(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2005-129066
(P2005-129066A)
(43)公開日　平成17年5月19日(2005.5.19)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 1/00	G06F 9/06	660G	5B014
G06F 12/14	G06F 12/14	520A	5B017
G06F 13/00	G06F 12/14	530B	5B076
G06F 13/10	G06F 13/00	530B	
	G06F 13/10	330B	

審査請求 未請求　請求項の数 22　ＯＬ　（全 29 頁）

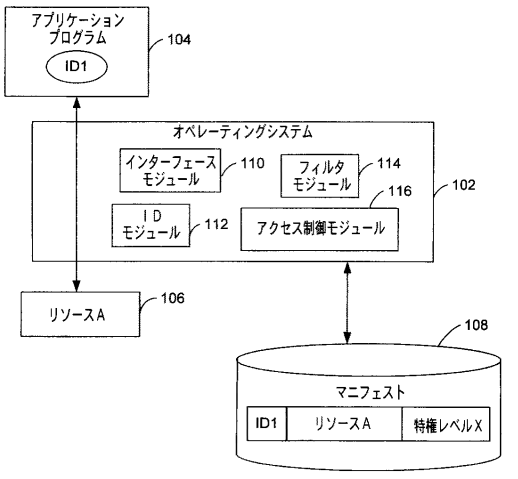
(21)出願番号　特願2004-310057 (P2004-310057)
(22)出願日　平成16年10月25日 (2004.10.25)
(31)優先権主張番号　60/513,941
(32)優先日　平成15年10月24日 (2003.10.24)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　10/868,182
(32)優先日　平成16年6月15日 (2004.6.15)
(33)優先権主張国　米国 (US)

(71)出願人　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　エドワード　ジェイ．プレイティス
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

最終頁に続く

(54)【発明の名称】オペレーティングシステムリソース保護

(57)【要約】
【課題】　インストールされているアプリケーションプログラムが、その上書きされたファイルにアクセスしようと試みた場合にクラッシュする問題を解決する。
【解決手段】　本発明の実施形態は、アプリケーションプログラムまたはグループのアプリケーションプログラムのコンポーネント群に関連付けられた永続する個々のＩＤを使用して、オペレーティングシステムが、コンピューティングシステムにインストール済みの異なるアプリケーションプログラムまたは異なるグループのアプリケーションプログラムを識別し、区別することができるようにする。アプリケーションプログラムの各コンポーネントに関連付けられたＩＤにより、そのアプリケーションプログラムの特定、および削除またはアンインストールが可能になる。また、ＩＤにより、アプリケーションプログラムのリソースの分離、およびオペレーティングシステムリソースの保護が可能になる。
【選択図】　図１



【特許請求の範囲】
【請求項１】
コンピューティングシステム上のリソースへのアクセスをアプリケーションプログラムに許可するコンピュータにより実行する方法であって、
要求の中で識別されたリソースへのアクセスを求める該要求をアプリケーションプログラムから受け取るステップと、
前記アプリケーションプログラムのアプリケーション識別子を判定するステップと、
当該判定されたアプリケーション識別子および前記識別されたリソースの関数としてマニフェストからの特権を識別するステップであって、
前記マニフェストは、前記識別されたリソースにアクセスすることに関して前記アプリケーションプログラムが有する特権を示すステップと、
前記識別された特権に応じて前記識別されたリソースへのアクセスを前記アプリケーションプログラムに許可するステップと
を含むことを特徴とする方法。
【請求項２】
前記マニフェストからの前記特権を識別するステップは、前記アプリケーションプログラムに関連付けられた前記マニフェストからの前記特権を識別するステップを含むことを特徴とする請求項１に記載のコンピュータにより実行する方法。
【請求項３】
前記マニフェストからの前記特権を識別するステップは、前記コンピューティングシステム上で実行されているオペレーティングシステムに関連付けられた前記マニフェストからの前記特権を識別するステップを含むことを特徴とする請求項１に記載のコンピュータにより実行する方法。
【請求項４】
前記アプリケーションプログラムの前記アプリケーション識別子を判定するステップは、前記アプリケーションプログラムを表す複数のファイルおよびシステム設定のそれぞれに関連する前記アプリケーション識別子を判定することを含むことを特徴とする請求項１に記載のコンピュータにより実行する方法。
【請求項５】
前記識別されたリソースへのアクセスを前記アプリケーションプログラムに許可するステップは、前記識別されたリソースへの読み取り専用アクセスを前記アプリケーションプログラムに許可するステップを含むことを特徴とする請求項１に記載のコンピュータにより実行する方法。
【請求項６】
前記識別されたリソースへのアクセスを前記アプリケーションプログラムに許可するステップは、前記識別されたリソースへのアクセスを前記アプリケーションプログラムに対して拒否するステップを含むことを特徴とする請求項１に記載のコンピュータにより実行する方法。
【請求項７】
前記識別されたリソースへのアクセスを前記アプリケーションプログラムに許可するステップは、前記アプリケーションプログラムが使用するために前記リソースのコピーを作成するステップを含むことを特徴とする請求項１に記載のコンピュータにより実行する方法。
【請求項８】
前記アプリケーションプログラムが、前記コンピューティングシステムにインストールされ、前記リソースの前記作成されたコピーを使用して前記アプリケーションプログラムを前記コンピューティングシステムにインストールするステップをさらに含むことを特徴とする請求項７に記載のコンピュータにより実行する方法。
【請求項９】
前記アプリケーションプログラムが、前記コンピューティングシステムにインストール

され、前記コンピューティングシステムにシステム設定を適用することも含め、前記アプリケーションプログラムを前記コンピューティングシステムにインストールするステップをさらに含むことを特徴とする請求項１に記載のコンピュータにより実行する方法。
【請求項１０】
前記要求の中で識別された前記リソースに対するアクセスを求める前記要求を前記アプリケーションプログラムから受け取るステップは、ファイル、ディレクトリ、名前付きオブジェクト、およびシステム設定の１つまたは複数に対するアクセスを求める前記要求を前記アプリケーションプログラムから受け取るステップを含むことを特徴とする請求項１に記載のコンピュータにより実行する方法。
【請求項１１】
前記アプリケーションプログラムのインストール後、前記アプリケーションプログラムの１つまたは複数のアクションに基づいて前記マニフェストを作成するステップをさらに含むことを特徴とする請求項１に記載のコンピュータにより実行する方法。
【請求項１２】
少なくとも１つのファイルが関連しており、コンピューティングシステムにインストール済みの複数のアプリケーションプログラムの１つである特定のアプリケーションプログラム、および関連するシステム設定およびオブジェクトを該コンピューティングシステムからアンインストールする、コンピュータにより実行する方法であって、
前記特定のアプリケーションプログラムをアンインストールする要求を受け取るステップと、
前記特定のアプリケーションプログラムに関連する識別子を判定するステップと、
前記判定された識別子を介して、前記複数のアプリケーションプログラムの前記特定のアプリケーションプログラムだけに関連する前記判定された識別子が関連付けられているファイルを識別するステップと、
前記識別されたファイルを削除するステップと
を含むことを特徴とする方法。
【請求項１３】
前記判定された識別子を介して、前記特定のアプリケーションプログラムをインストールしたことに応答して適用された１つまたは複数のリソースの変更を識別するステップと、
当該識別されたリソースの変更を元に戻すステップと
をさらに含むことを特徴とする請求項１２に記載のコンピュータにより実行する方法。
【請求項１４】
前記適用されたリソースの変更のログを保持するステップをさらに含むことを特徴とする請求項１３に記載のコンピュータにより実行する方法。
【請求項１５】
前記１つまたは複数のリソースの変更を識別するステップは、ファイルタイプの関連付けを識別するステップを含み、前記識別されたリソースの変更を元に戻すステップは、前記ファイルタイプ関連付けを前記ログの中に保持されている以前の関連付けに戻すステップを含むことを特徴とする請求項１４に記載のコンピュータにより実行する方法。
【請求項１６】
前記複数のアプリケーションプログラムの前記特定のアプリケーションプログラムだけに関連する前記ファイルを識別するステップは、前記複数のアプリケーションプログラムの前記特定のアプリケーションプログラムだけに関連する１つまたは複数のオブジェクトを識別するステップを含むことを特徴とする請求項１２に記載のコンピュータにより実行する方法。
【請求項１７】
請求項１から１６のいずれかに記載のコンピュータにより実行する方法を実行するためのコンピュータ実行可能コンポーネント群を有することを特徴とする１つまたは複数のコンピュータ可読媒体。

【請求項１８】
請求項１から１７のいずれかに記載のコンピュータにより実行する方法を実行することを特徴とするシステム。
【請求項１９】
複数のリソースにアクセスするアプリケーションプログラムのアクセス権を規定するマニフェストを表すデータ構造が格納されているコンピュータ可読媒体であって、
前記データ構造は、前記アプリケーションプログラムに対応するＩＤを表す値を格納する第１のフィールドと、
前記アプリケーションプログラムに関連するリソースのリストを格納する第２のフィールドと、
前記第１のフィールドからの前記ＩＤ、および前記第２のフィールドの中に格納されたリソースの前記リストに関連し、リソースの前記リストの中の各リソースにアクセスする前記アプリケーションプログラムのアクセス権を定義する特権を格納する第３のフィールドと
を含むことを特徴とする媒体。
【請求項２０】
前記第１のフィールドは、バージョン、中央処理装置、および公開キーの１つまたは複数に基づく前記値を格納することを特徴とする請求項１９に記載のコンピュータ可読媒体。
【請求項２１】
前記第２のフィールドは、ファイル、ディレクトリ、名前付きオブジェクト、およびシステム設定の１つまたは複数を含むリソースの前記リストを格納することを特徴とする請求項１９に記載のコンピュータ可読媒体。
【請求項２２】
前記第３のフィールドは、インテントの宣言を表す前記特権を格納することを特徴とする請求項１９に記載のコンピュータ可読媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、コンピュータのためのオペレーティングシステムの分野に関する。詳細には、本発明は、オペレーティングシステムによりアプリケーションのインストール、実行、および削除を管理することに関する。
【背景技術】
【０００２】
オペレーティングシステムは、使いやすさおよび信頼性を向上させることにおいて劇的に進歩したが、アプリケーションプログラム群のインストール、管理、および削除（すなわち、アンインストール）に関連するユーザ体験は、依然として改善を必要としている。例えば、アプリケーションプログラムが、インストール中にシステム設定を誤って構成する、または別のアプリケーションプログラムが必要とするファイルを上書きする可能性がある。また、ユーザが、アドウェアやスパイウェアなどの望ましくないアプリケーションをアンインストールすることも困難であり得る。多くのシステムクラッシュおよび性能の低下（例えば、遅い起動時間）も、アプリケーション問題に起因する可能性がある。例えば、次の状況により、アプリケーションプログラム、および、場合により、基礎にあるオペレーティングシステムに障害が生じる可能性がある。すなわち、アプリケーションの不完全なアンインストール、アプリケーションプログラムをアンインストールする際の過度な削除（ｏｖｅｒ　ｄｅｌｅｔｉｏｎ）、および不適切に格納されたファイルである。
【発明の開示】
【発明が解決しようとする課題】
【０００３】
したがって、アプリケーション影響を管理するための改良されたシステムおよび方法が

