日　平成20年7月28日(2008.7.28)
(31)優先権主張番号　11/096,871
(32)優先日　平成17年4月1日(2005.4.1)
(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　100140109
弁理士　小野　新次郎
(74)代理人　100075270
弁理士　小林　泰
(74)代理人　100080137
弁理士　千葉　昭男
(74)代理人　100096013
弁理士　富田　博行
(74)代理人　100153028
弁理士　上田　忠

最終頁に続く

(54)【発明の名称】開発者がシステム上の周知のロケーションを容易に発見し、または拡張するための能力

(57)【特許請求の範囲】
【請求項１】
１つ以上のコンピュータ上で動作する第１のコンポーネントと第２のコンポーネントとの間の通信の方法であって、
（ａ）第１のコンポーネントから、記憶装置上の既知のフォルダの識別を含むコールを、通信媒体を介して受け取るステップと、
（ｂ）前記既知のフォルダの記憶装置上のロケーションを求めて、レジストリを検索するステップと、
（ｃ）受け取った前記既知のフォルダの識別が有効な場合、決定されたロケーションにある前記既知のフォルダからデータをアクセスし、前記記憶装置上の前記既知のフォルダの前記第１のコンポーネントによる使用法を前記第２のコンポーネントへ知らせる新たなプロパティを前記既知のフォルダへ追加するステップと、
（ｄ）受け取った前記既知のフォルダの識別が有効でない場合、新規の既知のフォルダを記憶装置上に作成し、前記第２のコンポーネントに前記新規の既知のフォルダのロケーションを、通信媒体を介して提供するステップと、
を含む方法。
【請求項２】
前記決定されたロケーションがローカルハードドライブを含む、請求項１に記載の方法。
【請求項３】

前記決定されたロケーションがネットワークドライブを含む、請求項１に記載の方法。
【請求項４】
前記決定されたロケーションがウェブサイトを含む、請求項１に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は概して、１組のアプリケーションプログラミングインターフェイスを用いた、周知のロケーションに格納されるフォルダの作成および管理に関する。
【背景技術】
【０００２】
今日、開発者およびユーザは通常、様々なアプリケーションからのデータを、マイドキュメントフォルダやマイピクチャフォルダなど、特定の周知のフォルダに格納する。こうした周知のフォルダは、開発者およびユーザが情報にアクセスするために、多数のアプリケーションおよびコンピュータネットワーク環境に渡る容易なアクセスを提供する。こうした周知のフォルダは、Ｗｉｎｄｏｗｓ（登録商標）ブランドのオペレーティングシステムなどのオペレーティングシステムにおいて使用される。
【０００３】
たとえば、多くのアプリケーションは、インストールされる特定のアプリケーションに利用可能なデータを格納するマイドキュメントフォルダなど、特定の周知のフォルダを使用する。マイドキュメントフォルダの使用により、他のアプリケーションに対して、格納された情報にアクセスする能力を可能にする。たとえば、ｊｐｅｇファイルは、いくつかの写真公開（photo publishing）アプリケーションによって使用され得る。特定の写真公開アプリケーションは、他の写真公開アプリケーションがｊｐｅｇファイルにアクセスすることができるように、マイドキュメントフォルダなど、周知のフォルダにｊｐｅｇファイルを格納することができる。しかし、多数のアプリケーションファイルおよびフォルダをマイドキュメントフォルダの下に置くと、フォルダのリストが系統化されていない状態となり、特定のフォルダおよびデータの検索を煩雑かつ時間のかかるものとしてしまう。
【０００４】
マイクロソフト（登録商標）Ｗｉｎｄｏｗｓ（登録商標）などの既存のオペレーティングシステムは、シェル（Shell）を使用して、フォルダおよびファイルなどのデータオブジェクトを、ユーザインターフェイスまたはアプリケーションにより階層状の名前空間構造に系統化する。シェルは、そのロケーションおよび存在がシステムに知られ得るとともにそれに対するアクセスがスタートメニューなどのシェル中の数多くの場所から与えられる特殊なフォルダを含み得る。シェルは、こうした周知のフォルダを管理するための、１組のＳＨＦｏｌｄｅｒＰａｔｈ　ＡＰＩを開発者およびユーザに提供し得る。ＳＨＦｏｌｄｅｒＰａｔｈ　ＡＰＩは、固定された１組のＣＳＩＤＬを使用し得る。ＣＳＩＤＬは、フォルダを識別するのに使われる序数値を含み、アプリケーションによって頻繁に使われる特殊なフォルダを識別するための、独自のシステム非依存の方法を提供する。ディスクまたはドライブ上の周知のフォルダのロケーションでＣＳＩＤＬをマップするために、テーブルが使用される。マイクロソフト（登録商標）Ｗｉｎｄｏｗｓ（登録商標）ＸＰなど、既存のオペレーティングシステムの現在のバージョンでは、既知のフォルダは、ＣＳＩＤＬが拡張可能でないように、拡張可能ではない。さらに、既存のオペレーティングシステムにおける既存の既知のフォルダへの新規プロパティの追加も可能ではない。
【発明の開示】
【発明が解決しようとする課題】
【０００５】
したがって、開発者およびユーザが、様々なアプリケーションによって使用するために自分用の既知のフォルダを作成し得る方法およびデータ構造を提供することが、当該分野における進歩であろう。さらに、開発者またはユーザによって使用され得る既存のならびに新規の既知のフォルダに新規またはカスタムプロパティを追加することも進歩であろう

。こうしたカスタムプロパティは、たとえば、開発者またはユーザに、既知のフォルダの意図した使用法を知らせることができる。さらに、本方法およびデータ構造は、同一のコンピュータ上またはネットワーク上に置かれ得る様々なアプリケーションによって作成された他の既知のフォルダを、ユーザが最低限の作業量で見つけ出し使用することを可能にするものである。

【課題を解決するための手段】

【０００６】

本発明の方法およびデータ構造は、システム上の既知のロケーションを列挙するための、かつ／または特定の周知のロケーションを位置決めするための１組のＡＰＩを提供することによって、従来技術の課題を克服する。具体的には、本発明は、周知のフォルダを位置決めし、拡張し、列挙する機能性を提供する。さらに、本発明は、開発者および／またはユーザが周知のフォルダのプロパティをカスタマイズすることを可能にする。

【０００７】

添付の図面を併せ見て以下の説明を参照することによって、本発明がより完全に理解され、本発明の利点が理解されよう。図面において、同じ参照番号は同じ特徴を示す。

【発明を実施するための最良の形態】

【０００８】

本発明の開示内容を明らかにするために、ここで、いくつかの関連用語の定義が与えられる。

【０００９】

プロファイル：オペレーティングシステムおよびアプリケーションに、ユーザ固有データおよび設定を格納するための、ユーザごとのロケーションを与える。

【００１０】

既知のフォルダ／ＫｎｏｗｎＦｏｌｄｅｒ：Ｗｉｎｄｏｗｓ（登録商標）シェルならびに他のコンポーネントおよびアプリケーションに対して知らされるフォルダのカテゴリ。

【００１１】

ＳＨＦｏｌｄｅｒＰａｔｈ（）ＡＰＩ：ＳＨＧｅｔＦｏｌｄｅｒＬｏｃａｔｉｏｎ（）、ＳＨＧｅｔＦｏｌｄｅｒＰａｔｈ（）、およびＳＨＳｅｔＦｏｌｄｅｒＰａｔｈ（）を含む１組のシェルフォルダＡＰＩ。

【００１２】

ＣＳＩＤＬ：フォルダを識別し、アプリケーションによって頻繁に使われる特殊フォルダの、システム非依存の一意の識別法を提供するのに使われる序数値。

【００１３】

図１は、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過ぎず、本発明の利用または機能性の範囲に対するどのような限定を示唆することも意図されない。コンピューティング環境１００は、例示的な動作環境１００に示されるどのコンポーネントまたはその組合せに関するどのような依存も要件も有していると解釈されるべきでない。

【００１４】

図１を参照すると、本発明を実施する例示的なシステムは、汎用コンピューティングデバイスを、コンピュータ１１０の形で含む。コンピュータ１１０のコンポーネントは、処理ユニット１２０と、システムメモリ１３０と、システムメモリなど様々なシステムコンポーネントを処理ユニット１２０に結合するシステムバス１２１とを含み得るが、それに限定されない。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などいくつかのタイプのバス構造のいずれでもよい。限定ではなく例として、このようなアーキテクチャは、ＩＳＡ（業界標準アーキテクチャ）バス、ＭＣＡ（マイクロチャネルアーキテクチャ）バス、ＥＩＳＡ（拡張ＩＳＡ）バス、ＶＥＳＡ（米国ビデオ電子装置規格化協会）ローカルバス、およびメザニン（Ｍｅｚｚａｎｉｎｅ）バスとしても知られるＰＣＩ（周辺装置

相互接続）バスを含む。

【００１５】

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とともに揮発性媒体および不揮発性媒体両方、取外し可能媒体および固定式媒体を含む、利用可能などの媒体でもよい。限定ではなく例として、コンピュータ可読媒体は、コンピュータ記憶媒体および通信媒体を含み得る。コンピュータ記憶媒体は、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、または他のデータなどの情報を格納するためのどの方法でも技術でも実施される揮発性媒体および不揮発性媒体両方、取外し可能媒体および固定式媒体を含む。コンピュータ記憶媒体は、ＲＡＭ、ＲＯＭ、ＥＰＲＯＭ、フラッシュメモリまたは他のメモリ技術、ＣＤ－ＲＯＭ、ＤＶＤ（デジタル多用途ディスク）または他の光学ディスク記憶装置、磁気カセット、磁気テープ、磁気ディスク記憶装置または他の磁気記憶装置、あるいは、所望の情報を格納するのに使われ得るとともにコンピュータ１１０によってアクセスされ得る他のどの媒体も含むが、それに限定されない。通信媒体は一般に、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、または他のデータを、変調データ信号、たとえば搬送波や他の移送機構として具体化し、どの情報配信媒体も含む。「変調データ信号」という用語は、信号中の情報を符号化するようなやり方で設定され、または変更される信号特性の１つまたは複数を有する信号を意味する。限定ではなく例として、通信媒体は、有線ネットワークや直接有線接続などの有線媒体、ならびに音響、ＲＦ、赤外線、および他の無線媒体などの無線媒体を含む。上記のどの組合せも、やはりコンピュータ可読媒体の範囲に含まれるべきである。

【００１６】

システムメモリ１３０は、コンピュータ記憶媒体を、ＲＯＭ（読出し専用メモリ）１３１およびＲＡＭ（ランダムアクセスメモリ）１３２など、揮発性および／または不揮発性メモリの形で含む。ＢＩＯＳ（基本入出力システム）１３３は、たとえば起動中にコンピュータ１１０内部の要素間での情報の転送を助ける基本ルーチンを含み、通常はＲＯＭ１３１に格納される。ＲＡＭ１３２は一般に、処理ユニット１２０に対してただちにアクセス可能な、かつ／または処理ユニット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。限定ではなく例として、図１では、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、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。

【００１７】

コンピュータ１１０は、他の取外し可能／固定式、揮発性／不揮発性コンピュータ記憶媒体も含み得る。単なる例として、図１では、固定式不揮発性磁気媒体からの読出しまたはそこへの書込みを行うハードディスクドライブ１４０、取外し可能な不揮発性磁気ディスク１５２からの読出しまたはそこへの書込みを行う磁気ディスクドライブ１５１、および、ＣＤ　ＲＯＭや他の光学媒体など取外し可能な不揮発性光ディスク１５６からの読出しまたはそこへの書込みを行う光ディスクドライブ１５５を示す。例示的な動作環境で使われ得る他の取外し可能／固定式、揮発性／不揮発性コンピュータ記憶媒体は、磁気テープカセット、フラッシュメモリカード、デジタル多用途ディスク、デジタルビデオテープ、固体状態ＲＡＭ、固体状態ＲＯＭなどを含むが、それに限定されない。ハードディスクドライブ１４１は通常、インターフェイス１４０などの固定式メモリインターフェイスによって、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、インターフェイス１５０などの取外し可能メモリインターフェイスによって、システムバス１２１に接続される。

【００１８】

上述し、かつ図１に示されているドライブおよびそれに関連するコンピュータ可読媒体は、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、およびコンピュータ１１０のための他のデータの格納を可能にする。図１では、たとえば、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、

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る。こうしたコンポーネントは、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、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、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は、少なくとも異なるものであることを示すために、ここでは異なる番号が与えられている。ユーザは、キーボード１６２、および一般にマウス、トラックボール、またはタッチパッドと呼ばれるワイヤレスポインティングデバイス１６１を介して、コマンドおよび情報をコンピュータ１１０に入力することができる。他の入力デバイス（図示せず）は、マイクロフォン、ジョイスティック、ゲーム用パッド、衛星パラボラアンテナ、スキャナなどを含み得る。こうしたおよび他の入力デバイスはしばしば、システムバスに結合されるユーザ入力インターフェイス１６０を介して処理ユニット１２０に接続されるが、パラレルポート、ゲームポート、ＵＳＢ（ユニバーサルシリアルバス）など、他のインターフェイスおよびバス構造によっても接続され得る。モニタ１９１または他のタイプの表示デバイスも、ビデオインターフェイス１９０などのインターフェイスを介してシステムバス１２１に接続される。モニタに加え、コンピュータは、出力周辺インターフェイス１９０を介して接続され得るスピーカ１９７およびプリンタ１９６など、他の周辺出力デバイスも含み得る。

【００１９】

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用いて、ネットワーク接続された環境において動作し得る。リモートコンピュータ１８０は、パーソナルコンピュータ、サーバ、ルータ、ネットワークＰＣ、ピアデバイス、または他の共通ネットワークノードでよく、通常、コンピュータ１１０に関連して上述され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のみが示されている。図１に示される論理接続は、ＬＡＮ（ローカルエリアネットワーク）１７１およびＷＡＮ（ワイドエリアネットワーク）１７３を含むが、他のネットワークも含み得る。このようなネットワーク環境は、会社、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よく見られる。

【００２０】

ＬＡＮネットワーク環境において使われる場合、コンピュータ１１０は、ネットワークインターフェイスまたはアダプタ１７０を介してＬＡＮ１７１に接続される。ＷＡＮネットワーク環境において使われる場合、コンピュータ１１０は通常、モデム１７２、または、たとえば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、内部にあっても外部にあってもよく、ユーザ入力インターフェイス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。ネットワーク接続された環境では、コンピュータ１１０に関連して図示されるプログラムモジュールまたはその一部は、リモートメモリ記憶装置に格納され得る。限定ではなく例として、図１は、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に常駐するものとして示す。図示されるネットワーク接続は例示であり、コンピュータ間の通信リンクを確立する他の手段も使われ得ることが理解されよう。周辺インターフェイス１９５は、スキャナ（図示せず）やデジタルカメラ１９４などのビデオ入力デバイスとインターフェイスをとることができ、出力周辺インターフェイスは、ユニバーサルシリアルバス（ＵＳＢ）インターフェイスを含む、標準化されたインターフェイスをサポートし得る。

【００２１】

本発明は、他の数多くの汎用または専用のコンピューティングシステム環境または構成とともに動作する。本発明とともに使用するのに適切であり得る他の公知のコンピューティングシステム、環境、および／または構成の例は、パーソナルコンピュータ、サーバコンピュータ、ハンドヘルドデバイスまたはラップトップデバイス、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、プログラム可能な家電製品、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

システムまたはデバイスのいずれをも含む分散型コンピューティング環境などを含むが、それに限定されない。

【００２２】

本発明は、コンピュータ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という一般的な状況において説明され得る。概して、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タイプ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む。本発明は、通信ネットワークを介して連結されるリモート処理デバイスによってタスクが実施される分散型コンピューティング環境においても実施され得る。分散型コンピューティング環境では、プログラムモジュールは、メモリ記憶デバイスを含むローカルおよびリモートコンピュータ記憶媒体両方に置かれ得る。

【００２３】

プログラミングインターフェイス（またはより簡単には、インターフェイス）は、コードの１つまたは複数のセグメント（群）が、コードの１つまたは複数の他のセグメント（群）によって提供される機能性と通信し、またはその機能性にアクセスすることを可能にする、どの仕組み、処理、プロトコルとしても見なされ得る。あるいは、プログラミングインターフェイスは、他のコンポーネント（群）の、１つまたは複数の仕組み（群）、メソッド（群）、関数呼出し（群）、モジュール（群）などと通信結合することができる、システムのあるコンポーネントの、１つまたは複数の仕組み（群）、メソッド（群）、関数呼出し（群）、モジュール（群）、オブジェクト（群）などとして見なされ得る。上記の文における「コードのセグメント」という用語は、適用される専門範囲に関わらず、かつコードセグメントが個別にコンパイルされるかどうか、コードセグメントがソースコード、中間コード、またはオブジェクトコードとして提供されるのか、コードセグメントがランタイムシステムまたは処理において使用されるのか、コードセグメントが同じマシンもしくは異なるマシンに置かれるか、それとも複数のマシンに分散されるか、あるいはコードのセグメントによって表される機能性が、完全にソフトウェアとして、完全にハードウェアとして、またはハードウェアおよびソフトウェアの組合せとして実装されるのかに関わらず、１つまたは複数の命令またはコードの行を含み、たとえば、コードモジュール、オブジェクト、サブルーチン、関数などを含むことを意図している。

【００２４】

概念的には、プログラミングインターフェイスは、図２または図３に示すように、包括的に見なされ得る。図２は、第１および第２のコードセグメントが通信するためのパイプ（ｃｏｎｄｕｉｔ）としてのインターフェイス１というインターフェイスを示す。図３は、インターフェイスを、インターフェイスオブジェクトＩ１およびＩ２（第１および第２のコードセグメントの一部であってもなくてもよい）を備えるものとして示し、こうしたインターフェイスオブジェクトは、システムの第１および第２のコードセグメントが、媒体Ｍを介して通信することを可能にする。人によっては、図３を見て、インターフェイスオブジェクトＩ１およびＩ２を、同一のシステムの別個のインターフェイスと見なし、また、オブジェクトＩ１およびＩ２と媒体Ｍとがインターフェイスを構成すると見なすであろう。図２および３は、双方向の流れ、およびその流れの両端でのインターフェイスを示すが、特定の実装形態は、一方向での情報の流れしかもたない（または後で説明されるように情報の流れをもたない）場合もあり、一端でのインターフェイスオブジェクトしかもたない場合もある。限定ではなく例として、アプリケーションプログラミングインターフェイス（ＡＰＩ）、エントリポイント、メソッド、関数、サブルーチン、リモートプロシージャコール、およびコンポーネントオブジェクトモデル（ＣＯＭ）インターフェイスなどの用語は、プログラミングインターフェイスの定義範囲に包含される。

【００２５】

このようなプログラミングインターフェイスの様相は、第１のコードセグメントが、第２のコードセグメントに情報（この場合、「情報」は、広い意味で使われており、データ、コマンド、要求などを含む）を送信するメソッドと、第２のコードセグメントが、その

情報を受け取るメソッドと、その情報の構造、シーケンス、シンタクス、構成（organization）、スキーマ、計時、および内容とを含み得る。これに関して、情報が、インターフェイスによって定義されたやり方で移送される限り、基底の移送媒体自体は、有線でも無線でも、あるいは有線および無線の組合せであっても、インターフェイスの動作にとって重要ではない場合がある。特定の状況では、情報は、従来の意味での一方向または両方向で渡されることができない。というのは、情報の転送は、あるコードセグメントが、第２のコードセグメントによって実施される機能性に単にアクセスするときのように、別の仕組み（たとえば、コードセグメントの間の情報の流れとは別に、情報が、バッファ、ファイルなどに置かれる）を介するか、または存在しなくてもよいからである。こうしたアスペクトのいずれかまたは全部は、所与の状況、たとえば、コードセグメントが、疎結合構成または密結合構成どちらのシステムの一部であるかに応じて重要となる場合があり、したがって、このリストは、例示であって限定のためではないと見なされるべきである。

【００２６】

このプログラミングインターフェイス概念は、当業者に公知であり、本発明の上記の詳細な説明から明らかである。しかし、プログラミングインターフェイスを実装する他の方法もあり、明示的に除外されない限り、こうした方法も、本明細書に添付され定義される特許請求の範囲に包含されることを意図している。このような他の方法は、図２および３の単純化された概観より高度または複雑に見えることがあるが、それにもかかわらず、全体として同じ結果を達成する類似の機能を実施する。ここで、プログラミングインターフェイスのいくつかの例示的な代替実装形態を簡潔に説明する。

【００２７】

あるコードセグメントから別のコードセグメントへの通信は、その通信を、多数の別々の通信に分けることによって、間接的に達成されることができる。このことは、図４および１９で概略的に示される。図示されるように、いくつかのインターフェイスは、分割可能ないくつかの組の機能性として説明され得る。したがって、図２および３のインターフェイス機能性は、２４または２×２×３×２を数学的にもたらし得るように、同じ結果を達成するように因数分解され得る。したがって、図１８に示されるように、インターフェイス１というインターフェイスによって提供される機能は、インターフェイスの通信をインターフェイス１Ａ、インターフェイス１Ｂ、インターフェイス１Ｃなどの複数のインターフェイスに変換するように下位分割され、同じ結果を達成し得る。図５に示されるように、インターフェイスＩ１によって提供される機能は、複数のインターフェイスＩ１ａ、Ｉ１ｂ、Ｉ１ｃなどに下位分割され、同じ結果を達成し得る。同様に、第１のコードセグメントから情報を受け取る、第２のコードセグメントのインターフェイスＩ２は、複数のインターフェイスＩ２ａ、Ｉ２ｂ、Ｉ２ｃなどに因数分解され得る。因数分解する際、第１のコードセグメントに含まれるインターフェイスの数は、第２のコードセグメントに含まれるインターフェイスの数と一致しなくてもよい。図４および５のいずれの場合でも、インターフェイス１およびＩ１というインターフェイスの機能上の意図は、それぞれ図２および３の場合と同じままである。インターフェイスの因数分解は、因数分解が認識困難になり得るように、結合的な、可換な、および他の数学的なプロパティの後に起こってもよい。たとえば、動作の順序は重要でなくてよく、したがって、インターフェイスによって実現される機能は、インターフェイスの遂行に先立って、別のコード部分またはインターフェイスによって実現されても、システムの別のコンポーネントによって実施されてもよい。さらに、プログラミング分野の当業者であれば、同じ結果を達成する異なる関数呼出しを行う様々な方法があることを理解できよう。

【００２８】

いくつかの場合では、プログラミングインターフェイスの特定のアスペクト（たとえばパラメータ）を無視し、追加し、または再定義しながら、意図した結果を依然として達成することが可能であり得る。このことは、図６および７に示される。たとえば、図２のインターフェイス１というインターフェイスが、Ｓｑｕａｒｅ（ｉｎｐｕｔ，ｐｒｅｃｉｓｉｏｎ，ｏｕｔｐｕｔ）の関数呼出しを含み、この呼出しは、３つのパラメータ、すなわ

ちｉｎｐｕｔ、ｐｒｅｃｉｓｉｏｎ、およびｏｕｔｐｕｔを含み、第１のコードセグメントから第２のコードセグメントに発行されると仮定する。中央のパラメータｐｒｅｃｉｓｉｏｎは、図６に示すように、所与のシナリオに関係しない場合、適切に無視されても、（この状況において）ｍｅａｎｉｎｇｌｅｓｓパラメータで置き換えられてもよい。無関係なａｄｄｉｔｉｏｎａｌパラメータを追加することもできる。いずれの場合でも、第２のコードセグメントによって入力が二乗された後で出力が返される限り、二乗の機能性は達成され得る。ｐｒｅｃｉｓｉｏｎは、コンピューティングシステムの何らかの下流部分または他の部分にとって有意なパラメータでもよい。ただし、ｐｒｅｃｉｓｉｏｎは、二乗計算という狭い目的にとって必要ないと認識されると、置き換えられても無視されてもよい。たとえば、有効なｐｒｅｃｉｓｉｏｎ値を渡すのではなく、誕生日などの無意味な値が、結果に悪影響を与えることなく渡されてもよい。同様に、図７に示すように、インターフェイスＩ１は、インターフェイスＩ１’で置き換えられ、インターフェイスへのパラメータを無視しまたは追加するように再定義される。インターフェイスＩ２は、同様にインターフェイスＩ２’として再定義されることができ、不必要なパラメータまたは他の所で処理されることができるパラメータを無視するように再定義される。ここでのポイントは、いくつかの場合において、プログラミングインターフェイスは、何らかの目的には必要とされないパラメータなどのアスペクトを含む場合があるので、こうしたアスペクトは、無視されまたは再定義され、あるいは他の目的のために他の所で処理され得ることである。

【００２９】

２つの別々のコードモジュールの機能性の間の「インターフェイス」が形を変えるように、そうした機能性の一部または全部をマージすることも可能であり得る。たとえば、図２および３の機能性は、それぞれ図８および９の機能性にコンバートされ得る。図８において、図２の以前の第１および第２のコードセグメントは、その両方を含むモジュールにマージされる。この場合、コードセグメントは、依然として相互に通信していてもよいが、インターフェイスは、単一モジュールにより適した形に適合され得る。したがって、たとえば、正式な呼出しおよび戻りステートメントは不必要になり得るが、インターフェイス１というインターフェイスに準じた同様の処理または応答（群）は依然として有効であり得る。同様に、図９に示されるように、図３のインターフェイスＩ２の一部（または全部）は、インターフェイスＩ１にインラインに書き込まれて、インターフェイスＩ１’’を形成し得る。図示されるように、インターフェイスＩ２は、Ｉ２ａおよびＩ２ｂに分割され、インターフェイス部分Ｉ２ａは、インターフェイスＩ１とインラインにコーディングされて、インターフェイスＩ１’’を形成している。具体的な例として、図３のインターフェイスＩ１が、関数呼出しｓｑｕａｒｅ（ｉｎｐｕｔ，ｏｕｔｐｕｔ）を実施し、この呼出しが、インターフェイスＩ２によって受け取られ、インターフェイスＩ２は、第２のコードセグメントによる、ｉｎｐｕｔとともに渡された値の（ｉｎｐｕｔを二乗するための）処理の後、二乗された結果をｏｕｔｐｕｔとともに返す場合を考える。このような場合、第２のコードセグメントによって実施される（ｉｎｐｕｔを二乗する）処理は、インターフェイスへの呼出しなしで、第１のコードセグメントによって実施され得る。