、以上の欠点およびその他の欠点の１つまたは複数に対処するのに所望される。

【０００４】

一部の現在のオペレーティングシステムでは、新たにインストールされたアプリケーションプログラムが、その新たにインストールされたアプリケーションプログラムによって必要とされるより古いバージョンまたはより新しいバージョンで、共有ダイナミックリンクライブラリ（ＤＬＬ）を上書きする可能性がある。より古いファイルおよびより新しいファイルが上書きされたファイルと不適合である場合、上書きされたファイルに依存する現在、インストールされているアプリケーションプログラムが、その上書きされたファイルにアクセスしようと試みた場合にクラッシュする可能性がある。

【課題を解決するための手段】

【０００５】

本発明の実施形態には、オペレーティングシステムが自らのリソースを保護することを可能にするための方法が含まれる。一実施形態では、本発明は、アプリケーションプログラムまたはアプリケーションプログラムのグループに関連する永続する個別のＩＤを使用して、オペレーティングシステムが、異なるアプリケーションプログラム、またはアプリケーションプログラムのグループおよびアプリケーションプログラムのコンポーネントを識別し、区別することができるようにすることを含む。

【０００６】

オペレーティングシステムまたはその他のプログラムは、アプリケーションプログラムのそれぞれに関連するＩＤを介してアプリケーションプログラム群を操作する。例えば、オペレーティングシステムは、ＩＤを使用して（１）クリーンなアンインストールを確実にし、（２）アプリケーションが、許可を有していないサービスにアクセスする、または許可を有していないアクションを実行することを防止し、（３）アプリケーションを互いによりよく分離するようにシステムリソースを仮想化し、（４）アプリケーションインパクトロールバックを可能にし（例えば、ファイルタイプの関連付けをアプリケーションインストール前の状態に戻し）、（５）ファイルおよびレジストリの所有権の追跡を可能にする。保護メカニズムには、読み取り専用アクセスを提供すること、変更をログ記録してロールバックを可能にすること、アプリケーション別、およびユーザ別にリソースを仮想化することが含まれるが、以上には限定されない。例えば、オペレーティングシステムは、書き込み保護されたファイルへの書き込みアクセスを要求したアプリケーションプログラム用に、書き込み保護されたファイルのコピーを作成する。

【０００７】

本発明の一態様によれば、方法は、コンピューティングシステム上のリソースへのアクセスをアプリケーションプログラムに許可する。方法は、要求の中で識別されたリソースへのアクセスを求める要求をアプリケーションプログラムから受け取ることを含む。また、方法は、アプリケーションプログラムのアプリケーション識別子を判定することも含む。方法は、判定されたアプリケーション識別子および識別されたリソースの関数（ｆｕｎｃｔｉｏｎ）としてマニフェストからの特権を識別することを含む。マニフェストは、識別されたリソースにアクセスすることに関してアプリケーションプログラムが有する特権を示す。また、方法は、識別された特権に応じて識別されたリソースへのアクセスをアプリケーションプログラムに許可することも含む。

【０００８】

本発明の別の態様によれば、１つまたは複数のコンピュータ可読媒体が、リソースへのアクセスをアプリケーションプログラムに許可するためのコンピュータ実行可能コンポーネント群を有する。コンポーネント群には、要求の中で識別されたリソースへのアクセスを求める要求をアプリケーションプログラムから受け取るインターフェースモジュールが含まれる。コンポーネント群には、アプリケーションプログラムが、アプリケーションプログラム、およびアプリケーションプログラムのコンポーネントを他のアプリケーションプログラム群から区別するためのアプリケーション識別子を判定するＩＤモジュールも含まれる。コンポーネント群には、ＩＤモジュールおよび識別されたリソースによって判定

されたアプリケーション識別子の関数として、マニフェストからの特権を識別するフィルタモジュールも含まれる。マニフェストは、識別されたリソースにアクセスすることに関してアプリケーションプログラムが有する特権を示す。コンポーネント群には、フィルタモジュールによって識別された特権に応じて、識別されたリソースへのアクセスをアプリケーションプログラムに許可するアクセス制御モジュールも含まれる。

【０００９】

本発明のさらに別の態様によれば、コンピュータ可読媒体が、複数のリソースにアクセスするアプリケーションプログラムのアクセス権を規定するマニフェストを表すデータ構造を格納する。データ構造は、アプリケーションプログラムに対応するＩＤを表す値を格納する第１のフィールドを含む。データ構造は、アプリケーションプログラムに関連するリソースのリストを格納する第２のフィールドも含む。データ構造は、第１のフィールドからのＩＤ、および第２のフィールドの中に格納されたリソースのリストに関連する特権を格納する第３のフィールドをさらに含む。特権は、リソースのリストの中の各リソースにアクセスするアプリケーションプログラムのアクセス権を定義する。

【００１０】

本発明のさらに別の態様によれば、システムが、システムリソースへのアクセスをアプリケーションに許可する。システムは、マニフェストを格納するメモリ領域を含む。マニフェストは、アプリケーション識別子およびリソースを特権にマップする。アプリケーション識別子は、アプリケーションプログラムに関連付けられている。システムは、コンピュータ実行可能命令を実行して、リソースを求めるアプリケーションプログラムからの要求に応答して、アプリケーション識別子およびリソースの関数としてメモリ領域の中に格納されているマニフェストからの特権を識別するように構成されたプロセッサも含む。プロセッサは、コンピュータ実行可能命令を実行して、識別された特権に応じてリソースへのアクセスをアプリケーションプログラムに許可するようにさらに構成される。

【００１１】

本発明の別の態様によれば、方法は、コンピューティングシステムから特定のアプリケーションプログラムをアンインストールする。この特定のアプリケーションプログラムには、少なくとも１つのファイルが関連付けられている。この特定のアプリケーションプログラムは、コンピューティングシステムにインストールされている複数のアプリケーションプログラムの１つである。方法は、特定のアプリケーションプログラムをアンインストールする要求を受け取ることを含む。方法は、特定のアプリケーションプログラムに関連する識別子を判定することも含む。方法は、判定された識別子を介して、複数のアプリケーションプログラムの特定のアプリケーションプログラムにだけ関連するファイルを識別することをさらに含む。特定されたファイルには、判定された識別子が関連付けられている。方法は、識別されたファイルを削除することも含む。

【００１２】

代替として、本発明は、他の様々な方法および装置を含むことも可能である。

【００１３】

その他の特徴は、一部は明白であり、一部は以下に指摘する。

【発明を実施するための最良の形態】

【００１４】

対応する符合は、すべての図面で対応する部分を示している。

【００１５】

一実施形態では、本発明は、リソースを保護するための方法を提供する。詳細には、オペレーティングシステムの機能が、ファイルおよびシステム設定の保護の宣言を可能にする。宣言された保護は、アプリケーションライフサイクル中にオペレーティングシステムが使用して、アプリケーションプログラムのインストール、実行、修理、および削除を管理し、追跡し、予測し、軽減することが可能なアクションのセットすべてにわたって、オペレーティングシステムまたは他のアプリケーションプログラムによって永続させられ、実施される。リソース保護は、極めて重要なシステムデータ（例えば、ファイルの関連付

け）の参照の完全性を提供し、各アプリケーションプログラムによるリソースへのアクセスを追跡し、分離することによってアプリケーション脆弱性問題に対処して信頼性および整合性を向上させ、システムおよびアプリケーションによる保護されたリソースとの対話の影響を管理する。例えば、本発明の実施形態を使用して、ウイルスまたはワームに感染しているアプリケーションに対するセキュリティを提供することができる。本発明の実施形態は、任意のオペレーティングシステムモデルで機能して、拡張性を提供し、統合を可能にする。本発明の実施形態のリソース保護戦略および実施により、アプリケーションインストーラが、極めて重要なシステムリソースを偶然に、または悪意で変更するまたは置き換えることも防止される。本発明の実施形態を、システムリソースを保護するための他の戦略と組み合わせることもできる。例えば、コンピューティングシステムは、ロックダウン（ｌｏｃｋ　ｄｏｗｎ）、分離、仮想化、トランザクション、およびサンドボックス化（ｓａｎｄｂｏｘｉｎｇ）の組合せを含む戦略を実施することができる。

【００１６】

図１を参照すると、オペレーティングシステム１０２の典型的な実施形態が、マニフェスト１０８に従ったリソースへのアクセスをアプリケーションプログラム１０４に提供する。リソースには、ファイル、フォルダ、プロセス、スレッド、システム設定、名前付きオブジェクト、アプリケーションプログラミングインターフェース（ＡＰＩ）、特定のコードパス、実行可能ルーチンのライブラリ、オペレーティングシステムプロパティ値、およびオペレーティングシステムリソースが含まれるが、以上には限定されない。名前付きオブジェクトには、英字データ、数字データ、英数字データ、または非人間可読（ｎｏｎ－ｈｕｍａｎ　ｒｅａｄａｂｌｅ）（例えば、グローバル一意識別子）データで識別されたあらゆるオブジェクトが含まれる。例えば、ユーザＩＤで保護することができるあらゆるオブジェクトを、アプリケーションＩＤ（例えば、ネットワークソケット）で保護することができる。例えば、いくつかのＡＰＩおよびコードパスが、メール送信能力を提供し、それらのＡＰＩおよびコードパスへのアクセスを制限することができる。別の例では、システムを再起動する能力が制限される。リソースには、識別の名前付きオブジェクトだけでなく、システムの名前空間（例えば、「名前」自体）も含まれる。例えば、オブジェクトが作成される前に名前を確保すること、または「不法占拠すること（ｓｑｕａｔｔｉｎｇ）」により、脆弱性とセキュリティ問題がともにもたらされる。

【００１７】

一実施形態では、アプリケーションプログラム（例えば、アプリケーションプログラム１０４）によって提示されたマニフェスト１０８のようなマニフェストにより、アプリケーションプログラムが有することを所望する特権が示される。動作の際、オペレーティングシステムは、要求された特権のいくつかを許可する、または拒否することができ、オペレーティングシステムがアプリケーションプログラムに関して保持する計算されたマニフェスト、または有効なマニフェストがもたらされる。

【００１８】

アプリケーションプログラム１０４のような各アプリケーションプログラムに、アプリケーション識別子が割り当てられて、アプリケーションプログラムが他のアプリケーションプログラム群から区別される。一実施形態では、アプリケーション識別子は、アプリケーションプログラムのグループに割り当てられて、アプリケーションプログラムのグループ内の各アプリケーションプログラムが、リソースに対して、そのグループ内の他のアプリケーションプログラム群と同一のアクセスまたは特権を有することを可能にする。図１では、アプリケーションプログラム１０４に、識別子ＩＤ１が関連付けられている。オペレーティングシステム１０２は、リソースＡ１０６のようなリソースにアクセスしようとするアプリケーションプログラム１０４による試みをインターセプト（ｉｎｔｅｒｃｅｐｔ）する。オペレーティングシステム１０２は、マニフェスト１０８を調べて、リソースＡ１０６に関してアプリケーションプログラム１０４に許された特権またはアクセスを判定する。この例では、メモリ領域が、マニフェスト１０８を格納している。マニフェスト１０８は、アプリケーション識別子およびリソースを特権にマップする。マニフェスト１