【００３０】

一方のコードセグメントから別のコードセグメントへの通信は、その通信を複数の別々の通信に分けることによって間接的に遂行することができる。このことは、図１０および１１に概略的に示されている。図１０に示すように、第１のインターフェイス、すなわちインターフェイス１上での通信を、異なるインターフェイス、この場合インターフェイス２Ａ、インターフェイス２Ｂ、およびインターフェイス２Ｃというインターフェイスに通信を準拠させるようにコンバートするためのミドルウェアの１つまたは複数の断片（群）（元のインターフェイスから機能性および／またはインターフェイス機能を分離するので、分離インターフェイス（群））が提供される。これは、たとえば、インターフェイス１のプロトコルに従ってオペレーティングシステムと通信するように設計されたアプリケーションのベースがインストールされているが、次いで、オペレーティングシステムが、異

なるインターフェイス、この場合インターフェイス２Ａ、インターフェイス２Ｂ、およびインターフェイス２Ｃというインターフェイスを用いるように変更される場合に行われ得るであろう。重要なのは、第２のコードセグメントによって使われていた元のインターフェイスは、第１のコードセグメントによって使われるインターフェイスとの互換性がなくなるように変更され、したがって、古いおよび新しいインターフェイスを互換可能にするのに媒介が用いられる点である。同様に、図１１に示すように、ＤＩ２と共同で作用するように設計し直されているが同じ機能的結果をもたらす、インターフェイスＩ１からの通信を受けるための分離インターフェイスＤＩ１、およびたとえばインターフェイスＩ２ａおよび１２ｂにインターフェイス機能性を送るための分離インターフェイスＤＩ２を有する、第３のコードセグメントが導入され得る。同様に、ＤＩ１およびＤＩ２は、図３のインターフェイスＩ１およびＩ２の機能性を、新しいオペレーティングシステムに変換するように共同作用し、同じまたは類似した機能的結果をもたらし得る。

【００３１】

インターフェイスの機能性を他のもので置き換えるが全体的に同じ結果を達成するようにコードを動的に書き換えることになるさらに別の変形体が可能である。たとえば、（．Ｎｅｔフレームワーク、Ｊａｖａ（登録商標）ランタイム環境、または他の同様のランタイムタイプの環境によって提供されるような）実行環境において、中間言語（たとえば、マイクロソフトＩＬ、Ｊａｖａ（登録商標）バイトコードなど）の形で示されるコードセグメントが、ジャストインタイム（ＪＩＴ）コンパイラまたはインタープリタに与えられるシステムがあり得る。ＪＩＴコンパイラは、第１のコードセグメントから第２のコードセグメントへの通信を動的にコンバートするように、すなわち、第２のコードセグメント（元のまたは異なる第２のコードセグメントのどちらか）によって要求され得る異なるインターフェイスに通信を準拠させるように書かれることができる。このことは、図１２および１３に示されている。図１２に見られ得るように、この手法は、上述された分離シナリオと似ている。これは、たとえば、インターフェイス１のプロトコルに従ってオペレーティングシステムと通信するように設計されたインストール済みのアプリケーションのベースが、次いで、オペレーティングシステムが、異なるインターフェイスを用いるように変更される場合に行われ得るであろう。ＪＩＴコンパイラは、インストールベースのアプリケーションからの実行中（on the fly）の通信を、オペレーティングシステムの新しいインターフェイスに準拠させるのに用いられ得る。図１３に示されるように、インターフェイス（群）を動的に書き換えるこの手法は、インターフェイス（群）を動的に因数分解し、あるいは作り変えるのにも適用され得る。

【００３２】

代替実施形態によって、インターフェイスと同じまたは類似の結果を達成する上述したシナリオは、様々な方法、すなわち直列および／または並列で、あるいは他の介在コードと組み合わされてもよいことにも留意されたい。したがって、上で提示された代替実施形態は、互いに排他的でなく、図２および３に示される汎用的なシナリオと同じまたは等価なシナリオを作り出すように、混合され、調和され、組み合わされ得る。ほとんどのプログラミング構成の場合のように、本明細書では説明されないが、それにもかかわらず本発明の精神および範囲によって表されるインターフェイスの同じまたは類似の機能性を達成する他の同様の方法があることにも留意されたい。すなわち、インターフェイスの価値を決定するのは、少なくとも一部は、インターフェイスによって表される機能性、およびインターフェイスによって可能にされる有利な結果であることに留意されたい。

【００３３】

ユーザインターフェイスは、アプリケーションの稼動およびオペレーティングシステムの管理に必要なオブジェクトへのアクセスを、ユーザに許可する。こうしたオブジェクトは、コンピュータのディスクドライブ上に常駐し得るフォルダおよびファイルを含み得る。シェルは、こうしたオブジェクトを、ユーザインターフェイスを介して、またはアプリケーションを介して階層状の名前空間構造に系統化する。シェルは、システムにロケーションおよび存在が知られ得るとともにそれに対するアクセスがスタートメニューなどシェ

ル中の数多くの場所から与えられる特殊なフォルダを含み得る。

【００３４】

本発明の態様において、ＧＵＩＤ（グローバルに一意の識別子）は、システム上の個別の既知の各フォルダを指定するのに使用され得る。既知のフォルダは、仮想フォルダ、固定ファイルシステムフォルダ、共通フォルダ、およびユーザごとのフォルダを含む４つのカテゴリの１つに属し得る。

【００３５】

仮想フォルダは、シェル名前空間中に現れる仮想シェルフォルダでよく、関連づけられた実際のどのファイルシステムフォルダももたなくてよい。たとえば、制御パネルフォルダおよびプリンタフォルダは、実際のどのファイルシステムフォルダにもバックアップされないが、シェル仮想名前空間中にのみ存在する仮想フォルダであり得る。固定ファイルフォルダは、シェルによって管理されず、システムがインストールされる際にそのロケーションが固定されるファイルシステムフォルダであり得る。たとえば、「Ｗｉｎｄｏｗｓ（登録商標）」フォルダおよび「Ｐｒｏｇｒａｍ　Ｆｉｌｅｓ」フォルダは、固定フォルダである。共通フォルダは、ユーザ間のデータおよび設定の共有に使われ得るファイルシステムフォルダでよい。たとえば、あるマシンの全ユーザが、共通デスクトップフォルダを共有することができる。最終的に、ユーザごとのフォルダは、個人のプロファイルの下に置かれ、個人ユーザによって所有されるファイルシステムフォルダであり得る。たとえば、「％ＵＳＥＲＰＲＯＦＩＬＥ％＼Ｐｉｃｔｕｒｅｓ」は、現在のユーザのピクチャ用のフォルダである。

【００３６】

本発明の態様において、Ｗｉｎ３２およびＣｏｍ　ＡＰＩ両方に対して、ｋｎｏｗｎｆｏｌｄｅｒ機能性が提供され得る。Ｗｉｎ３２　ＡＰＩは、ＳＨＦｏｌｄｅｒＰａｔｈ　ＡＰＩとの下位互換性を提供し得る。ＳＨＦｏｌｄｅｒＰａｔｈ　ＡＰＩは、そうしたそれぞれのフォルダ向けの新規ＦＯＬＤＥＲＩＤに対する、ＣＳＩＤＬのハードコードされたマッピングリストを有するＣｏｍ　ＡＰＩのラッパーであり得る。

【００３７】

下位互換性をサポートするために、ｋｎｏｗｎｆｏｌｄｅｒインターフェイスは、ＳＨＧｅｔＦｏｌｄｅｒＬｏｃａｔｉｏｎＥｘ（）、ＳＨＧｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）、およびＳＨＳｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）を含む３つのＷｉｎ３２　ＡＰＩコールをサポートすることができる。

【００３８】

ＳＨＧｅｔＦｏｌｄｅｒＬｏｃａｔｉｏｎＥｘ（）ＡＰＩは、ＳＨＧｅｔＦｏｌｄｅｒｌｏｃａｔｉｏｎ（）ＡＰＩを包含し、指定されたＫｎｏｗｎＦｏｌｄｅｒＩＤに対するｋｎｏｗｎｆｏｌｄｅｒ　ＰＩＤＬの取得などの付加能力、および／または要求されたｋｎｏｗｎｆｏｌｄｅｒがまだ存在しない場合は、その作成を指定する能力を呼出し側に与えることができる。ＳＨＧｅｔＦｏｌｄｅｒＬｏｃａｔｉｏｎＥｘ（）ＡＰＩ１４００は、図１４に示される。ＳＨＧｅｔＦｏｌｄｅｒＬｏｃａｔｉｏｎＥＸ（）ＡＰＩ１４００パラメータは、ｒｆｉｄパラメータ１４０１を含み得る。ｒｆｉｄパラメータ１４０１は、既知のフォルダ向けのＧＵＩＤ識別を表し得る。ｄｗＦｌａｇｓパラメータ１４０２は、フォルダが参照されたときに実施されるべき特殊なオプションを指定することができる。ｄｗＦｌａｇパラメータ１４０２のデフォルト値はゼロでよい。ｄｗＦｌａｇパラメータ１４０２の他の例示的な値は、下の表１に示される。

【００３９】

【表１】

表１

dwFlag値	コメント
KF_FLAG_CREATE	フラグが指定されると、APIは、フォルダが存在しないときはそのフォルダの作成を試みる。
KF_FLAG_DONT_VERIFY	フラグが指定されると、APIは、フォルダパスが格納されることを検証しない。
KF_FLAG_NO_ALIAS	フラグが指定されると、APIは、別名をつけられたPIDLのマップを試みない。
KF_FLAG_PER_USER_INIT	ユーザごとの初期化が稼動するべきと特定する。

ｈＴｏｋｅｎパラメータ１４０３は、ユーザごとの既知のフォルダの所有者を指定することができる。既知のフォルダのいくつか、たとえば、「マイドキュメント」フォルダが、ユーザごとのフォルダである。すべてのユーザは、自分の「マイドキュメント」フォルダに対する異なるパスを有し得る。ｈＴｏｋｅｎパラメータ１４０３がヌルの値をもつ場合、ＡＰＩは、フォルダの現在のユーザ（呼出し側）のインスタンスへのアクセスを試みることができる。ｈＴｏｋｅｎパラメータ１４０３が有効なユーザトークンをもつ場合、ＡＰＩは、このトークンを使って、ユーザの役割を演ずることを試み、そのユーザのインスタンスへのアクセスを試みる。さらに、ｈＴｏｋｅｎパラメータ１４０３が－１の値をもつ場合、ＡＰＩは、デフォルトのユーザのフォルダへのアクセスを試みることができる。ｐｐｉｄｌパラメータ１４０４は、要求された既知のフォルダのＰＩＤＬを返し得る。

【００４０】

ＳＨＧｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）ＡＰＩは、ＳＨＧｅｔＦｏｌｄｅｒＰａｔｈ（）ＡＰＩを包含し、既知のフォルダの実際のファイルシステムパスを呼出し側に与え得る。ＳＨＧｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）ＡＰＩ１５００は、図１５に示されている。ＳＨＧｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）ＡＰＩ１５００パラメータは、ｒｆｉｄパラメータ１４０１、ｄｗＦｌａｇｓパラメータ１４０２、およびｈＴｏｋｅｎパラメータ１４０３など、ＳＨＧｅｔＦｏｌｄｅｒＬｏｃａｔｉｏｎＥｘ（）ＡＰＩ１４００で上述されたのと同様のパラメータを含み得る。さらに、ＳＨＧｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）ＡＰＩ１５００は、ｐｓｚＰａｔｈパラメータ１５０８およびｃｃｈＰａｔｈパラメータ１５０９も含み得る。ｐｓｚＰａｔｈパラメータ１５０８は、既知のフォルダのパスを返すことができ、ｃｃｈＰａｔｈパラメータ１５０９は、ｐｓｚＰａｔｈパラメータ１５０８のバッファサイズを指定することができる。さらに、ＳＨＧｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）ＡＰＩ１５００は、下の表２に示されるように、使用可能な追加のｄｗＦｌａｇ値を有し得る。

【００４１】

【表２】

表２

dwFlag値	コメント
KF_FLAG_DEFAULT_PATH	一部の既知のフォルダは、デフォルトのロケーションとは異なるロケーションにリダイレクトされ得る。フラグが何も指定されていない場合、APIは、フォルダの現在の（リダイレクトされる）パスを返さない。フラグが指定されると、APIは、指定された既知のフォルダの元のデフォルトのロケーションを返す。
KF_FLAG_NOT_PARENT_RELATIVE	KF_FLAG_DEFAULT_PATHとともに使われて、その親の現在のロケーションに依存しない「名前空間の元の」デフォルトのパスを与える。
KF_FLAG_SHARE_PATH	このフォルダに対応するリモート共有パス、たとえば、\\computer\users$\ming\Documentsを指定するのに使われる。

ＳＨＳｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）ＡＰＩ１６００は、ＳＨＳｅｔＦｏｌｄｅｒＰａｔｈ（）ＡＰＩを包含し、所与の既知のフォルダに対するパスを呼出し側に設定させることができる。ＳＨＳｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）ＡＰＩ１６００は、図１６に示される。ＳＨＳｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）ＡＰＩ１６００パラメータは、ｒｆｉｄパラメータ１４０１、ｄｗＦｌａｇｓパラメータ１４０２、およびｈＴｏｋｅｎパラメータ１４０３など、ＳＨＧｅｔＦｏｌｄｅｒＬｏｅａｔｉｏｎＥｘ（）ＡＰＩ１４００で上述されたのと同様のパラメータを含み得る。さらに、ＳＨＳｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）ＡＰＩ１６００は、ｋｎｏｗｎｆｏｌｄｅｒに対する、リダイレクトされたパスを指定するのに使われ得るｐｓｚＰａｔｈパラメータ１６０８も含み得る。さらに、ＳｈＳｅｔＦｏｌｄｅｒＰａｔｈＥｘ（）ＡＰＩ１６００は、下の表３に示されるように、使用可能な追加のｄｗＦｌａｇ値を有し得る。

【００４２】

【表３】

表３

dwFlag値	コメント
KF_FLAG_DONT_UNEXPAND	与えられた通りにレジストリに文字列が確実に書き込まれるように、KF_FLAG_DONT_UNEXPAND値を追加する。KF_FLAG_DONT_UNEXPANDフラグが含まれない場合、パスのいくつかの部分が、%USERPROFILE%などの環境文字列で置き換えられ得る。

上で述べたように、本発明の別の態様では、ｋｎｏｗｎｆｏｌｄｅｒ機能性は、ＣｏｍＡＰＩによって提供され得る。ＣＯＭインターフェイスは、ＩＫｎｏｗｎＦｏｌｄｅｒ

ＡＰＩ、ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ ＡＰＩ、ＩＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒインターフェイス、およびＩＫｎｏｗｎＦｏｌｄｅｒＨａｎｄｌｅｒを含み得る。

【００４３】

図１７に示されるように、ＩＫｎｏｗｎＦｏｌｄｅｒ ＡＰＩ１７００は、定義済みのｋｎｏｗｎｆｏｌｄｅｒ用のＧＵＩＤ値および／またはＰＩＤＬ値を得る能力を、アプリケーションに与えることができる。さらに、ＩＫｎｏｗｎＦｏｌｄｅｒ ＡＰＩ１７００は、定義済みのｋｎｏｗｎｆｏｌｄｅｒに対するパスを得ることも、設定することもできる。ＩＫｎｏｗｎＦｏｌｄｅｒ ＡＰＩパラメータは、ＧｅｔＩＤ（）パラメータ１７０１を含み得る。ＧｅｔＩＤ（）パラメータ１７０１は、指定されたｋｎｏｗｎｆｏｌｄｅｒ用のＧＵＩＤを取得することができる。ＧｅｔＣａｔｅｇｏｒｙ（）パラメータ１７０２は、指定されたｋｎｏｗｎｆｏｌｄｅｒ用のｋｎｏｗｎｆｏｌｄｅｒカテゴリを取得することができる。ｋｎｏｗｎｆｏｌｄｅｒカテゴリは、仮想フォルダカテゴリ、固定ファイルシステムカテゴリ、共通フォルダカテゴリ、および／またはユーザごとのフォルダカテゴリを含み得る。

【００４４】

ＩＫｎｏｗｎＦｏｌｄｅｒ ＡＰＩ１７００の追加パラメータは、ＧｅｔＰａｔｈ（）パラメータ１７０３、ＳｅｔＰａｔｈ（）パラメータ１７０４、ＧｅｔＬｏｃａｔｉｏｎ（）パラメータ１７０５、およびＧｅｔＩｔｅｍ（）パラメータ１７０６を含み得る。ＧｅｔＰａｔｈ（）パラメータ１７０３は、所与のｋｎｏｗｎｆｏｌｄｅｒに対するパスを取得することができる。ＳｅｔｈＰａｔｈ（）パラメータ１７０４は、ｋｎｏｗｎｆｏｌｄｅｒに対するパスを設定することができる。ＧｅｔＬｏｃａｔｉｏｎ（）パラメータ１７０５は、ｋｎｏｗｎｆｏｌｄｅｒに関連づけられたＰＩＤＬを与えることができ、ＧｅｔＩｔｅｍ（）パラメータ１７０６は、指定されたフォルダに関連づけられたシェルインターフェイスを取得することができる。

【００４５】

さらに、ＩＫｎｏｗｎＦｏｌｄｅｒ ＡＰＩ１７００によってリダイレクションが提供され得る。リダイレクションは、ＩｓＲｅｄｉｒｅｃｔａｂｌｅ（）パラメータ１７０７、ＩｓＶａｌｉｄＦｏｌｄｅｒＰａｔｈ（）パラメータ１７０８、Ｒｅｄｉｒｅｃｔ（）パラメータ１７０９、およびＲｅｄｉｒｅｃｔＷｉｔｈＵＩ（）パラメータ１７１０の使用により、指定され得る。ＩｓＲｅｄｉｒｅｃｔａｂｌｅ（）パラメータ１７０７は、指定された既知のフォルダがリダイレクトされることを許可されるかどうか確かめるために提供され得る。ＩｓＶａｌｉｄＦｏｌｄｅｒＰａｔｈ（）パラメータ１７０８は、与えられたパスがリダイレクション用に有効なパスであるかどうか検証することができる。Ｒｅｄｉｒｅｃｔ（）パラメータ１７０９は、指定されたｋｎｏｗｎｆｏｌｄｅｒを指定されたパスへリダイレクトすることができる。ＲｅｄｉｒｅｃｔＷｉｔｈＵＩ（）パラメータ１７１０は、ｋｎｏｗｎｆｏｌｄｅｒを指定されたパスへリダイレクトする間、ユーザインターフェイスを表示することができる。

【００４６】

本発明の別の態様では、ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ ＡＰＩが提供され得る。図１８に示されるように、ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ ＡＰＩ１８００は、ｋｎｏｗｎｆｏｌｄｅｒを作成し、または削除する能力を与えることができる。ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ ＡＰＩ１８００は、ｋｎｏｗｎｆｏｌｄｅｒの記述、ｋｎｏｗｎｆｏｌｄｅｒのカテゴリ、ｋｎｏｗｎｆｏｌｄｅｒの所有権情報、および／またはｋｎｏｗｎｆｏｌｄｅｒの相対パスなど、ｋｎｏｗｎｆｏｌｄｅｒの定義を管理することもできる。さらに、ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ ＡＰＩ１８００は、コンピューティングシステムにおいて、またはコンピューティングシステム環境において利用可能な全ｋｎｏｗｎｆｏｌｄｅｒを列挙する能力も与え得る。たとえば、ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ ＡＰＩ１８００は、コンピュータネットワークへのアクセスを有するユーザに利用可能な、周知のフォルダをすべて列挙することができ

る。

【００４７】

本発明の態様によると、ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ　ＡＰＩ１８００は、いくつかのパラメータを含み得る。たとえば、ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ　ＡＰＩ１８００は、ＦｏｌｄｅｒＩｄＦｒｏｍＣＳＩＤＬ（）パラメータ１８０１を含み得る。ＦｏｌｄｅｒＩｄＦｒｏｍＣＳＩＤＬ（）パラメータ１８０１は、指定されたＣＳＩＤＬに関連づけられたＫｎｏｗｎＦｏｌｄｅｒ＿ＩＤを取得するのに使われ得る。ＦｏｌｄｅｒＩｄＦｒｏｍＣＳＩＤＬ（）パラメータ１８０１はしたがって、ＣＳＩＤＬとＫｎｏｗｎＦｏｌｄｅｒ＿ＩＤとの間の変換を提供し得る。同様に、ＦｏｌｄｅｒＩｄＴｏＣＳＩＤＬ（）パラメータ１８０２も、指定されたＫｎｏｗｎＦｏｌｄｅｒ＿ＩＤ用のＣＳＩＤＬ値を取得するように定義され得る。

【００４８】

ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ　ＡＰＩ１８００のパラメータとして、ＧｅｔＦｏｌｄｅｒ（）パラメータ１８０３も定義され得る。ＧｅｔＦｏｌｄｅｒ（）パラメータ１８０３は、ｋｎｏｗｎｆｏｌｄｅｒのＩＤが入手可能な場所で、特定のｋｎｏｗｎｆｏｌｄｅｒ向けの情報を直接取得するのに使用され得る。ＧｅｔＦｏｌｄｅｒ（）パラメータ１８０３は、ｋｎｏｗｎｆｏｌｄｅｒ用のＧＵＩＤ値、ｋｎｏｗｎｆｏｌｄｅｒカテゴリ、ｋｎｏｗｎｆｏｌｄｅｒパス、および／またはｋｎｏｗｎｆｏｌｄｅｒに関連づけられたＰＩＤＬなどの情報を取得するために、ＩＫｎｏｗｎＦｏｌｄｅｒポインタを返し得る。ＧｅｔＦｏｌｄｅｒ（）パラメータ１８０３と同様、ある特定のユーザに属すｋｎｏｗｎｆｏｌｄｅｒに対するパスを得る能力を呼出し側に与えるＧｅｔＦｏｌｄｅｒＦｏｒＵｓｅｒ（）パラメータ１８０４も定義され得る。

【００４９】

図１８を参照すると、指定されたＫｎｏｗｎＦｏｌｄｅｒ＿ＩＤに関連づけられた全プロパティを取得するために、ＧｅｔＦｏｌｄｅｒＤｅｆｉｎｉｔｉｏｎ（）パラメータ１８０５が提供され得る。本発明の態様によると、ＫｎｏｗｎＦｏｌｄｅｒ＿Ｄｅｆｉｎｉｔｉｏｎ構造体１９００が、図１９に示されるように定義され得る。ＫｎｏｗｎＦｏｌｄｅｒ＿Ｄｅｆｉｎｉｔｉｏｎ構造体１９００は、図１９に示され、さらに表４に記載されるように、いくつかの定義済みフィールドを含み得る。こうしたフィールドは、ｃａｔｅｇｏｒｙ１９１０、ｐｓｚＮａｍｅ１９１１、ｐｓｚＣｒｅａｔｏｒ１９１２、ｐｓｚＤｅｓｃｒｉｐｔｉｏｎ１９１３、ｐｆｉｄＰａｒｅｎｔ１９１４、ｐｓｚＲｅｌａｔｉｖｅＰａｔｈ１９１５、ｐｓｚＰａｒｓｉｎｇＮａｍｅ１９１６、ｐｓｚＴｏｏｌｔｉｐ１９１７、ｐｓｚＬｏｃａｌｉｚｅｄＮａｍｅ１９１８、ｐｓｚＩｃｏｎ１９１９、ｐｓｚＳｅｃｕｒｉｔｙ１９２０、ｄｗＡｔｔｒｉｂｕｔｅｓ１９２１、ｐｓｚＬｅｇａｃｙＰａｔｈ１９２２、ｃｌｓｉｄＨａｎｄｌｅｒ１９２３、およびｋｆｄＦｌａｇｓ１９２４を含み得る。

【００５０】

【表４】

表4

フィールド	説明
Category	指定されたフォルダが、どのknownfolderカテゴリに属し得るかを示す。
pszName	既知のフォルダ用の非ローカライズ名を与える。
pszCreator	既知のフォルダを定義するアプリケーション名。
pszDescription	既知のフォルダが表す内容の簡単な記述。
pfidParent & pszRelativePath	共通およびユーザごとのフォルダとともに使用されて、ある特定のknownfolderが別のknownfolderの下に作成されるべきであることを示し得る。
pszParsingName	このフィールドは、仮想フォルダ用に使用されて、シェル名前空間フォルダパスを示し得る。
pszTooltip	このフィールドは、APIによってフォルダが作成される際にknownfolder用に使われる、デフォルトのツールチップを表し得る。
pszLocalizedName	APIによってフォルダが作成される際に、このknownfolder用に使われるデフォルトのローカライズ名。
pszIcon	APIによってフォルダが作成される際に、knownfolder用に使われるデフォルトのアイコン。
pszSecurity	APIによってフォルダが作成される際に、この既知のフォルダ用に使われるデフォルトのセキュリティ記述子を記述するためのSDDL形式文字列。ヌルの場合、この既知のフォルダのセキュリティは、親から継承され得る。
dwAttributes	このフォルダが作成される際に設定される、読取り専用などのデフォルトのファイルシステム属性。
pszLegacyPath	フォルダに対するレガシーパスを与える。移行／ローミングおよびアプリケーション互換性において使われ得る。
clsidHandler	この既知のフォルダの作成および削除など、あるイベントにおいて、何らかのカスタマイズ作業を行うために呼び出され得るIKnownFolderHandlerオブジェクトを指定し得る。
dwDefinitionFlags	いくつかのフラグを定義して、このknownfolderに対する比較的些細な挙動を記述する。こうしたフラグは、以下を含み得る。 KFDF_PERSONALIZE：フォルダ用の個別化名を表示する、 KFDF_LOCAL_REDIRECT_ONLY：ローカルディスクへリダイレクトされ得る、 KFDF_ROAMABLE：PCを介してPC sycnへローミングされ得る。

図１８を参照すると、ＲｅｇｉｓｔｅｒＦｏｌｄｅｒ（）パラメータ１８０６およびＵｎｒｅｇｉｓｔｅｒＦｏｌｄｅｒ（）パラメータ１８０７などのパラメータが、システム用のｋｎｏｗｎｆｏｌｄｅｒを作成し、または削除するために与えられ得る。Ｒｅｇｉｓ

ｔｅｒＦｏｌｄｅｒ（）パラメータ１８０６は、ユーザまたは呼出し側に、有効なＫｎｏｗｎＦｏｌｄｅｒ＿Ｄｅｆｉｎｉｔｉｏｎを指定するよう求め得る。ＵｎｒｅｇｉｓｔｅｒＦｏｌｄｅｒ（）パラメータ１８０８は、要求があったときにＫｎｏｗｎＦｏｌｄｅｒ＿Ｄｅｆｉｎｉｔｉｏｎ定義を削除することができる。

【００５１】

ＧｅｔＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒｓ（）パラメータ１８０９およびＧｅｔＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒｓＦｏｒＵｓｅｒ（）パラメータ１８１０などの追加パラメータも、ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ　ＡＰＩ１８００と使用するために定義され得る。ＧｅｔＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒｓ（）パラメータ１８０９は、システム上の全ｋｎｏｗｎｆｏｌｄｅｒを列挙するためのポインタを返すことができ、ＧｅｔＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒｓＦｏｒＵｓｅｒ（）パラメータ１８１０は、ある特定のユーザ用のｋｎｏｗｎｆｏｌｄｅｒを列挙するための能力を呼出し側に与えることができる。最終的に、ＦｉｎｄＦｏｌｄｅｒＦｒｏｍＰａｔｈ（）パラメータ１８１１は、与えられたファイルシステムパスに対する関連づけられた既知のフォルダＩＤを取得するための既知のフォルダポインタを返すことができる。

【００５２】

本発明の別の態様では、ＩＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒ（）ＡＰＩが提供され得る。図２０に示されるように、ＩＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒ（）ＡＰＩ２０００は、ｋｎｏｗｎｆｏｌｄｅｒを列挙するための能力を提供し得る。ＧｅｔＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒｓ（）パラメータ１８０９およびＧｅｔＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒｓＦｏｒＵｓｅｒ（）パラメータ１８１０は、システム上の全ｋｎｏｗｎｆｏｌｄｅｒの列挙を得るためのＩＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒ（）ＡＰＩ２０００へのポインタを返すことができる。ＩＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒ（）ＡＰＩ２０００は、Ｎｅｘｔ（）２００１、Ｓｋｉｐ（）２００２、Ｒｅｓｅｔ（）２００３、およびＣｌｏｎｅ（）２００４などのパラメータを含み得る。Ｎｅｘｔ（）パラメータ２００１は、列挙シーケンス中の指定された数の項目識別子を取得し、取り出された項目の数だけ、現在の位置を進めることができる。Ｓｋｉｐ（）パラメータ２００２は、列挙シーケンス中の指定された数の要素をスキップすることができる。Ｒｅｓｅｔ（）パラメータ２００３は、列挙シーケンスの先頭にインターフェイスを戻すことができる。Ｃｌｏｎｅ（）パラメータ２００４は、現在のものと同じ内容および状態をもつ新規列挙オブジェクトを作成することができる。

【００５３】

本発明の別の態様では、ＩＫｎｏｗｎＦｏｌｄｅｒＨａｎｄｌｅｒ（）ＡＰＩが提供され得る。図２１に示されるように、ＩＫｎｏｗｎＦｏｌｄｅｒＨａｎｄｌｅｒ（）ＡＰＩ２１００は、ｋｎｏｗｎｆｏｌｄｅｒの作成および／または削除のための特殊コードを追加する能力を、他のコンポーネントに提供することができる。ＩＫｎｏｗｎＦｏｌｄｅｒＨａｎｄｌｅｒ（）ＡＰＩ２１００は、ＧｅｔＤｅｆａｕｌｔＬｏｃａｔｉｏｎ（）２１０１、ＦｏｌｄｅｒＣｒｅａｔｅｄ（）２１０２、およびＦｏｌｄｅｒＲｅｍｏｖｅｄ（）２１０３などのパラメータを含み得る。ＧｅｔＤｅｆａｕｌｔＬｏｃａｔｉｏｎ（）パラメータ２１０１は、ｋｎｏｗｎｆｏｌｄｅｒ用のデフォルトのロケーションを取得することができる。ＦｏｌｄｅｒＣｒｅａｔｅｄ（）パラメータ２１０２は、指定されたｋｎｏｗｎｆｏｌｄｅｒが作成されると、ハンドラを開始することができる。さらに、ＦｏｌｄｅｒＲｅｍｏｖｅｄ（）パラメータ２１０３は、指定されたｋｎｏｗｎｆｏｌｄｅｒが削除されると、ハンドラを開始することができる。このパラメータは、アプリケーションによって作成された既知のフォルダが作成され、または削除されると、カスタムコードを稼動させる能力を、アプリケーションに提供することができる。

【００５４】

図２２は、本発明の態様による、オペレーティングシステム２２０２へのプログラムインターフェイスコールを使用する要求コンポーネント２２０１を示す図である。本発明の本態様において、要求コンポーネント２２０１はアプリケーションであるが、他の実施形

態では、要求コンポーネント２２０１は、図１に示されるように、コンピュータ１１０の周辺のハードウェア内部に統合されてよい。

【００５５】

要求コンポーネント２２０１は、開発者またはユーザによってインストールされると、コンピュータ１１０上の既存のｋｎｏｗｎｆｏｌｄｅｒを作成し、列挙し、または管理することを決定し得る。たとえば、図２２のアプリケーション２２０１は、アプリケーションがインストールされると、レジストリ２２０３に対して直接、新規ｋｎｏｗｎｆｏｌｄｅｒを追加することによって、ユーザプロファイルの中に新規フォルダを作成することができる。さらに、新規ｋｎｏｗｎｆｏｌｄｅｒは、ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ　ＡＰＩなど、ＫｎｏｗｎＦｏｌｄｅｒ　ＡＰＩ２２１０を介して追加されてもよい。開発者またはユーザは、アプリケーション２２０１を介して、新規ｋｎｏｗｎｆｏｌｄｅｒを作成するためのＩＫｎｏｗｎＦｏｌｄｅｒマネージャＡＰＩ１８００などのＡＰＩをコールする（２２５０）ことができる。開発者またはユーザは、アプリケーションを介して、新規ｋｎｏｗｎｆｏｌｄｅｒにとって一意の識別子となるＧＵＩＤ２２５１を与えることができる。開発者またはユーザは、標準プロパティ以外に、新規ｋｎｏｗｎｆｏｌｄｅｒに関連づけられることになる追加プロパティ２２５２を定義することもできる。図２２を参照すると、アプリケーション２２０１は、ＫｎｏｗｎＦｏｌｄｅｒ　ＡＰＩ２２１０へのコールなどのコール２２５０を送信する。入力２２５０に応答して、ＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ　ＡＰＩ１８００は、オペレーティングシステム２２０２に、オペレーティングシステム２２０２のレジストリ２２０３を有する新規ｋｎｏｗｎｆｏｌｄｅｒを登録する。オペレーティングシステムは、それに応答して、新規ｋｎｏｗｎｆｏｌｄｅｒのパス２２６０をアプリケーション２２０１に送り得る。

【００５６】

図２２ａは、本発明の別の態様を示す。図２２ａを参照すると、プログラムインターフェイスコールを使用するアプリケーション２２０１が、既知のフォルダＡＰＩ２２１０を介して、既知のフォルダのロケーション用の識別２２６０とともに要求を送り得る。既知のフォルダＡＰＩ２２１０は、既知のフォルダＩＤの一覧２２８０と、それに対応する記憶位置を求めて、レジストリ２２０３を検索することができる。既知のフォルダＡＰＩ２２１０は、既知のフォルダが置かれ得るローカル記憶装置２２５５などの特定の記憶デバイスへの要求２２６１を介して、レジストリ内で指定された記憶位置が有効であることを検証し得る。ロケーションを検証すると、Ｋｎｏｗｎｆｏｌｄｅｒ　ＡＰＩは、要求された既知のフォルダのパスを、アプリケーションに返し得る。たとえば、マイピクチャ２２７０という既知のフォルダが、コール２２６０を介してアプリケーション２２０１によって要求され得る。既知のフォルダＡＰＩ２２１０は、ピクチャ２２７０という既知のフォルダのロケーションを、レジストリ一覧２２８０の調査により決定することができる。既知のフォルダＡＰＩ２２１０は、ロケーションが実際に存在することを、ローカル記憶装置２２５５への要求２２６１により検証することができ、確認すると、ローカル記憶装置２２５５上のピクチャのロケーション２２６２へのパスを、アプリケーション２２０１に返すことができる。

【００５７】

既知のフォルダ「マイピクチャ」２２７０は、ローカル記憶装置２２５５からネットワーク記憶装置２２５６に移動（２２９０）され得る。既知のフォルダ「マイピクチャ」２２７０を再配置すると、既知のフォルダ「マイピクチャ」２２７０用のロケーションパスを変えることになる。既知のフォルダ「マイピクチャ」２２７０のロケーションは、レジストリ２２０３内でアップデートされ得る。同様に、ウェブサイトロケーション２２５７など別のロケーションへの、既知のフォルダ「マイピクチャ」２２７０の移動２２９５により、既知のフォルダ「マイピクチャ」２２７０の新しい位置で、レジストリ２２０３のアップデートが開始され得る。

【００５８】

図２３は、本発明の態様による、新規ｋｎｏｗｎｆｏｌｄｅｒを作成する方法を示す図

である。図２３を参照すると、オペレーティングシステムなどのコンポーネントが、ステップ２３０１で、ＧＵＩＤを有するコールを、第１のコンポーネントから受け取り得る。第１のコンポーネントは、開発者またはユーザによってインストールされ、または初期化されるアプリケーションプログラムを含み得る。オペレーティングシステムによってコールを受け取ると、オペレーティングシステムは、ステップ２３０２で、アプリケーションプログラムによって与えられたＧＵＩＤを抽出することができる。コールは、新規ｋｎｏｗｎｆｏｌｄｅｒの作成において使用するためのプロパティなどの付加情報も含み得る。プロパティは、ｃａｔｅｇｏｒｙ、ｐｓｚＮａｍｅ、ｐｓｚＣｒｅａｔｏｒ、ｐｓｚＤｅｓｃｒｉｐｔｉｏｎ、ｐｆｉｄＰａｒｅｎｔ、ｐｓｚＲｅｌａｔｉｖｅＰａｔｈ、ｐｓｚＰａｒｓｉｎｇＮａｍｅ、ｐｓｚＴｏｏｌｔｉｐ、ｐｓｚＬｏｃａｌｉｚｅｄＮａｍｅ、ｐｓｚＩｃｏｎ、ｐｓｚＳｅｃｕｒｉｔｙ、ｄｗＡｔｔｒｉｂｕｔｅｓ、ｐｓｚＬｅｇａｃｙＰａｔｈ、ｃｌｓｉｄＨａｎｄｌｅｒ、およびｋｄｆＦｌａｇｓなど、新規ｋｎｏｗｎｆｏｌｄｅｒを定義するための情報を含み得る。

【００５９】

抽出されたＧＵＩＤおよび与えられた付加情報に基づいて、新規ｋｎｏｗｎｆｏｌｄｅｒがステップ２３０３で作成され得る。新規ｋｎｏｗｎｆｏｌｄｅｒは、ステップ２３０４で示されるように、オペレーティングシステムのレジストリ内に含められ得る。オペレーションシステムは、ステップ２４０５で、新規ｋｎｏｗｎｆｏｌｄｅｒのパスをアプリケーションプログラムに送ることができる。アプリケーションプログラムは、ローカルまたはネットワークシステム上の既存のｋｎｏｗｎｆｏｌｄｅｒをすべて列挙することができる。

【００６０】

本発明を実施する、現時点で好まれるモードを含む具体例を参照して本発明が説明されたが、添付の特許請求の範囲で説明されるように、本発明の精神および範囲内である、上述したシステムおよび技法の多数の変形および置換えが存在することが、当業者には理解されよう。