０８は、アプリケーション識別子（例えば、ＩＤ１）およびリソース（例えば、リソースＡ１０６）の関数として特権（例えば、特権Ｘ）を格納している。特権Ｘは、例えば、読み取り専用アクセスに対応することが可能である。オペレーティングシステム１０２は、識別された特権に応じて、リソースＡ１０６へのアクセスをアプリケーション識別子に提供する。

【００１９】

一実施形態では、オペレーティングシステム１０２は、コンピュータ可読媒体上に１つまたは複数のコンピュータ実行可能コンポーネントを格納しているか、またはコンピュータ可読媒体上の１つまたは複数のコンピュータ実行可能コンポーネントにアクセスできる。オペレーティングシステム１０２に関連するプロセッサが、そのコンピュータ実行可能コンポーネント、または他のコンピュータ実行可能命令を実行して、リソースを求めるアプリケーションプログラム１０４からの要求に応答して、アプリケーション識別子およびリソースの関数としてメモリ領域の中に格納されているマニフェスト１０８からの特権を識別するように構成される。プロセッサは、コンピュータ実行可能命令を実行して、識別された特権に応じて、リソース、またはリソースのコピーへのアクセスをアプリケーションプログラム１０４に許可するようにさらに構成される。

【００２０】

詳細には、コンピュータ実行可能コンポーネント群は、リソースへのアクセスをアプリケーションプログラム１０４に許可する。図１の特定の実施形態では、コンポーネント群には、インターフェースモジュール１１０、ＩＤモジュール１１２、フィルタモジュール１１４、およびアクセス制御モジュール１１６が含まれる。図１のモジュール群は、オペレーティングシステム１０２とは別個に、独立して存在することが可能である。さらに、本発明の実施形態の機能および構造は、任意の数のモジュール、コンポーネントなどに編成することができる。例えば、モジュール群を分散させることができる。

【００２１】

インターフェースモジュール１１０は、要求の中で識別されたリソースへのアクセスを求める要求をアプリケーションプログラム１０４から受け取る。一実施形態では、インターフェースモジュール１１０は、次の１つまたは複数へのアクセスを求める要求をアプリケーションプログラム１０４から受け取る。すなわち、ファイル、ディレクトリ、およびシステム設定（例えば、レジストリエントリ）である。ＩＤモジュール１１２は、アプリケーションプログラム１０４のアプリケーション識別子を判定して、アプリケーションプログラム１０４、およびアプリケーションプログラム１０４のコンポーネントを他のアプリケーションプログラム群から区別する。一実施形態では、ＩＤモジュール１１２は、アプリケーションプログラムのグループのアプリケーション識別子（例えば、分離識別子）を判定する。アプリケーションプログラム１０４は、複数のファイルおよびシステム設定を含むことが可能であるため、ＩＤモジュール１１２は、アプリケーションプログラム１０４を表す複数のファイルおよびシステム設定のそれぞれに関連するアプリケーション識別子を判定する。フィルタモジュール１１４は、ＩＤモジュール１１２によって判定されたアプリケーション識別子、および識別されたリソースの関数として、マニフェスト１０８からの特権を識別する。マニフェスト１０８は、識別されたリソースにアクセスすることに関してアプリケーションプログラム１０４が有する特権を示す。アクセス制御モジュール１１６は、フィルタモジュール１１４によって識別された特権に応じて、識別されたリソースへのアクセスをアプリケーションプログラム１０４に許可する。一実施形態では、構成モジュールが、アプリケーションプログラム１０４に関連するインストール媒体からアプリケーションマニフェストを受け取る。アプリケーションマニフェストは、アプリケーションプログラム１０４に関連するファイルおよびリソースの変更（例えば、システム設定）のリストを表す。構成モジュールは、アプリケーションマニフェストの中に含まれるデータでオペレーティングシステムマニフェストを更新することができる。代替として、構成モジュールは、インストールされた各アプリケーションに関する各アプリケーションマニフェストを保持してもよい。

【００２２】
マニフェスト
マニフェスト１０８は、アプリケーションプログラム１０４、またはオペレーティングシステム１０２のようなオペレーティングシステムに関連するアイテム（例えば、ファイルおよびリソースの変更）またはオブジェクトのリストを含む。代替として、アプリケーションプログラム１０４に関連するアイテムのリストは、各リソースプロバイダに関する構成ファイルまたはストアの中に格納してもよい。別の実施形態では、オブジェクトの作成者が、オブジェクト上で直接にアクセス特権を規定する。
【００２３】
マニフェスト１０８は、アプリケーションプログラム１０４またはオペレーティングシステム１０２に関連するリソースに関する特権のリストも含むことが可能である。例えば、アプリケーションプログラム１０４の作成者が、マニフェストの中で、オペレーティングシステム１０２のリソースに対する特権、および／またはアプリケーションプログラム１０４が作成することができるリソースに対する特権を規定することができる。代替として、マニフェストは、アプリケーションプログラム１０４に関連するＩＤ情報を単に格納してもよい。別の例では、インストールされるべきアプリケーションプログラム１０４を格納しているインストール媒体が、アプリケーションプログラム１０４に関連するアイテム、およびアプリケーション専用リソースに関連する特権をリストするアプリケーションマニフェストも格納していることが可能である。アプリケーションプログラム１０４の展開を担当するサードパーティアプリケーションベンダまたは担当者が、アプリケーションマニフェストを作成してもよい。別の例では、オペレーティングシステムマニフェストが、オペレーティングシステム１０２にインストールされたアプリケーションプログラム群に関連するアイテムのリストを格納する。オペレーティングシステムマニフェストは、オペレーティングシステム１０２に関連するコンポーネントのリストをさらに格納することが可能である。一実施形態では、オペレーティングシステムマニフェストは、オペレーティングシステムコンポーネントまたはインストール済みのアプリケーションプログラムのそれぞれに関する保護動作の集約を表す。集約されたマニフェストは、各ファイル、各ディレクトリ、および各システム設定に関して許される対話のタイプを定義する。
【００２４】
オペレーティングシステム１０２は、自らがどのように保護されることを望むか、オペレーティングシステムコンポーネント群とその他のコンポーネント群がどのように対話し、システムを拡張することができるかを規定するという点で、自己記述型である。本発明の一実施形態では、オペレーティングシステム１０２の一部であるすべてのアイテムまたはリソース（例えば、ファイル、ディレクトリ、レジストリキーおよびレジストリ値、ドライバなど）に関して、オペレーティングシステム１０２により実行するべき保護動作のタイプを宣言することが可能である。
【００２５】
マニフェスト１０８は、コンピュータ可読媒体上のデータ構造として格納される。マニフェスト１０８は、複数のリソースにアクセスするアプリケーションプログラム１０４のようなアプリケーションプログラムのアクセス権を規定する。図１の典型的なデータ構造は、アプリケーションプログラム１０４に対応するＩＤを表す値（例えば、ＩＤ１）を格納する第１のフィールドを含む。例えば、第１のフィールドは、次の１つまたは複数に基づく値を格納することができる。すなわち、バージョン、中央処理装置、および公開キーである。データ構造は、アプリケーションプログラム１０４に関連するリソース（例えば、リソースＡ１０６）のリストを格納する第２のフィールドも含む。例えば、第２のフィールドは、次のようなリソースのリストを格納することができる。すなわち、ファイル、ディレクトリ、およびシステム設定である。データ構造は、第１のフィールドからのＩＤ、および第２のフィールドの中に格納されたリソースのリストに関連する特権（例えば、特権Ｘ）、またはインテント（ｉｎｔｅｎｔ）の他の宣言を格納する第３のフィールドも含む。特権は、リソースのリストの中の各リソースにアクセスするアプリケーションプロ

グラム１０４のアクセス権を定義する。

【００２６】

アプリケーションの作成者は、信頼情報セクションを使用してマニフェスト１０８のようなマニフェストを作成することができる。アプリケーション作成者は、厳密な名前をアプリケーションに割り当て、アプリケーションのマニフェストに署名する（デジタル署名またはデジタル証明書を使用して）こともできる。アプリケーションがインストールされる際、１つまたは複数の証明書ストアを検査して、アプリケーションマニフェストの証明書および署名を検証するようにオペレーティングシステム１０２を構成することができる。一実施形態では、署名済みのドライバパッケージだけがインストールされる。例えば、企業が、独自の証明書ストアを有することが可能である。同様に、特定のシステムが、証明書ストアを有することも可能であり、その証明書ストアに照らしてアプリケーションマニフェストを検証することができる。検証が済むと、マニフェストデータ、およびあらかじめ構成された既定のポリシーに基づいて信頼アクションを管理するようにオペレーティングシステム１０２を構成することができる。

【００２７】

マニフェスト１０８のようなマニフェストは、いくつかの仕方で署名することができる。例えば、マニフェストは、証明書がストアの中に検証のために保持されて、Ａｕｔｈｅｎｔｉｃｏｄｅプロセスを使用して署名することができる。ドメイン管理者が、自身の特定の企業またはドメインのためにマニフェストに署名することもできる。例えば、展開マニフェストを使用して、いずれのアプリケーションが特定のインストールのために署名されるかを規定することができる。ローカル管理者が、マニフェストに署名することもできる。それぞれの個別のマシンが、署名キーを有するように構成することもできる。

【００２８】

一実施形態では、マニフェスト１０８は、脆弱な名前（ｗｅａｋ　ｎａｍｅ）と厳密な名前（ｓｔｒｏｎｇ　ｎａｍｅ）をともに含むことが可能である。脆弱な名前は、従来のアプリケーションファイル名に相当することが可能であり、厳密な名前は、ファイル名、バージョン番号、カルチャ（ｃｕｌｔｕｒｅ）、および公開キーに相当することが可能である。別の実施形態では、厳密な名前は、モジュールが署名した（ｍｏｄｕｌｅ　ｓｉｇｎｅｄ）秘密キーのハッシュであることが可能である。さらに別の実施形態では、厳密な名前は、公開キートークンであることが可能である。

【００２９】

例えば、以下のＸＭＬが、マニフェスト１０８に関する１つの厳密な名前を表すことが可能である。

【００３０】

【表１】

```
<assemblyIdentity
        version="1.0.0.0"
        processorArchitecture="x86"
        name="SampleApp"
        publicKeyToken="0123456789abcdef"
        type="typeA"/>
```

【００３１】

以下は、マニフェスト１０８の一実施形態の信頼情報セクションの見本である。

【００３２】

【表２】

```
<trustInfo>
  <security>
    <requestedPrivileges>
      <requestedExecutionLevel
        leastPrivileged="true"
        adminPrivileged="true"
        requireDefaultDesktop="false"/>
    </requestedPrivileges>
  </security>
</trustInfo>
```

【００３３】

マニフェスト１０８を有さないアプリケーションの場合、オペレーティングシステム１０２が、事前定義された既定に従って要求された特権を有するマニフェスト１０８を作成するように構成されることが可能である。例えば、マニフェスト１０８は、ユーザアクセスの最小限の特権レベルを要求するように構成することができる。