【図面の簡単な説明】

【００６１】

【図１】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の例を示す図である。

【図２】第１および第２のコードセグメントが通信するためのパイプとしてのインターフェイスを示す図である。

【図３】インターフェイスを、インターフェイスオブジェクトを備えるものとして示す図である。

【図４】インターフェイスの通信を多数のインターフェイスにコンバートするために細分され得るインターフェイスによって提供される機能を示す図である。

【図５】多数のインターフェイスに細分され得るインターフェイスによって提供される機能を示す図である。

【図６】プログラミングインターフェイスのアスペクトを無視し、追加し、または再定義しながら、依然として同じ結果を達成する例を示す図である。

【図７】プログラミングインターフェイスのアスペクトを無視し、追加し、または再定義しながら、依然として同じ結果を達成する別の例を示す図である。

【図８】図２に示される例に関連したコードセグメントのマージを示す図である。

【図９】図３に示される例に関連したインターフェイスのマージを示す図である。

【図１０】通信を、異なるインターフェイスに準拠するようにコンバートするミドルウェアを示す図である。

【図１１】分離インターフェイスに関連づけられたコードセグメントを示す図である。

【図１２】アプリケーションのインストールされたベースが、インターフェイスプロトコルに従って、異なるインターフェイスを用いるように変更されるオペレーティングシステムと通信するように設計される例を示す図である。

【図１３】インターフェイスを動的に因数分解し、あるいは改めるための書換えインターフェイスを示す図である。
【図１４】本発明の態様による、指定されたＫｎｏｗｎＦｏｌｄｅｒＩＤに対するＫｎｏｗｎｆｏｌｄｅｒ　ＰＩＤＬの取得などの機能性を提供することができる拡張ＡＰＩを示す図である。
【図１５】本発明の態様による、呼出し側への、Ｋｎｏｗｎｆｏｌｄｅｒの実際のファイルシステムパスの提供などの機能性を提供することができる付加拡張ＡＰＩを示す図である。
【図１６】本発明の態様による、所与の既知のフォルダに対するパスを呼出し側に設定させる機能性を提供することができる第３の拡張ＡＰＩを示す図である。
【図１７】本発明の態様による、アプリケーションへの、定義済みｋｎｏｗｎｆｏｌｄｅｒに対するＧＵＩＤ値および／またはＰＩＤＬ値を得る能力の付与などの機能性を提供することができるＩＫｎｏｗｎＦｏｌｄｅｒ　ＡＰＩを示す図である。
【図１８】本発明の態様による、ｋｎｏｗｎｆｏｌｄｅｒを作成し、または削除する能力を与えることができるＩＫｎｏｗｎＦｏｌｄｅｒＭａｎａｇｅｒ　ＡＰＩを示す図である。
【図１９】本発明の態様によるＫｎｏｗｎＦｏｌｄｅｒ＿Ｄｅｆｉｎｉｔｉｏｎ構造体を示す図である。
【図２０】本発明の態様による、ｋｎｏｗｎｆｏｌｄｅｒを列挙する能力を与えることができるＩＥｎｕｍＫｎｏｗｎＦｏｌｄｅｒ（）ＡＰＩを示す図である。
【図２１】本発明の態様による、他のコンポーネントが、ｋｎｏｗｎｆｏｌｄｅｒの作成および／または削除用の特殊コードを追加するための能力を与えることができるＩＫｎｏｗｎＦｏｌｄｅｒＨａｎｄｌｅｒ（）ＡＰＩを示す図である。
【図２２】本発明の態様による、オペレーティングシステムへのプログラムインターフェイスコールを使用する要求コンポーネントを示す図である。
【図２２ａ】本発明の態様による、アプリケーションが、既知のフォルダのロケーションを決定するために、既知のフォルダの識別とともに要求を送信する、本発明の別の態様を示す図である。
【図２３】本発明の態様による、新規ｋｎｏｗｎｆｏｌｄｅｒを作成する方法を示す図である。

【図１４】

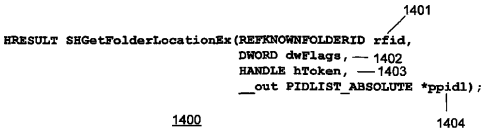


FIGURE 14

【図１５】

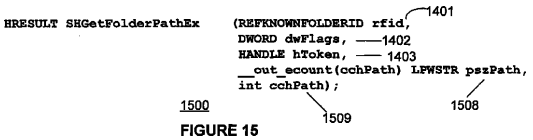


FIGURE 15

【図１６】

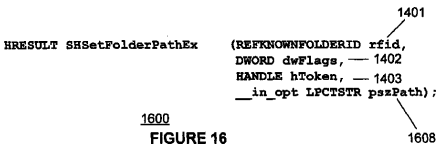


FIGURE 16

【図１７】

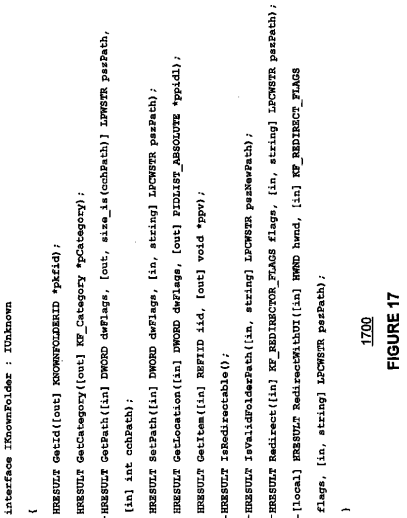


FIGURE 17

【図１８】

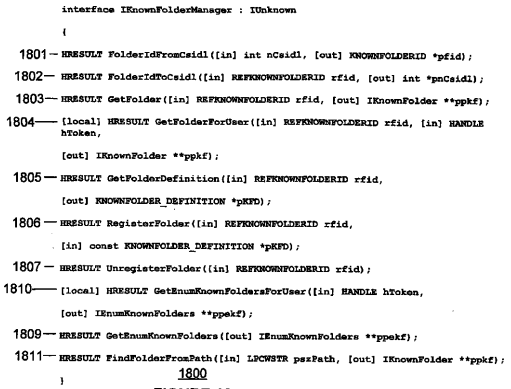
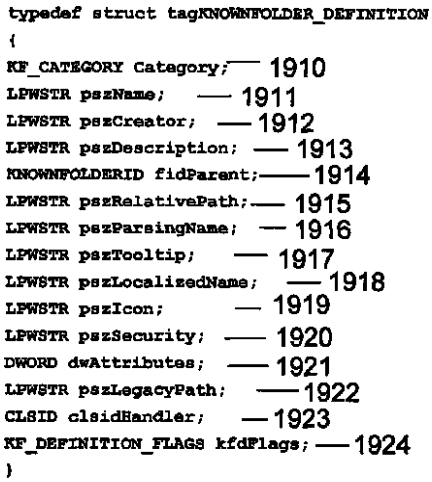


FIGURE 18

【図１９】



1900

FIGURE 19

【図２０】

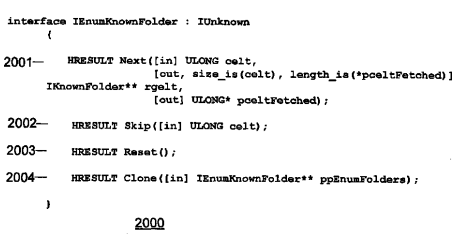


FIGURE 20

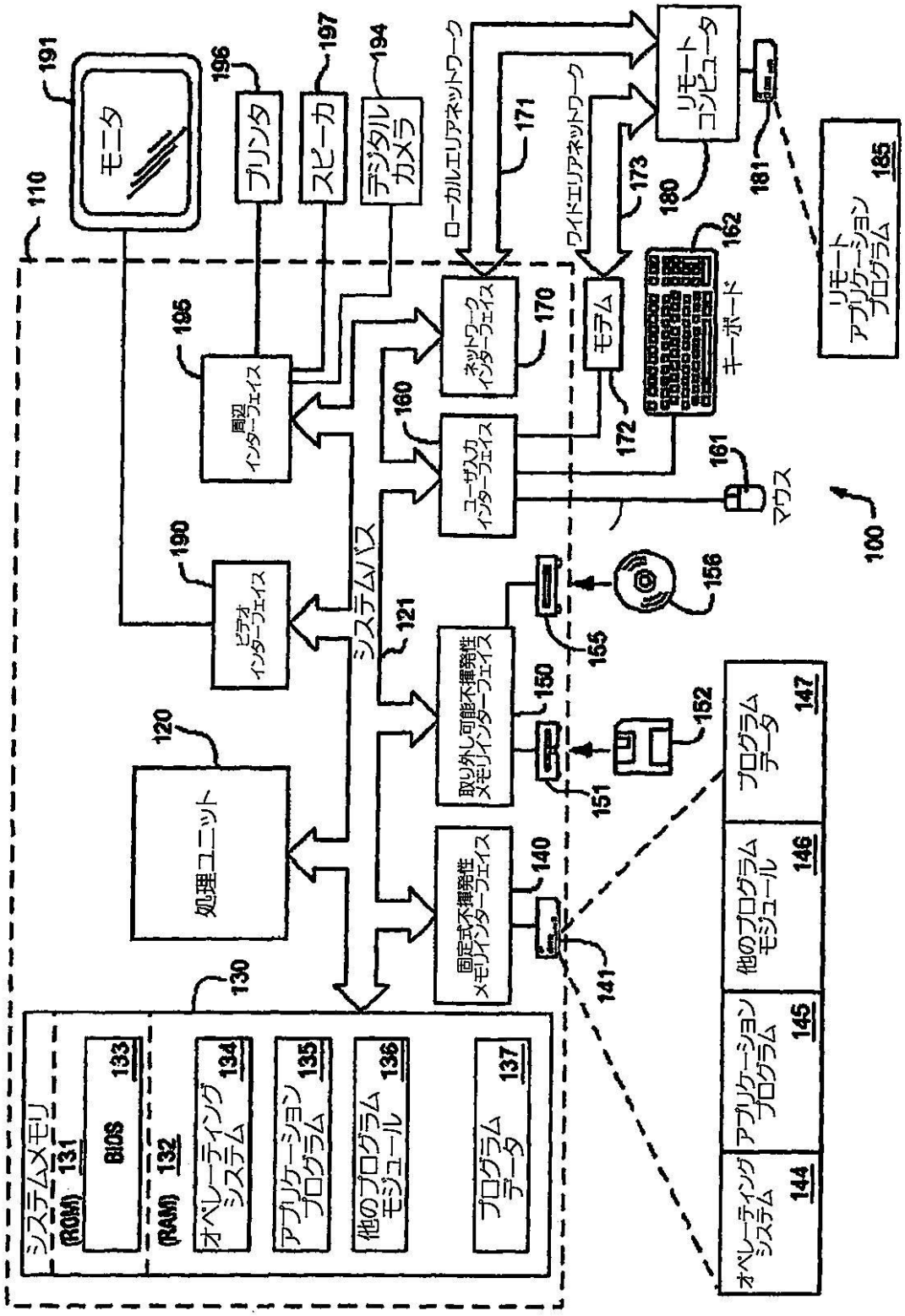
【図２１】

```
interface IKnownFolderHandler : IUnknown
    {
2101—  HRESULT GetDefaultLocation([out, size_is(cchBuffer)]
                                  LPWSTR pszBuffer,
                                  [in] int cchBuffer);

2102—  HRESULT FolderCreated([in] REFKNOWNFOLDERID rfid,
                             [in] DWORD dwFlags,
                             [in] LPCWSTR pszPath);

2103—  HRESULT FolderRemoved([in] REFKNOWNFOLDERID rfid,
                             [in] DWORD dwFlags,
                             [in] LPCWSTR pszPath);
    }
```

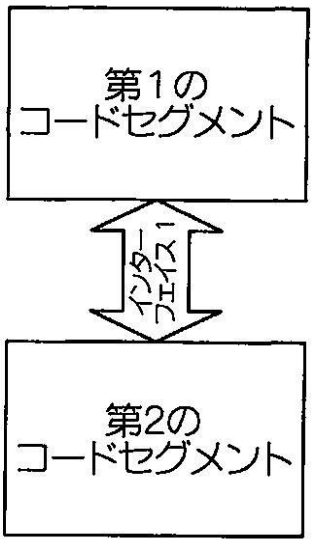
2100

FIGURE 21

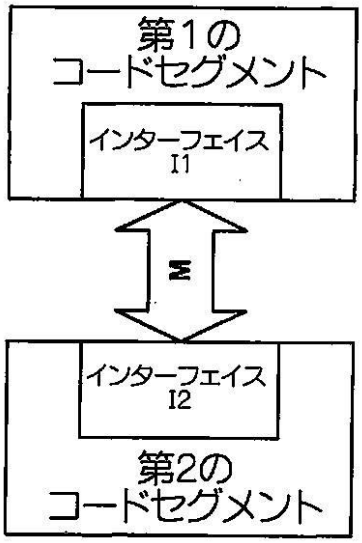
【図１】


モニタ
プリンタ
スピーカ
デジタルカメラ
ローカルエリアネットワーク
ワイドエリアネットワーク
リモートコンピュータ
リモートアプリケーションプログラム
キーボード
マウス
周辺インターフェイス
ネットワークインターフェイス
モデム
ユーザ入力インターフェイス
ビデオインターフェイス
システムバス
処理ユニット
取り外し可能不揮発性メモリインターフェイス
固定式不揮発性メモリインターフェイス
システムメモリ
(ROM)
BIOS
(RAM)
オペレーティングシステム
アプリケーションプログラム
他のプログラムモジュール
プログラムデータ