【００３４】

代替として、またはさらに、オペレーティングシステムは、アプリケーションのアクションを観察し、アプリケーションが実際に使用する特権だけを提供するようにマニフェストをカスタマイズすることもできる。アプリケーションの数回の実行後、マニフェストはロックされ、マニフェストによって許可される特権を拡張するのに、明示的なユーザ入力または管理者ポリシーが要求される。一部の実施形態では、脆弱なアプリケーションが、インストール直後に侵害される可能性は、後にアプリケーションが侵害される可能性と比べて比較的低い。アプリケーションが、マニフェストがロックされた後に侵害された場合、侵害されたアプリケーションの動作は、マニフェストによって許される動作に制限され、その動作は、アプリケーションの侵害を受けていない動作によって決まっている。

【００３５】

次に、アプリケーションＩＤを使用してリソースを保護するための方法を説明する。

【００３６】

アクセス制御を提供すること

次に、図２を参照すると、典型的な流れ図が、アクセス制御方法の動作を示している。一実施形態では、本発明は、コンピューティングシステム上のリソースへのアクセスをアプリケーションプログラムに与える。方法は、２０２で、要求の中で識別されたリソースへのアクセスを求める要求をアプリケーションプログラムから受け取ること、２０４で、アプリケーションプログラムのアプリケーション識別子を判定すること、２０６で、判定されたアプリケーション識別子、および識別されたリソースの関数としてマニフェスト（例えば、オペレーティングシステムマニフェストおよび／またはアプリケーションプログラムマニフェスト）からの特権を識別すること、および２０８で、識別された特権に応じて識別されたリソースへのアクセスをアプリケーションプログラムに許可することを含む。一実施形態では、２０４でアプリケーションプログラムのアプリケーション識別子を判定することは、すべてのファイル、フォルダ、システム設定変更（例えば、レジストリキーおよびレジストリ値）またはリソースに、固有で、整合性があり、永続的な、繰り返し可能な識別子でタグ付けすることを含む。

【００３７】

一実施形態では、オペレーティングシステムは、図２に示した方法を実行する。別の実

施形態では、オペレーティングシステムとは別個のアプリケーションプログラムまたはアプリケーションサービスが、図２に示した方法を実行する。１つまたは複数のコンピュータ可読媒体が、図２に示した方法を実行するためのコンピュータ実行可能命令を有する。

【００３８】

次に、様々な典型的な特権、または他の形態のアクセスを、リソースを保護するための軽減アーキテクチャの見本を参照して説明する。

【００３９】

典型的な軽減アーキテクチャ

次に、図３を参照すると、典型的な流れ図が、様々なリソースを保護するための軽減アーキテクチャを示している。一実施形態では、図３に示した方法は、リソースにアクセスしようと試みているアプリケーションプログラムのアプリケーション識別子に基づき、マニフェストの中で記述されているリソース特権を実施する。一部の特権を本明細書で説明するが、本明細書で説明しない様々な特権、特権のレベル、またはアクセスも本発明の範囲に含まれる。同様に、一部のリソースを本明細書で説明するが、本明細書で説明しない様々なリソースも本発明の範囲に含まれる。

【００４０】

図３で、識別子ＩＤ１を有するアプリケーションプログラムのようなアプリケーションプログラムが、様々なリソースへのアクセスを要求する。本発明の実施形態は、その要求を受け取り、リソースに関してそのアプリケーションプログラムに識別された特権またはアクセスに応じて要求を処理する。図３の例では、アプリケーションプログラムは、読み取り専用特権で一部のオペレーティングシステムリソース（例えば、ファイルおよび設定）に対するアクセスを有する。アプリケーションプログラムは、それらのリソースの１つへの読み取りアクセスを求める要求を送り、本発明の実施形態が、その１つのリソースへの読み取りアクセスをアプリケーションプログラムに許可する。アプリケーションプログラムが、３０２で、それらの読み取り専用リソースの１つを変更する要求を送った場合、本発明の実施形態は、その１つのリソースへのアクセスをアプリケーションプログラムに対して拒否する。要求は、３０４で、暗黙に（例えば、アプリケーションプログラムに応答がまったく戻されない）失敗するか、または明示的に（例えば、アプリケーションプログラムに否定応答が戻される）失敗する。アプリケーションプログラムは、読み取り－書き込み特権でのアプリケーション専用リソース（例えば、アプリケーションプログラムに関連するファイルおよび設定）へのアクセスも有する。それらのリソースは、アプリケーションプログラムに関連しているため、それらのリソースの変更が、オペレーティングシステム脆弱性の問題を生じさせることは一般的にない。オペレーティングシステムは、それらのリソースのセマンティクス（ｓｅｍａｎｔｉｃｓ）の知識をほとんど有さず、ほとんど関心がない。

【００４１】

アプリケーションプログラムは、保護された特権で、他のオペレーティングシステムリソースに対するアクセスを有する。アプリケーションプログラムが、３０６で、それらの保護されたオペレーティングシステムリソース（例えば、設定またはファイル）の１つを変更する要求を送った場合、本発明の実施形態は、３０８で、アプリケーションプログラム向けに保護されたリソースの仮想ビューを戻す。詳細には、保護された特権に関して、本発明の実施形態は、アプリケーションプログラムによる読み取り－書き込みアクセスのために、要求されたリソースのコピーを、そのコピーが既存でない場合、作成する。コピーがまだ存在していない一実施形態では、アプリケーションプログラムからの要求が読み取りアクセスだけを求めている場合、コピーは作成されない。リソースのそのコピーは、そのアプリケーションプログラムによる、または同一のアプリケーション識別子を有するアプリケーションプログラムのグループによる使用に専用である。アプリケーション識別子により、本発明の実施形態は、異なるアプリケーション識別子を有するアプリケーションプログラムに、１つまたは複数のリソースの独自の仮想ビューまたはコピーを提供することができるようになる。例えば、オペレーティングシステムが、システム設定の、オペ

レーティングシステムに独自のコピーを所有する一方で、そのシステム設定に値を書き込むアプリケーションプログラムは、システム設定の、アプリケーションプログラムに独自のコピーを受け取る。一部の典型的な実施形態では、異なるアプリケーションが、システム設定（例えば、レジストリエントリ）の異なる仮想ビューを受け取ることが可能である。所望される（例えば、ユーザにより）システム保護のタイプに応じて、ユーザ別および／またはアプリケーションプログラム別にリソースを仮想化することができる。判定のアプリケーション識別子を有するアプリケーションプログラムによる仮想化されたリソースに対する変更は、他のアプリケーション識別子を有するアプリケーションプログラムにまったく影響を与えない（例えば、見えない）。個々のアプリケーション、またはアプリケーションの個々のグループに選択されたシステムリソースの独自のビューを提供することにより、オペレーティングシステムは、１つのアプリケーションプログラムが、他のアプリケーションプログラム群によって必要とされているリソースに上書きする、または別の形で破壊するのを防止することができる。

【００４２】

一実施形態では、アプリケーションプログラムは、アプリケーションプログラムのコンピューティングシステムへのインストール中に、リソースの仮想化されたコピーを使用する。例えば、アプリケーションプログラムは、システム設定を格納しているファイルの作成済みのコピーを使用して、コンピューティングシステムにシステム設定を適用することができる。

【００４３】

アプリケーションプログラムは、アプリケーション専用リソースに対するアクセスを有する。アプリケーション専用リソースには、アプリケーションプログラムに特有のリソースが含まれる。オペレーティングシステムおよびその他のアプリケーションプログラム群は、アプリケーション専用リソースによって一般に影響されない。アプリケーションプログラムが、３１０で、アプリケーション専用リソースを変更する要求を送った場合、本発明の実施形態は、３１２で、その要求を許可し、処理する。

【００４４】

アプリケーションプログラムは、３１４で、システム拡張性（例えば、オペレーティングシステムに機能を追加する）を変更する要求を送ることができる。一実施形態では、本発明の実施形態は、３１２で、要求された変更を許可する。

【００４５】

システム拡張性およびアプリケーション専用リソース（例えば、ファイルおよびシステム設定）に対する変更は、３１８で、ログ記録されるか、または別の形で記録される。一般に、システム拡張性の変更は、オペレーティングシステムに対する変更なしに、オペレーティングシステムに追加の機能を与える。システム拡張性の変更、およびアプリケーション専用リソースの変更を記録することにより、その変更のロールバック、ならびにその変更に関連するアプリケーションプログラムの完全な削除またはアンインストールが可能になる。

【００４６】

典型的な軽減戦略

次に、図４を参照すると、典型的な流れ図が、ファイル、システム設定、および拡張機能に関するアクセス制御を提供する方法の動作を示している。図４の例では、オペレーティングシステムが、この方法を実施する。しかし、オペレーティングシステムに関連していないアプリケーションプログラムまたはアプリケーションサービスが、この方法を実施することもできる。図４では、ＣｒｅａｔｅＰｒｏｃｅｓｓ（）のような関数を介してアプリケーションプログラム（例えば、ｘｘｘｘ．ｅｘｅ）を実行するプロセスが作成される。オペレーティングシステムは、４０２で、アプリケーションプログラムに関連するアプリケーション識別子が存在するかどうかを判定する。存在しない場合、オペレーティングシステムは、４０４で、アプリケーション識別子を決め、その情報を保存する（例えば、決められたアプリケーション識別子をマニフェストの中に格納する）。アプリケーショ

ンプログラムが、４０６で実行され、操作を実行する。オペレーティングシステムが、その操作を分析する。例えば、一実施形態では、特殊な特権で実行される許可された信頼されるインストールプロセスだけが、保護された領域内で追加、変更、または削除を行うことができる。アプリケーションプログラム群は、保護された領域内でデータを作成する、または変更することを阻止される。

【００４７】

図４の実施形態では、操作が、４０８で、ファイル操作であった場合、オペレーティングシステムは、４１０で、そのファイル操作が、ファイルに影響を与える（例えば、ファイル操作が、ファイルを変更する）かどうかを判定する。ファイル操作がファイルに影響を与えない場合、オペレーティングシステムは、４１４で、ファイルシステムに対してそのファイル操作が実行されることを許す。ファイル操作がファイルに影響を与える場合、オペレーティングシステムは、４１２で、図３に示したような軽減戦略に従って軽減されたファイル操作を実行する。ファイルシステムに対する変更は、行われる場合、４１５で、ログに記録される。

【００４８】

操作が、４１６で、システム設定操作である場合、オペレーティングシステムは、４１８で、そのシステム設定操作がシステム設定に影響を与える（例えば、システム設定操作が、システム設定を変更する）かどうかを判定する。システム設定操作が、システム設定に影響を与えない場合、オペレーティングシステムは、４２２で、システム設定に対してそのシステム設定操作が実行されることを許す。システム設定操作がシステム設定に影響を与える場合、オペレーティングシステムは、４２０で、図３に示したような軽減戦略に従って、軽減されたシステム設定操作を実行する。システム設定に対する変更は、行われる場合、４１５で、ログに記録される。

【００４９】

操作が、４２４で、オペレーティングシステムに拡張機能を読み込む要求を表す場合、オペレーティングシステムは、４２６で、アプリケーションプログラム（例えば、ｘｘｘｘ．ｅｘｅ）が、保護（例えば、「元に戻す」を可能にする）を所望するかどうかを判定する。例えば、アプリケーションプログラムは、保護の要望をオペレーティングシステムに明示的に知らせることができる。アプリケーションプログラムが保護を望まない場合、オペレーティングシステムは、４２８で、拡張機能が読み込まれることを許す。アプリケーションプログラムが、保護が所望されることを示した場合、オペレーティングシステムは、４３０で、拡張機能が異種の（ｆｏｒｅｉｇｎ）拡張機能（例えば、第三者によって供給された）であるかどうかを判定する。拡張機能が異種ではない場合、オペレーティングシステムは、４２８で、拡張機能が読み込まれることを許す。拡張機能が異種である場合、オペレーティングシステムは、４３２で、図３に示したような軽減戦略に従って、軽減された拡張機能の読み込みを実行する。拡張機能の読み込みは、ログに記録することができる。例えば、記録は、オペレーティングシステムを実行しているコンピューティングシステムのユーザが構成できることが可能である。

【００５０】

仮想化を使用すると、アプリケーションが、独自のローカル名前空間内でオブジェクトを作成し、変更する一方で、オペレーティングシステムは、グローバル名前空間内でオブジェクトを作成し、変更する。１つのグローバル名前空間が存在し、複数のローカル名前空間が存在する可能性がある。作成操作に関して、アプリケーションは、アプリケーションのローカル名前空間内でオブジェクトを作成する。アプリケーションがオブジェクトを変更しようと試みた場合、オペレーティングシステムは、そのオブジェクトがアプリケーションのローカル名前空間内に存在するかどうかを調べる。ローカルオブジェクトが存在する場合、アプリケーションは、アプリケーションのローカル名前空間内でオブジェクトを開く。アプリケーションがグローバル名前空間内でオブジェクトを変更しようと試みた場合、オペレーティングシステムは、そのオブジェクトをアプリケーションのローカル名前空間内にコピーして、そのローカルオブジェクトに対して操作が行われることを許す。

リソースがローカル名前空間内またはグローバル名前空間内に存在しない場合、開く操作は失敗する。

【００５１】

次に、図５を参照すると、典型的な流れ図が、システム設定に関するアクセス制御を提供する方法の動作を示している。図５は、システム設定に関連する例を示しているが、本発明の仮想化の態様は、その他のオブジェクト（例えば、名前付きオブジェクト）および名前空間に関して利用することもできる。図５では、オペレーティングシステムのような本発明の実施形態が、例えば、アプリケーションプログラムによって要求されたシステム設定に対する操作を分析する。詳細には、オペレーティングシステムは、５０２で、要求された操作が、システム設定を書き込む、または削除するかどうかを判定する。要求された操作が、システム設定を書き込む、または削除することがない（例えば、読み取り専用アクセスが要求された）場合、オペレーティングシステムは、５０４で、システム設定の仮想コピーが、現在、存在するかどうかを判定する。仮想コピーが存在する場合、オペレーティングシステムは、５０６で、その仮想コピーを識別し、５０８で、システム設定の仮想コピーに対して要求された操作を実行する。仮想コピーが存在しない場合、オペレーティングシステムは、５０８で、システム設定に対して要求された操作を実行する。

【００５２】

要求された操作が、システム設定を書き込む、または削除する場合、オペレーティングシステムは、５１０で、要求側のアプリケーションプログラムが、読み取り専用キーに関連付けられている（例えば、要求側のアプリケーションプログラムが、信頼されるインストーラではない）かどうかを判定する。要求側のアプリケーションプログラムが、読み取り専用アクセスに関連付けられている（例えば、オペレーティングシステムによって保持されるアクセス制御リストを介して）場合、オペレーティングシステムは、５１２で、要求された操作を不合格にするか、または拒否する。要求側のアプリケーションプログラムが、読み取り専用アクセスに関連付けられていない場合、オペレーティングシステムは、５１４で、要求された操作がシステムによって制限された（ｓｙｓｔｅｍ－ｒｅｓｔｒｉｃｔｅｄ）設定を書き込む、または削除するかどうかを判定する。要求された操作が、システムによって制限された設定を書き込む、または削除する場合、オペレーティングシステムは、５１６で、要求側のアプリケーションプログラムが、その操作を実行することを許可されているかどうかを判定する。例えば、オペレーティングシステムは、要求側のアプリケーションプログラムが、コンピューティングシステム上で管理者特権を有するかどうかを判定することができる。要求側のアプリケーションプログラムが、その操作を実行することを許可されている場合、オペレーティングシステムは、５０８で、要求された操作を実行する。要求側のアプリケーションプログラムが、その操作を実行することを許可されていない場合、オペレーティングシステムは、５１２で、要求された操作を不合格にするか、または拒否する。

【００５３】

要求された操作が、システムによって制限された設定を書き込む、または削除することをしない場合、オペレーティングシステムは、５１８で、要求された操作が、保護された設定（例えば、要求側のアプリケーションプログラムに関連付けられたシステム設定のコピー）であるかどうかを判定する。オペレーティングシステムは、要求された操作が保護された設定に関すると判定した場合、５２０で、その保護された設定を要求側のアプリケーションプログラムのアプリケーション識別子によって仮想化する。つまり、オペレーティングシステムは、システム設定の仮想コピーを識別し、５０８で、システム設定の識別された仮想コピーに対して要求された操作を実行する。オペレーティングシステムは、要求された操作が保護された設定に関してではないと判定した場合、５２２で、要求された操作が専用設定（例えば、要求側のアプリケーションプログラムに関連するシステム設定）であるかどうかを判定する。オペレーティングシステムは、要求された操作が専用設定に関すると判定した場合、５０８で、その専用システム設定に対して要求された操作を実行する。オペレーティングシステムは、要求された操作が専用設定に関してではないと判

定した場合、処理を終了し、その要求を暗黙に、または明示的に不合格にする。

【００５４】

アプリケーションが、ローカル名前空間からあるオブジェクトを削除しようと試み、同一の名前を有するグローバルオブジェクトが存在する場合、システムは、ローカルオブジェクトに削除済みとしてマークを付けるが、そのオブジェクトを名前空間内に残す。このため、システムは、そのオブジェクトに対するアプリケーションのクエリが、そのオブジェクトの名前を見るべきでないことを検出することができる。アプリケーションが、ローカル名前空間内に存在するが、グローバル名前空間内には存在しないオブジェクトを削除しようと試みた場合、システムは、そのローカルオブジェクトを削除する。オペレーティングシステム構成に依存して、グローバルオブジェクトを削除することにより、すべての対応するローカルオブジェクト群が削除されることがもたらされる可能性がある。システムは、対応するオブジェクトがこの形で削除されるべきかどうかをアプリケーションが規定できるようにすることができ、リソースプロバイダが、その規定をローカルオブジェクト上に格納する。また、グローバルオブジェクトを追加することにより、削除済みとしてマークが付けられたすべての対応するオブジェクトがすべてのローカル名前空間から削除されることがもたらされる可能性もある。

【００５５】

この設計では、アプリケーションは、自らがグローバル名前空間内で作業していると思っているが、現実には、アプリケーション独自の名前空間内で作業している。システムは、完全なパスのクエリ、列挙、およびその他の操作を扱って、アプリケーションに、アプリケーションがグローバル名前空間内で作業していると思わせる。例えば、名前空間列挙には、特定のディレクトリの下のすべてのファイルをリストアップすることが含まれる。システムは、規定された名前空間内ですべてのオブジェクトのクエリを行う（例えば、まず、ローカル名前空間から始めて、次にグローバル名前空間に進む）。システムは、ローカル名前空間内で見つかった複製のオブジェクトをグローバル名前空間列挙で無視する。また、列挙は、ローカル名前空間から削除済みとしてマークが付けられたオブジェクト、および対応するグローバル名前空間オブジェクトも無視する。

【００５６】

リソースを共有することが予期されるアプリケーション群に関して、オペレーティングシステムは、そのアプリケーション群を同一の仮想化アプリケーショングループ（例えば、同一の分離ＩＤ）に入れることができる。代替として、オペレーティングシステムは、名前空間の特定の部分が仮想化されるべきでないことを規定することができる。さらに別の代替では、アプリケーション群は、他のアプリケーション群がアクセスすることができる自らの仮想化された名前空間の部分を規定する。クライアントアプリケーションは、アクセスが所望されるアプリケーション群を規定する。クライアントアプリケーションが共有の仮想化された名前空間にアクセスした場合、オペレーティングシステムは、目標アプリケーション群の対応するエクスポートされた名前空間を探索する。

【００５７】

一部の環境では、オペレーティングシステムは、複数の仮想化レイヤを有することを所望する可能性がある。ユーザ別の仮想化レイヤ、およびアプリケーショングループ別の仮想化レイヤが存在することが可能である。複数の仮想化レイヤの様々な順序が、本発明の範囲に含まれる。この例では、ユーザ仮想化レイヤが、アプリケーション仮想化レイヤよりも優先される。したがって、オブジェクトに関するクエリ要求および開く要求はまず、現行ユーザの仮想化レイヤを調べ、次に、現行アプリケーショングループの仮想化レイヤを調べ、最後にグローバル名前空間を調べる。オペレーティングシステムは、見つかった最初のオブジェクトを戻し、仮想化レイヤまたはグローバル名前空間のいずれにもオブジェクトが存在しない場合、まったくオブジェクトを戻さない。書き込み操作に関しても同様に、オペレーティングシステムはまず、オブジェクトを開く。オブジェクトが最高優先順位のレイヤの中に存在する場合、そのオブジェクトに対して書き込み操作が行われる。オブジェクトが最高優先順位のレイヤの中に存在しない場合、オブジェクトは、最高優先

順位のレイヤの中にコピーされ、そのコピーされたオブジェクトに対して書き込み操作が行われる。作成操作は、最高優先順位のレイヤで行われるが、一部の実施形態におけるオペレーティングシステムは、コードが、特定の仮想化レイヤを優先として規定することを許すことができる。

【００５８】

同様に、オブジェクトを削除する場合、操作は、最高優先順位の仮想化レイヤで行われるが、一部の実施形態におけるオペレーティングシステムは、コードが、特定の仮想化レイヤを優先として規定することを許すことができる。正確なオブジェクトが見つかると、オペレーティングシステムは、そのオブジェクトが、該当するより低い優先順位の名前空間内に存在するかどうかを調べる。オブジェクトがより低い優先順位の名前空間内に存在する場合、対象の削除オブジェクトは、「削除済み」というマークを付けられて、その名前空間内に留まる。オブジェクトがより低い優先順位の名前空間内に存在しない場合、オブジェクトは、その名前空間から削除され、除去される。一部の構成では、オペレーティングシステムは、より高い優先順位の名前空間から対応するオブジェクトを削除することができる。しかし、より高い優先順位のオブジェクトの作成者が、その場合、オブジェクトが削除されないように規定することができる。

【００５９】

より低い優先順位の名前空間にオブジェクトを追加する場合、オペレーティングシステムは、削除済みというマークが付けられたすべての対応するオブジェクトをより高い優先順位の名前空間から除去する。探索および除去は、目標の名前空間から始められ、削除済みというマークが付けられていない対応するオブジェクトが探索によって見つかると、またはすべての該当するレイヤの探索が済むと、次の該当する、より高い優先順位のレイヤに進む。