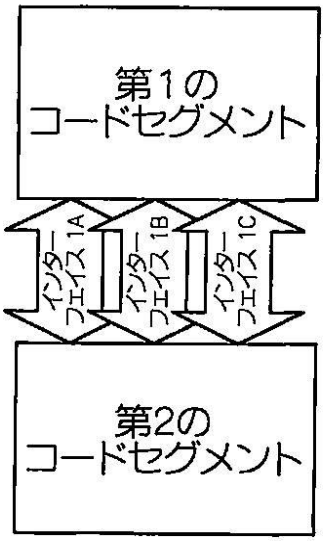
【図２】


第1の
コードセグメント
インターフェイス1
第2の
コードセグメント

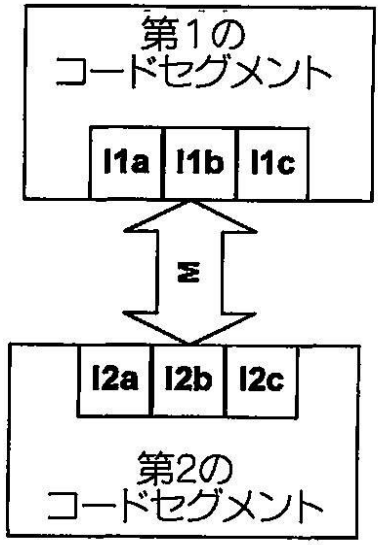
【図３】


第1の
コードセグメント
インターフェイス
I1
M
インターフェイス
I2
第2の
コードセグメント

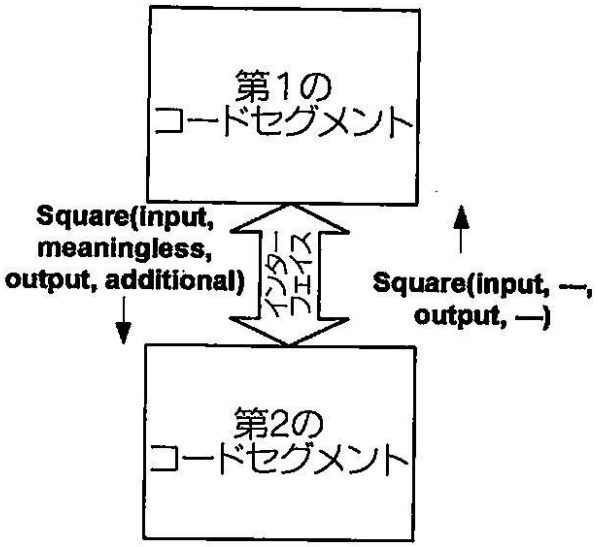
【図４】


第1の
コードセグメント
インターフェイス 1A
インターフェイス 1B
インターフェイス 1C
第2の
コードセグメント

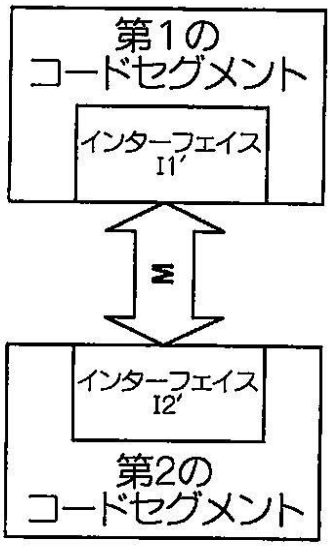
【図５】


第1の
コードセグメント
I1a
I1b
I1c
M
I2a
I2b
I2c
第2の
コードセグメント

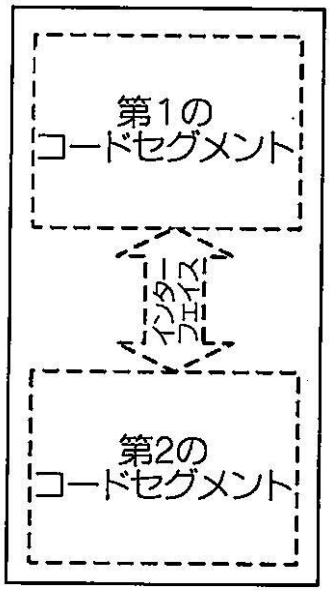
【図６】


第1の
コードセグメント
Square(input,
meaningless,
output, additional)
インターフェイス
Square(input, —,
output, —)
第2の
コードセグメント

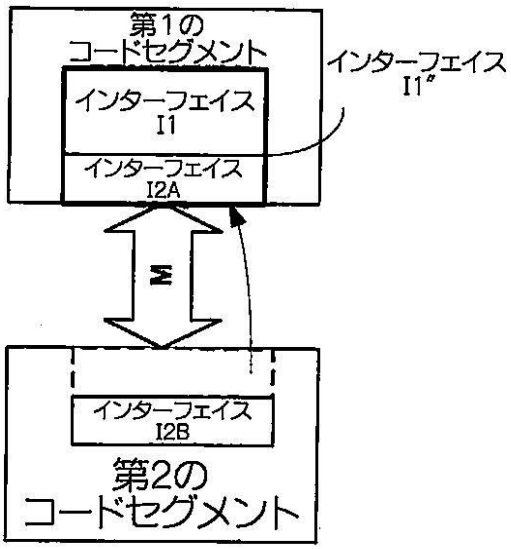
【図７】


第1の
コードセグメント
インターフェイス
I1'
M
インターフェイス
I2'
第2の
コードセグメント

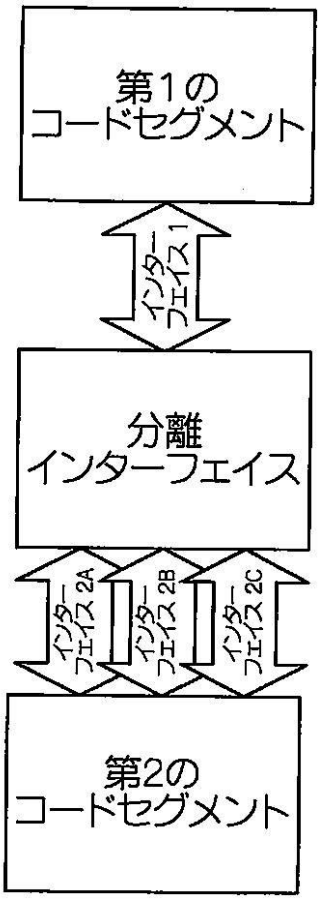
【図８】


第1の
コードセグメント
インターフェイス
第2の
コードセグメント

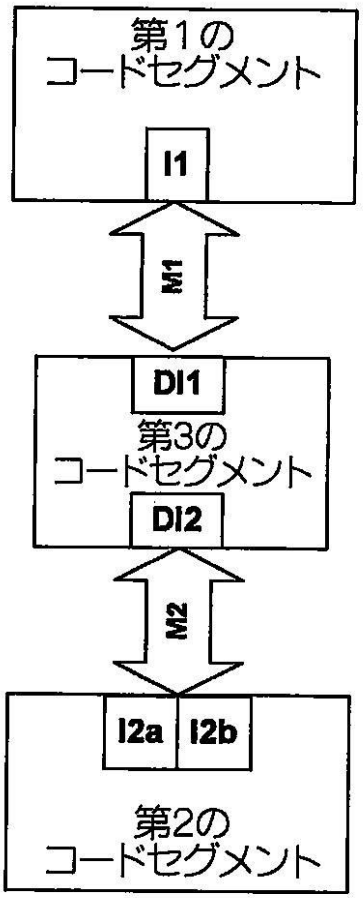
【図９】


第1の
コードセグメント
インターフェイス
I1
インターフェイス
I2A
インターフェイス
I1'
M
インターフェイス
I2B
第2の
コードセグメント

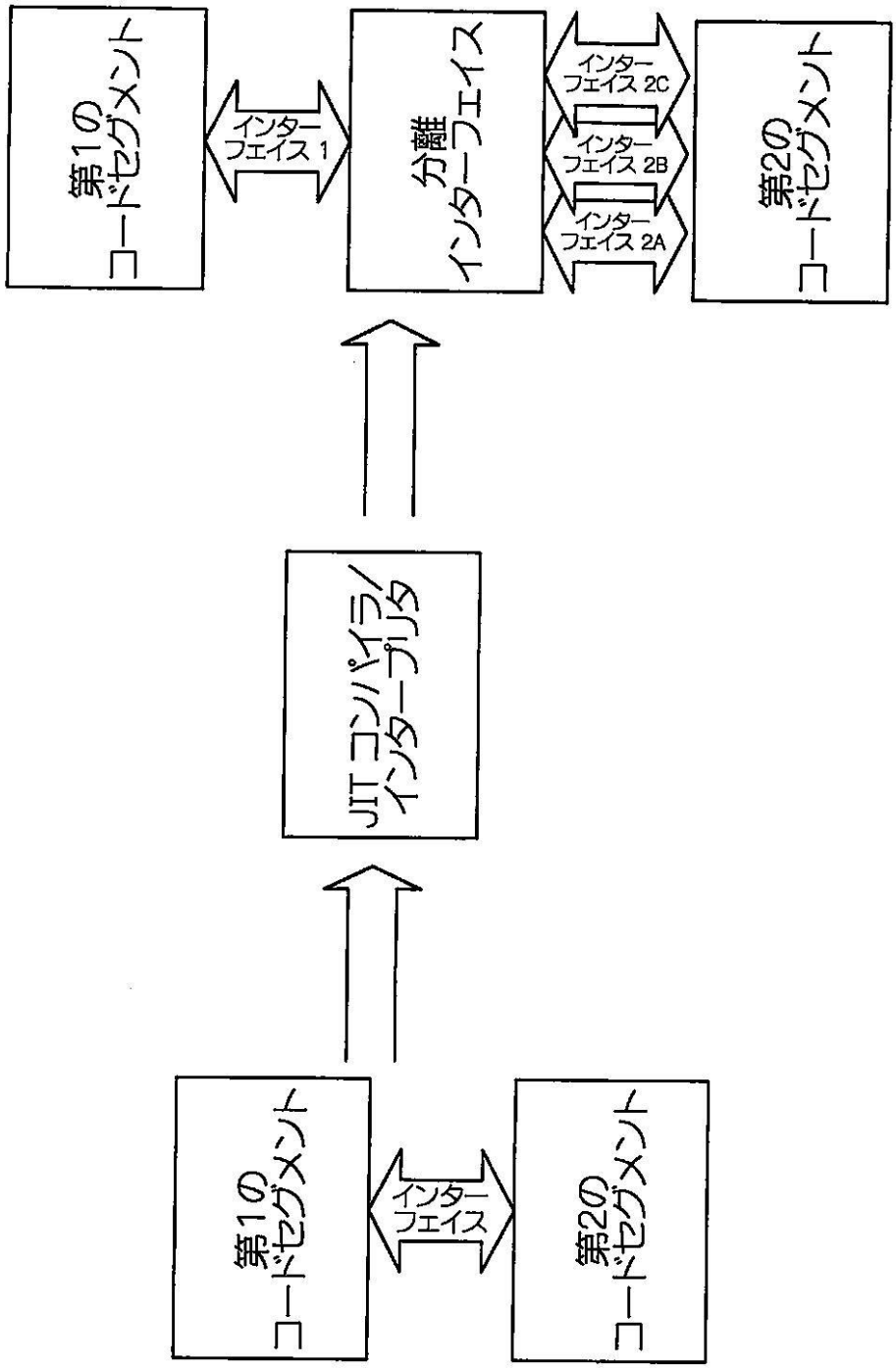
【図１０】


第1の
コードセグメント
インター
フェイス1
分離
インターフェイス
インター
フェイス2A
インター
フェイス2B
インター
フェイス2C
第2の
コードセグメント

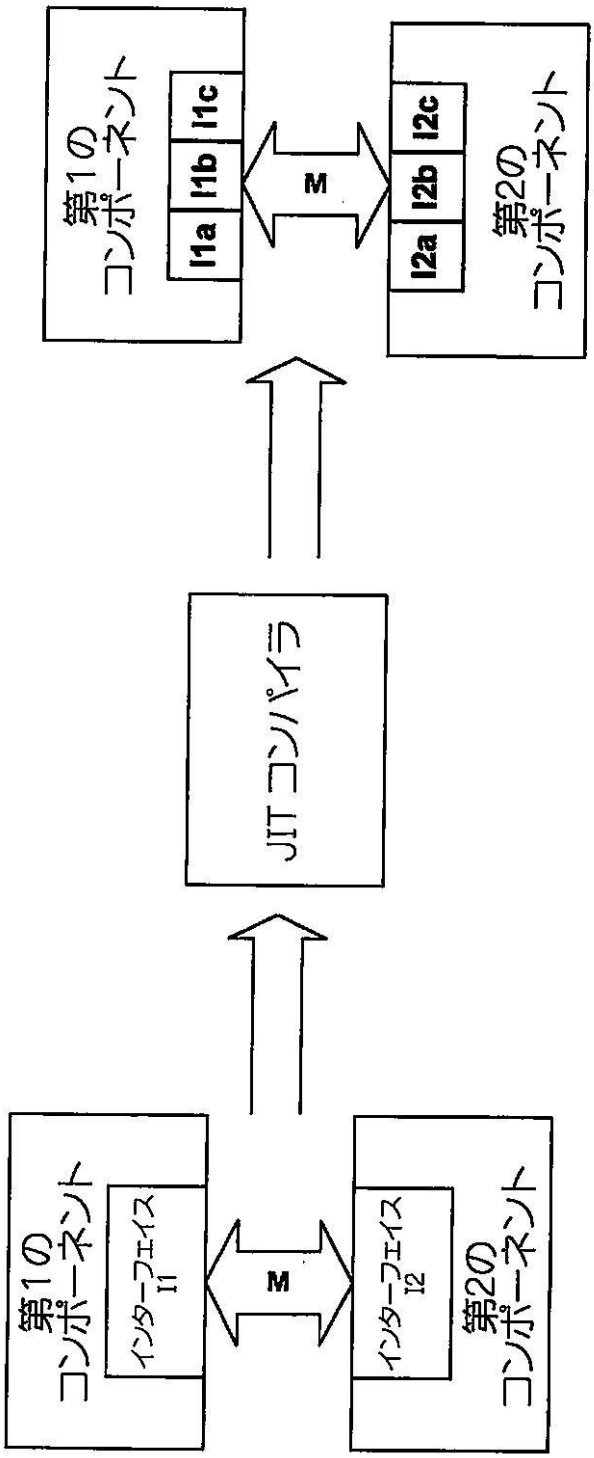
【図１１】


第1の
コードセグメント
I1
M1
DI1
第3の
コードセグメント
DI2
M2
I2a
I2b
第2の
コードセグメント

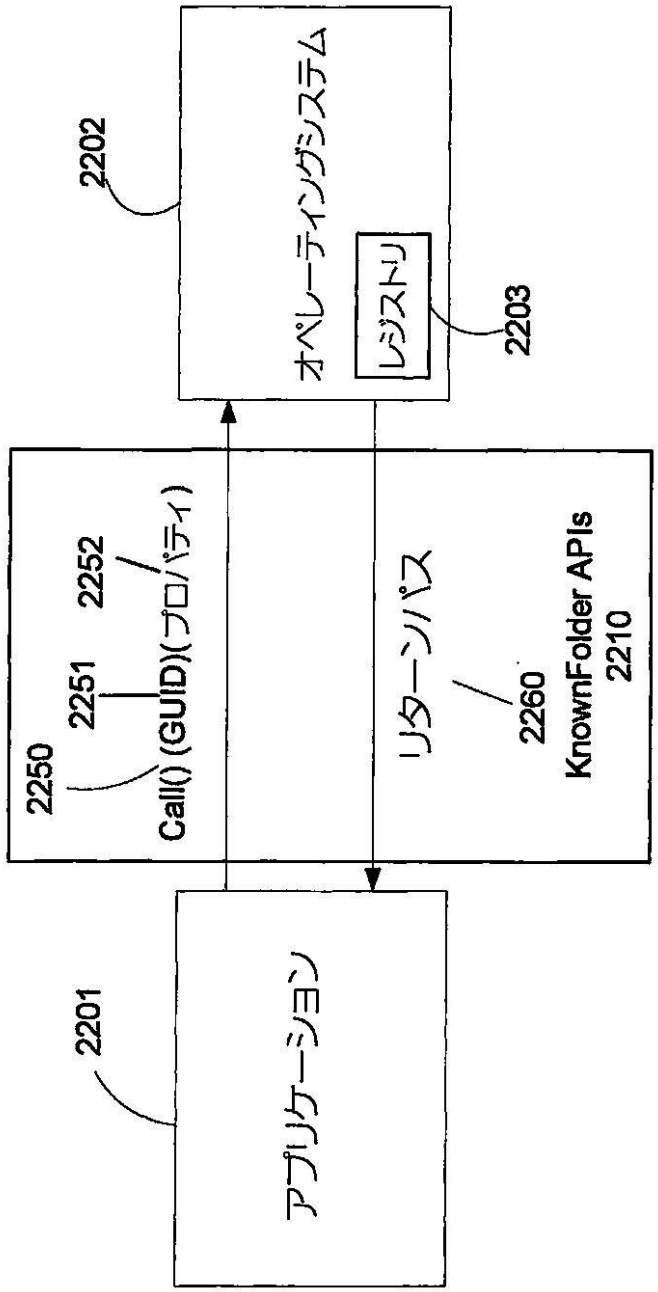
【図１２】


第1の
コードセグメント
インター
フェイス 1
分離
インターフェイス
インター
フェイス 2C
インター
フェイス 2B
インター
フェイス 2A
第2の
コードセグメント
JIT コンパイラ/
インタープリタ
第1の
コードセグメント
インター
フェイス
第2の
コードセグメント

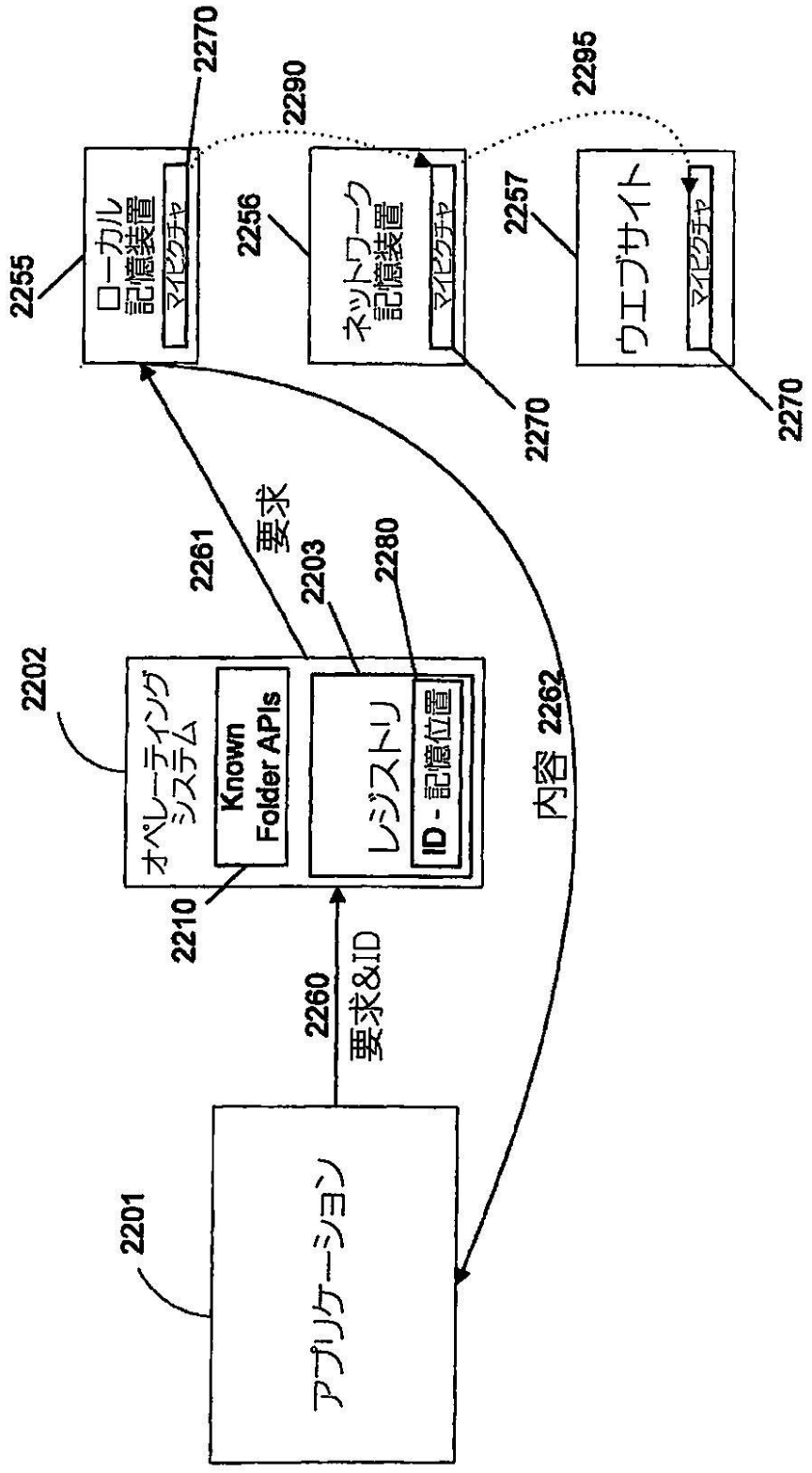
【図１３】


第1の
コンポーネント
I1a
I1b
I1c
M
第2の
コンポーネント
I2a
I2b
I2c
JIT コンパイラ
第1の
コンポーネント
インターフェイス
I1
M
インターフェイス
I2
第2の
コンポーネント

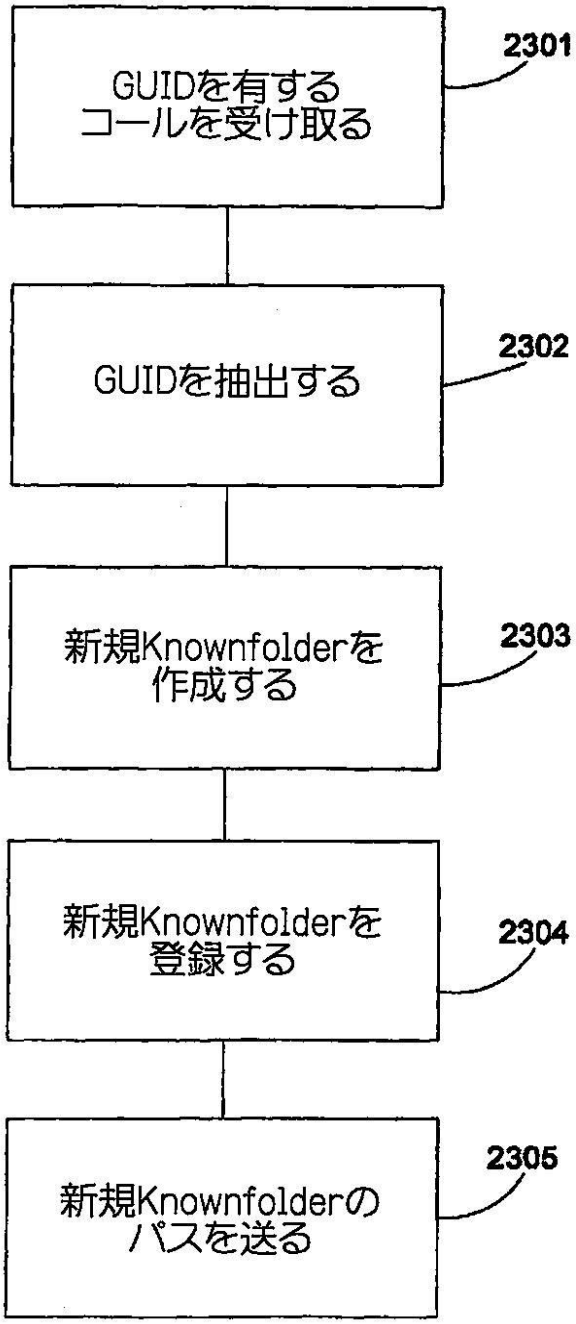
【図２２】


2202
オペレーティングシステム
レジストリ
2203
2250
2251
2252
Call() (GUID)(プロパティ)
リターンパス
2260
KnownFolder APIs
2210
2201
アプリケーション

【図２２ａ】


2270
2290
2295
2255
ローカル記憶装置
マイピクチャ
2256
ネットワーク記憶装置
マイピクチャ
2257
ウエブサイト
マイピクチャ
2270
2270
2261
要求
2203
2280
2202
オペレーティングシステム
Known Folder APIs
レジストリ
ID - 記憶位置
内容 2262
2210
2260
要求&ID
2201
アプリケーション

【図２３】


GUIDを有する
コールを受け取る
2301
GUIDを抽出する
2302
新規Knownfolderを
作成する
2303
新規Knownfolderを
登録する
2304
新規Knownfolderの
パスを送る
2305

フロントページの続き

(72)発明者　ラワット，アンシュル
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
(72)発明者　ウェンツ，ブライアン・ディー
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
(72)発明者　グザック，クリス・ジェイ
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
(72)発明者　デ・ボルチック，ディビッド・ジー
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
(72)発明者　ブレザック，ジョン・イー
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
(72)発明者　チュー，ミング
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
(72)発明者　サムジ，モハメド・エイ
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

審査官　田川　泰宏

(56)参考文献　石川　竜也，プログラミングのスパイス，ソフトバンクパブリッシング株式会社，２００５年３月　４日，初版，p.86-95,267-273

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F　12/00