【００６０】

列挙操作は、コンテキストおよびグローバル名前空間に関してすべての該当する仮想化レイヤを考慮に入れる。列挙は、最高優先順位の該当する名前空間から始められ、グローバル名前空間まで下方に進む。列挙が、削除済みというマークが付けられたオブジェクトを見つけると、そのオブジェクトに関する列挙は、より低い優先順位の名前空間において無視される。また、列挙は、より高い優先順位の名前空間において以前に見つかっている対応するオブジェクトも無視する。

【００６１】

オペレーティングシステムに関する内部オブジェクト保護

オペレーティングシステムは、様々なオブジェクトを作成する。そのオブジェクトの一部は、アプリケーション群によるアクセス用であり、別の一部（例えば、内部オブジェクト）には、オペレーティングシステムコンポーネント群だけがアクセスすることができる。オペレーティングシステムは、そのオブジェクト群に関するアクセス権（例えば、開くアクセスおよび読み取りアクセス）を定義する。

【００６２】

一実施形態では、内部オペレーティングシステムオブジェクト群には、内部オペレーティングシステムコンポーネント群だけがアクセスすることができなければならない。外部コードが内部オブジェクト群にアクセスするのを防止するため、オペレーティングシステムは、内部オペレーティングシステムコンポーネント群だけによるアクセス用に内部オブジェクトにマークを付ける。内部オペレーティングシステムコードとして実行されているランタイムオブジェクトには、内部オペレーティングシステムＩＤが関連付けられる。したがって、ランタイムオブジェクトが内部オブジェクトにアクセスしようと試みた場合、オペレーティングシステムは、そのランタイムオブジェクトが内部オペレーティングシステムＩＤに関連付けられているかどうかを調べる。ランタイムオブジェクトが内部オペレーティングシステムＩＤを有する場合、オペレーティングシステムは、アクセスを許す。有さない場合、オペレーティングシステムは、適切なアクションを実施する。適切なアクションには、アクセスを拒否すること、アクセスの試みをログ記録することなどが含まれ

ることが可能である。

【００６３】

内部オペレーティングシステムコンポーネントがオブジェクトを作成した場合、オブジェクトは、作成者が、外部アクセスに供されるとしてオブジェクトに特別にマークを付けない限り、内部オペレーティングシステムコンポーネント群だけによるアクセス用にマークが付けられる。オペレーティングシステムは、ストア、マニフェスト、構成ファイル、デジタル署名などからのリソース情報を使用して、内部オブジェクトにオフラインでマークを付けることができる。

【００６４】

一部のオペレーティングシステムコンポーネントは、ミドルウェアコンポーネントとして分類され、これは、そのコンポーネント群がオペレーティングシステムの一部ではあっても、外部アプリケーション群もアクセスすることが許されているいくつかの特殊な予想状況（ｅｘｐｅｃｔａｔｉｏｎｓ）を除き、内部オブジェクト群にアクセスすべきでないことを意味する。一部の実施形態におけるオペレーティングシステムは、ミドルウェアコンポーネント群が、特殊な例外の内部オブジェクトを使用することを止め、外部オブジェクトに移行することを所望する。この問題に対処するため、オペレーティングシステムは、ミドルウェアアプリケーションＩＤをミドルウェアコンポーネント群に関連付ける。特殊な例外の内部オブジェクト群には、廃止（ｄｅｐｒｅｃａｔｅｄ）属性でさらにマークが付けられる。ミドルウェアコンポーネントが廃止オブジェクトにアクセスする場合、システムは、アクセスを監査すること、および／またはアクセスを阻止することなどの適切なアクションで応対する。外部オブジェクト群、または他の外部オブジェクト群、または他の内部オブジェクト群を廃止するために、ミドルウェア廃止リソース検出をより一般的に適用してもよい。

【００６５】

アプリケーションプログラムの削除

次に、図６を参照すると、典型的な流れ図が、アプリケーションＩＤ情報を使用してアプリケーション取り消し（ｕｎｄｏ）を実行するための方法を示している。詳細には、この流れ図は、アプリケーションプログラム、およびアプリケーションプログラムのコンポーネント（例えば、ファイルおよびリソース）に関連するアプリケーション識別子を介して、インストール済みのアプリケーションプログラムをコンピューティングシステムから完全に削除する方法を示している。既存のアプリケーションプログラム群は、アンインストール機能を現在、サポートしている可能性があるが、現在のオペレーティングシステムは、特定のアプリケーションプログラムに関連するコンポーネントが、アンインストールプロセス中に削除されることを確実にするメカニズムを欠いている。本発明の実施形態は、いずれのファイルが特定のアプリケーションプログラムに関連しているかを記録にとるデータストア（例えば、ログ）を保持する。このため、アプリケーションがアンインストールされる場合、オペレーティングシステムは、アンインストールプロセスによって後に残されたあらゆるファイル（例えば、アプリケーションの名前空間内で仮想化されたファイルを含む）を識別し、削除する。これは、オペレーティングシステムの動作を拡張する、または変更することにより作成された要素、または仮想化の他の（例えば、より低い）レベルで作成された要素を含むアプリケーションのすべての要素をアンインストールすることにより、アプリケーションプログラムのより完全なアンインストールを提供する。また、この方法は、アプリケーションプログラムがコンピューティングシステムからアンインストールされた後に、インストールされたアプリケーションにしばしば付随するスパイウェア、アドウェア、またはその他の不要なアプリケーションプログラム群を削除するのにも役立つ。

【００６６】

一部の実施形態では、削除されるべき特定のアプリケーションプログラムは、コンピューティングシステムにインストールされた複数のアプリケーションプログラムの１つである。本発明の実施形態が、６０２で、その特定のアプリケーションプログラムをアンイン

ストールする要求を受け取る。要求は、例えば、コンピューティングシステムのユーザを発信元とすることが可能である。代替として、要求は、オペレーティングシステムの現在のバージョンをインストールする前に、アプリケーションプログラムの以前のバージョンをアンインストールするアップグレードユーティリティによって生成されることが可能である。本発明の実施形態は、６０４で、その特定のアプリケーションプログラムに関連付けられた識別子を判定する。例えば、識別子は、アプリケーションプログラムの一部であること、またはメモリ領域の中に別個に格納されていることが可能である。本発明の実施形態は、６０６で、判定された識別子を介して、その特定のアプリケーションプログラムにだけ関連する１つまたは複数のファイルを識別する。つまり、識別されたファイルは、コンピューティングシステムにインストールされているその他のアプリケーションプログラムのいずれにも関連していない。システム上の各ファイルには、少なくとも１つのアプリケーション識別子が関連付けられているので、その特定のアプリケーションプログラムだけに関連するファイルの識別は、判定された識別子の探索を実行することでもたらされる。本発明の実施形態は、６０８で、識別されたファイルを削除する。一実施形態では、本発明は、あらゆるユーザファイル（例えば、ワードプロセッシングドキュメント、スプレッドシートドキュメント）を削除することを回避する。

【００６７】

さらに、その特定のアプリケーションプログラムをインストールすることに応答して適用されたシステム設定またはリソースの変更が、６１０で識別され、６１２で元に戻される。例えば、アプリケーションプログラムのインストール中、コンピューティングシステムに適用されたあらゆるシステム設定が、本発明の実施形態によってログ記録され、保持される。アプリケーションによってファイルおよびシステム設定に対して行われた変更には、所有権追跡のためにタグが付けられる。ログは、変更のそれぞれをインストールされているアプリケーションプログラムのアプリケーション識別子に関連付ける。一実施形態では、ログは、変更の１つまたは複数のロールバックを可能にするように保持される。例えば、ユーザが、システムに対して行われた最新の変更を元に戻すことを所望する可能性がある。別の例では、オペレーティングシステムが、特定のアプリケーションプログラムに関連する変更をロールバックすることにより、その特定のアプリケーションプログラムの完全なアンインストールを実行する。アプリケーションプログラムの削除またはアンインストール中、本発明の実施形態は、判定された識別子を使用して、アンインストールされているアプリケーションプログラムのアプリケーション識別子に関連する適用された設定または変更を識別し、元に戻すか、または別の形で除去する。例えば、ファイルタイプ関連付けに対する変更をログ記録して、アプリケーションプログラムをアンインストールすることにより、関連するアプリケーションプログラムを有さない特定のファイルタイプが残されないようにすることができる。つまり、アプリケーションプログラムのインストール中にファイルタイプ関連付けが行われた場合、アプリケーションプログラムをアンインストールする際に、そのファイルタイプ関連付けが元に戻される。

【００６８】

典型的な動作環境

図７は、汎用コンピューティングデバイスの一例をコンピュータ１３０の形態で示している。本発明の一実施形態では、コンピュータ１３０のようなコンピュータ、または他のコンピューティングシステムが、本明細書で例示し、説明するその他の図において使用するのに適している。コンピュータ１３０は、１つまたは複数のプロセッサまたは処理装置１３２、およびシステムメモリ１３４を有する。図示した実施形態では、システムバス１３６が、システムメモリ１３４からプロセッサ１３２までを含む様々なシステムコンポーネントを結合している。バス１３６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、アクセラレーテッドグラフィックスポート（ａｃｃｅｌｅｒａｔｅｄ　ｇｒａｐｈｉｃｓ　ｐｏｒｔ）、およびプロセッサバスまたはローカルバスを含め、いくつかのタイプのバス構造のいずれかの１つまたは複数を表す。例として、限定としてではなく、そのようなアーキテクチャには、インダストリス

タンダードアーキテクチャ（Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ）（ＩＳＡ）バス、マイクロチャネルアーキテクチャ（Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈｉｔｅｃｔｕｒｅ）（ＭＣＡ）バス、エンハンストＩＳＡ（Ｅｎｈａｎｃｅｄ ＩＳＡ）（ＥＩＳＡ）バス、ビデオエレクトロニクススタンダーズアソシエーション（Ｖｉｄｅｏ Ｅｌｅｃｔｒｏｎｉｃｓ Ｓｔａｎｄａｒｄｓ Ａｓｓｏｃｉａｔｉｏｎ）（ＶＥＳＡ）ローカルバス、およびメザニン（Ｍｅｚｚａｎｉｎｅ）バスとしても知られるペリフェラルコンポーネントインターコネクト（Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ）（ＰＣＩ）バスが含まれる。

【００６９】

コンピュータ１３０は、通常、少なくとも何らかの形態のコンピュータ可読媒体を有する。揮発性媒体と不揮発性媒体、リムーバブルな媒体とリムーバブルでない媒体がともに含まれるコンピュータ可読媒体は、コンピュータ１３０がアクセスすることができる任意の利用可能な媒体であることが可能である。例として、限定としてではなく、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む。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群、またはその他のデータなどの情報を格納するために任意の方法または技術で実装された揮発性媒体および不揮発性媒体、リムーバブルな媒体およびリムーバブルでない媒体が含まれる。例えば、コンピュータ記憶媒体には、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリまたは他のメモリ技術、ＣＤ－ＲＯＭ、デジタルバーサタイルディスク（ＤＶＤ）または他の光ディスクストレージ、磁気カセット、磁気テープ、磁気ディスクストレージまたは他の磁気記憶装置、または所望の情報を格納するのに使用することができ、コンピュータ１３０がアクセスすることができる他の任意の媒体が含まれる。通信媒体は、通常、搬送波などの変調されたデータ信号、または他のトランスポートメカニズムで、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群、またはその他のデータを実体化し、あらゆる情報配信媒体が含まれる。信号内に情報を符号化するような形で特性の１つまたは複数が設定されている、または変更されている変調されたデータ信号について、当業者は熟知している。有線ネットワークまたは直接有線接続などの有線媒体、および音響媒体、ＲＦ媒体、赤外線媒体、およびその他の無線媒体などの無線媒体が、通信媒体の例である。また、以上のいずれかの組合せも、コンピュータ可読媒体の範囲に含まれる。

【００７０】

システムメモリ１３４は、リムーバブルなメモリおよび／またはリムーバブルでないメモリ、揮発性メモリおよび／または不揮発性メモリの形態でコンピュータ記憶媒体を含む。図示した実施形態では、システムメモリ１３４は、読み取り専用メモリ（ＲＯＭ）１３８およびランダムアクセスメモリ（ＲＡＭ）１４０を含んでいる。始動中などにコンピュータ１３０内部の要素間で情報を転送するのを助ける基本ルーチンを含む基本入出力システム１４２（ＢＩＯＳ）が、通常、ＲＯＭ１３８の中に格納される。ＲＡＭ１４０は、通常、処理装置１３２が即時にアクセスすることができる、および／または処理装置１３２によって現在処理されている、データおよび／またはプログラムモジュール群を含む。例として、限定としてではなく、図７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群１４６、その他のプログラムモジュール群１４８、およびプログラムデータ１５０を示している。

【００７１】

コンピュータ１３０は、その他のリムーバブル／リムーバブルでない、揮発性／不揮発性のコンピュータ記憶媒体も含むことができる。例えば、図７は、リムーバブルでない不揮発性の磁気媒体に対して読み取りまたは書き込みを行うハードディスクドライブ１５４を示している。また、図７は、リムーバブルな不揮発性の磁気ディスク１５８に対して読み取りまたは書き込みを行う磁気ディスクドライブ１５６、およびＣＤ－ＲＯＭまたはその他の光媒体などのリムーバブルな不揮発性の光ディスク１６２に対して読み取りまたは書き込みを行う光ディスクドライブ１６０も示している。典型的な動作環境において使用することができるその他のリムーバブルな／リムーバブルでない、揮発性／不揮発性のコ

ンピュータ記憶媒体には、磁気テープカセット、フラッシュメモリカード、デジタルバーサタイルディスク、デジタルビデオテープ、ソリッドステートＲＡＭ、ソリッドステートＲＯＭなどが含まれるが、以上には限定されない。ハードディスクドライブ１５４、磁気ディスクドライブ１５６、および光ディスクドライブ１６０は、通常、インターフェース１６６のような不揮発性メモリインターフェースでシステムバス１３６に接続される。

【００７２】

以上に説明し、図７に示すドライブ群およびその他の記憶装置群、ならびに関連するコンピュータ記憶媒体群により、コンピュータ可読命令、データ構造、プログラムモジュール、およびその他のデータのストレージがコンピュータ１３０に提供される。図７では、例えば、ハードディスクドライブ１５４が、オペレーティングシステム１７０、アプリケーションプログラム群１７２、その他のプログラムモジュール群１７４、およびプログラムデータ１７６を格納しているのが示されている。以上のコンポーネントは、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群１４６、その他のプログラムモジュール群１４８、およびプログラムデータ１５０と同じであることも、異なることも可能であることに留意されたい。オペレーティングシステム１７０、アプリケーションプログラム群１７２、その他のプログラムモジュール群１７４、およびプログラムデータ１７６に、ここでは、少なくともそれらが異なるコピーであることを示すために異なる符号を付けている。

【００７３】

ユーザは、キーボード１８０やポインティングデバイス１８２（例えば、マウス、トラックボール、ペン、タッチパッド）などの入力デバイス群またはユーザインターフェース選択デバイス群を介してコンピュータ１３０にコマンドおよび情報を入力することができる。その他の入力デバイス（図示せず）には、マイク、ジョイスティック、ゲームパッド、サテライトディッシュ、スキャナなどが含まれることが可能である。以上、およびその他の入力デバイス群は、システムバス１３６に結合されたユーザ入力インターフェース１８４を介して処理装置１３２に接続されるが、パラレルポート、ゲームポート、またはユニバーサルシリアルバス（ＵＳＢ）などの他のインターフェースおよびバス構造で接続することもできる。モニタ１８８または他のタイプのディスプレイデバイスも、ビデオインターフェース１９０のようなインターフェースを介してシステムバス１３６に接続される。モニタ１８８に加えて、コンピュータは、しばしば、出力周辺インターフェース（図示せず）を介して接続することができるプリンタやスピーカなどの他の周辺出力デバイス群（図示せず）も含む。

【００７４】

コンピュータ１３０は、リモートコンピュータ１９４のような１つまたは複数のリモートコンピュータに対する論理接続を使用するネットワーク化された環境において動作することができる。リモートコンピュータ１９４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、またはその他の一般的なネットワークノードであることが可能であり、通常、コンピュータ１３０に関連して前述した要素の多く、またはすべてを含む。図７に示した論理接続は、ローカルエリアネットワーク（ＬＡＮ）１９６およびワイドエリアネットワーク（ＷＡＮ）１９８を含むが、その他のネットワークを含むことも可能である。ＬＡＮ１３６および／またはＷＡＮ１３８は、有線ネットワーク、無線ネットワーク、有線ネットワークと無線ネットワークの組合せなどであることが可能である。そのようなネットワーキング環境は、オフィス、企業全体のコンピュータネットワーク、イントラネット、および世界的なコンピュータネットワーク群（例えば、インターネット）において一般的である。

【００７５】

ローカルエリアネットワーキング環境で使用される場合、コンピュータ１３０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１８６を介してＬＡＮ１９６に接続される。ワイドエリアネットワーキング環境で使用される場合、コンピュータ１３０は、通常、インターネットなどのＷＡＮ１９８を介した通信を確立するためのモデム１７８

、またはその他の手段を含む。内部にあることも、外部にあることも可能なモデム１７８は、ユーザ入力インターフェース１８４、またはその他の適切なメカニズムを介してシステムバス１３６に接続される。ネットワーク化された環境では、コンピュータ１３０に関連して示したプログラムモジュール群、またはプログラムモジュール群の部分は、リモートメモリ記憶装置（図示せず）の中に格納することができる。例として、限定としてではなく、図７は、メモリデバイス上に常駐するものとしてリモートアプリケーションプログラム群１９２を示している。図示したネットワーク接続群は、典型的であり、コンピュータ間で通信リンクを確立するその他の手段を使用してもよい。

【００７６】

一般に、コンピュータ１３０のデータプロセッサ群は、コンピュータの様々なコンピュータ可読記憶媒体の中に異なる時点で格納された命令を使用してプログラミングされる。プログラム群およびオペレーティングシステム群は、通常、例えば、フロッピー（登録商標）ディスク上またはＣＤ－ＲＯＭ上で配布される。フロッピー（登録商標）ディスクまたはＣＤ－ＲＯＭから、プログラム群およびオペレーティングシステム群は、コンピュータの２次メモリの中にインストールされる、または読み込まれる。実行中に、プログラム群およびオペレーティングシステム群は、コンピュータの１次電子メモリの中に少なくとも部分的に読み込まれる。本明細書で説明する本発明は、以上、およびその他の様々なタイプのコンピュータ可読記憶媒体を、そのような媒体が、マイクロプロセッサまたはその他のデータプロセッサと連携して以下に説明するステップを実施するための命令群またはプログラム群を含む場合に含む。また、本発明は、本明細書で説明する方法および技術に従ってプログラミングされている場合、コンピュータ自体も含む。

【００７７】

例示のため、オペレーティングシステムなどのプログラム群およびその他の実行可能なプログラムコンポーネント群を本明細書では、別個のブロックとして示している。しかし、そのようなプログラム群およびコンポーネント群は、様々な時点で、コンピュータの異なる記憶コンポーネントの中に存在し、コンピュータのデータプロセッサによって実行されることが認識されよう。

【００７８】

コンピュータ１３０を含め、典型的なコンピューティングシステム環境に関連して説明しているが、本発明は、他の多数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コンピューティングシステム構成でも機能する。コンピューティングシステム環境は、本発明の用途または機能の範囲について何ら限定を示唆するものではない。さらに、コンピューティングシステム環境が、典型的な動作環境において例示したコンポーネントのいずれの１つ、または組合せに関連する依存関係または要件も有していると解釈してはならない。本発明で使用するのに適する可能性がある周知のコンピューティングシステム、コンピューティング環境、および／またはコンピューティング構成の例には、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、プログラマブル家庭用電化製品、モバイル電話機、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、以上のシステムまたはデバイスのいずれかを含む分散コンピューティング環境などが含まれるが、以上には限定されない。

【００７９】

本発明は、１つまたは複数のコンピュータ、またはその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群などの、コンピュータ実行可能命令の一般的な文脈で説明することができる。一般に、プログラムモジュールには、特定のタスクを実行する、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、およびデータ構造などが含まれるが、以上には限定されない。本発明は、通信ネットワークを介してリンクされたリモート処理装置によってタスクが実行される分散コンピューティング環境においても実施することができる。分散コンピューティング環境では、プログラムモジュール群は、メモリ記憶装置を含むローカルコンピュータ記憶媒体とリモートコ

ンピュータ記憶媒体の両方の中に配置することができる。

【００８０】

ソフトウェアアーキテクチャの文脈におけるインターフェースには、ソフトウェアモジュール、コンポーネント、コード部分、またはその他のシーケンスのコンピュータ実行可能命令が含まれる。インターフェースには、例えば、第１のモジュールが、第１のモジュールに代行して計算タスクを実行する第２のモジュールにアクセスすることが含まれる。第１のモジュールおよび第２のモジュールには、一例では、オペレーティングシステムによって提供されるようなアプリケーションプログラミングインターフェース（ＡＰＩ）、コンポーネントオブジェクトモデル（ＣＯＭ）インターフェース群（例えば、ピアツーピアアプリケーション通信用の）、および拡張マークアップ言語メタデータ交換フォーマット（ＸＭＩ）インターフェース群（例えば、Ｗｅｂサービス間の通信用の）が含まれる。

【００８１】

インターフェースは、Ｊａｖａ（登録商標）２　Ｐｌａｔｆｏｒｍ　Ｅｎｔｅｒｐｒｉｓｅ　Ｅｄｉｔｉｏｎ（Ｊ２ＥＥ）、ＣＯＭ、または分散ＣＯＭ（ＤＣＯＭ）の例におけるように、緊密に結合された同期実装であることが可能である。代替として、またはさらに、インターフェースは、Ｗｅｂサービス（例えば、シンプルオブジェクトアクセスプロトコルを使用する）におけるような緩く結合された非同期実装であることが可能である。一般に、インターフェースには、次の特性の任意の組合せが含まれる。すなわち、緊密に結合された特性、緩く結合された特性、同期の特性、および非同期の特性である。さらに、インターフェースは、標準のプロトコル、独自のプロトコル、または標準のプロトコルと独自のプロトコルの任意の組合せに適合することが可能である。

【００８２】

本明細書で説明するインターフェース群はすべて、単一のインターフェースの一部であることも、別々のインターフェースとして実装されることも、その任意の組合せであることも可能である。インターフェース群は、ローカルまたは遠隔で実行されて機能を提供することが可能である。さらに、インターフェース群は、本明細書に示し、説明した機能に追加した機能、またはより少ない機能を含むことも可能である。

【００８３】

動作の際、コンピュータ１３０は、図に示したようなコンピュータ実行可能命令を実行して、アプリケーションプログラムおよびリソースに関連する特権に応じて、アプリケーションプログラムがリソースにアクセスすることを許可する。図に示し、本明細書で説明するシステムおよび方法は、一部は当技術分野で周知である技術を使用して、ソフトウェアまたはハードウェアとして実装することができる。

【００８４】

マニフェストの例

以下の例は、本発明をさらに例示する。以下の例のいくつかは、レジストリへの言及を含むが、本発明の実施形態は、レジストリに限定されない。本発明の実施形態は、システム設定を格納するための任意のメカニズムで機能する。属性は、一部のメカニズムでは継承されるが、他のメカニズムでは、継承は保証されない。以下のテーブル１は、マニフェストの中の典型的な特権をリストアップし、レベルのそれぞれに関連するリソース保護のタイプを説明している。

【００８５】

【表３】

特権	保護のタイプ
readOnlyIgnoreWrites	読み取り専用-この特権に関連するファイルまたは設定は、インストール時または調整（servicing）（例えば、アップグレード）時にオペレーティングシステムだけが変更することができる。そのファイルまたは設定に書き込もうとするその他の試みは、暗黙に無視される（例えば、書き込みがまったく行われなくても、成功応答が戻される）。
readOnlyFailWrites	読み取り専用-この特権に関連するファイルまたは設定は、インストール時または調整（例えば、アップグレード）時にオペレーティングシステムだけが変更することができる。そのファイルまたは設定に書き込もうとするその他の試みは、明示的に無視される（例えば、失敗応答が戻される）。
OSOnlyIgnoreWrites	この特権に関連するファイルまたは設定は、オペレーティングシステムコンポーネントだけが変更することができる。そのファイルまたは設定に書き込もうとするその他の試みは、暗黙に無視される（例えば、書き込みがまったく行われなくても、成功応答が戻される）。
OSOnlyFailWrites	この特権に関連するファイルまたは設定は、オペレーティングシステムコンポーネントだけが変更することができる。そのファイルまたは設定に書き込もうとするその他の試みは、明示的に無視される（例えば、失敗応答が戻される）。
change recording	この特権に関連する設定のために格納された異なる値は、アプリケーション別に保持されるが、グローバル（書き込みを行う最後のアプリケーション）に可視である。現在のグローバルな値は、そのグローバルな値がアンインストール中のアプリケーションに属する場合、アプリケーションアンインストール後に、最新アプリケーションアルゴリズムを使用してロールバックされる。
applicationVirtualized	この特権に関連するファイルまたは設定の変更は、アプリケーションからの書き込み要求に応答して、アプリケーション別に仮想化される。
userVirtualized	この特権に関連するファイルまたは設定の変更は、ユーザからの書き込み要求に応答して、ユーザ別に仮想化される。
applicationAndUserVirtualized	この特権に関連するファイルまたは設定の変更は、アプリケーションプログラムを実行するユーザからの書き込み要求に応答して、ユーザ別およびアプリケーション別に仮想化される。
notProtected	この特権に関連するファイルおよびシステム設定には、保護または軽減がまったく関連付けられていない。適切な許可を有する任意のサードパーティアプリケーションまたは管理者が、それらのファイルおよび設定を変更することができる。

表１．典型的な特権

【００８６】

別の例では、見本のオペレーティングシステムコンポーネント（例えば、「ＣｏｍｐＮａｍｅ」）が、そのコンポーネントに関連するリソースに関して以下の保護動作を所望する。

【００８７】
【表４】

ディレクトリ名	保護動作
C:＼Comp Name＼	IDベースのアクセス特権
C:＼Comp Name＼Sub＼	アプリケーション仮想
C:＼Common Files＼Shared＼Comp Name＼	IDベースのアクセス特権（書き込みを失敗させる）

表２．典型的なディレクトリおよび所望される保護動作

【００８８】
【表５】

ディレクトリ名	ファイル名	保護動作
C:＼	CompName.dll	IDベースのアクセス特権
C:＼Comp Name＼	Samplesys	IDベースのアクセス特権
C:＼Comv Name＼Sub＼	ComnName.dat	アプリケーション仮想
C:＼Common Files＼Shared＼Comp Name＼	Common.dll	IDベースのアクセス特権（書き込みを失敗させる）
C:＼Common Files＼Shared	Base.dll	アプリケーション仮想

表３．典型的なファイルおよび所望される保護動作

【００８９】
【表６】

キー名	保護動作
HKLM＼Software＼Comp Name＼	IDベースのアクセス特権
HKLM＼Software＼Comp Name＼SubKey＼	IDベースのアクセス特権
HKLM＼Software＼Comp Name＼Settings＼	アプリケーション仮想
HKCR＼.comp＼	IDベースのアクセス特権

表４．典型的なレジストリキーおよび所望の保護動作

【００９０】
【表７】

キー名	値の名前	保護動作
HKLM＼Software＼Comp Name＼	Version	IDベースのアクセス特権
HKLM＼Software＼Comp Name＼SubKey＼	SubValue	IDベースのアクセス特権
HKCR＼.comp＼	(既定)	IDベースのアクセス特権
HKEY_USERS＼.Default＼Environment＼	MyEnv	IDベースのアクセス特権

表５．典型的なレジストリ値および所望の保護動作

【００９１】
本明細書で示し、説明する方法の実行または遂行の順序は、特に明記しない限り、必須ではない。つまり、方法の要素は、特に明記しない限り、任意の順序で実行することができ、方法は、本明細書で開示したよりも多い、または少ない要素を含んでいてもよい。
【００９２】

本発明の諸要素、または本発明の実施形態を概説する際、冠詞、「ある（ａ）」、「ある（ａｎ）」、「その（ｔｈｅ）」および「前記（ｓａｉｄ）」は、要素の１つまたは複数が存在することを意味するものとする。「含む（ｃｏｍｐｒｉｓｉｎｇ）」、「含む（ｉｎｃｌｕｄｉｎｇ）」、および「有する（ｈａｖｉｎｇ）」という用語は、包含的であり、リストアップした要素以外に追加の要素が存在する可能性があることを意味するものとする。

【００９３】

以上に鑑みて、本発明のいくつかの目的が達せられ、その他の有利な結果が実現されることが理解されよう。

【００９４】

本発明の範囲を逸脱することなく、以上の構築物、製品、および方法で変更を行うことができるので、以上の説明に含まれ、添付の図面に示したすべての内容は、例示として、限定としてではなく解釈されなければならない。

【図面の簡単な説明】

【００９５】

【図１】リソースへのアクセスをアプリケーションプログラムに与えるオペレーティングシステムの典型的な実施形態を示す図である。

【図２】アクセス制御方法の動作を示す典型的な流れ図である。

【図３】様々なリソースを保護するための軽減アーキテクチャを示す典型的な流れ図である。

【図４】ファイル、システム設定、および拡張機能に関するアクセス制御を提供する方法の動作を示す典型的な流れ図である。

【図５】システム設定に関するアクセス制御を提供する方法の動作を示す典型的な流れ図である。

【図６】コンピューティングシステムからインストール済みのアプリケーションプログラムを削除する動作を示す典型的な流れ図である。

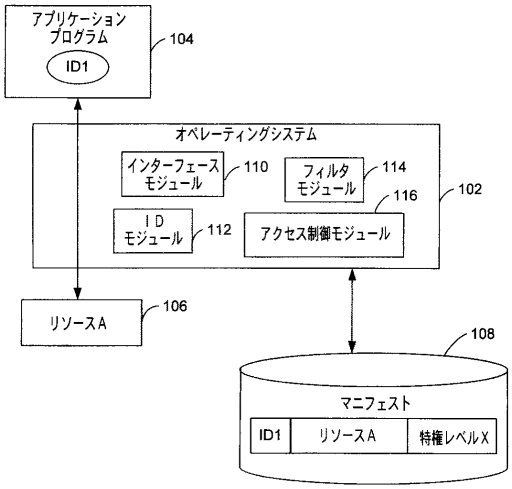
【図７】本発明を実施することができる適切なコンピューティングシステム環境の一例を示すブロック図である。

【符号の説明】

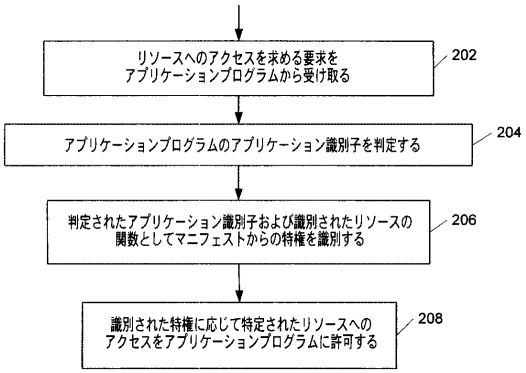
【００９６】

１３２　処理装置
１３４　システムメモリ
１３６　システムバス
１４４　オペレーティングシステム
１４６　アプリケーションプログラム群
１４８　他のプログラムモジュール群
１５０　プログラムデータ
１６６　不揮発性メモリインターフェース
１７０　オペレーティングシステム
１７２　アプリケーションプログラム群
１７４　他のプログラムモジュール群
１７６　プログラムデータ
１８４　ユーザ入力インターフェース

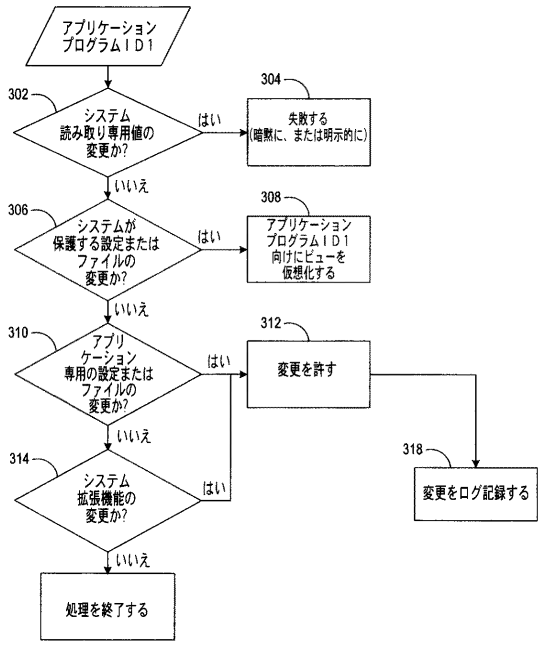
【図１】


アプリケーション
プログラム
ID1
104
オペレーティングシステム
インターフェース
モジュール
110
フィルタ
モジュール
114
116
102
ＩＤ
モジュール
112
アクセス制御モジュール
リソースA
106
108
マニフェスト
ID1
リソースA
特権レベルX

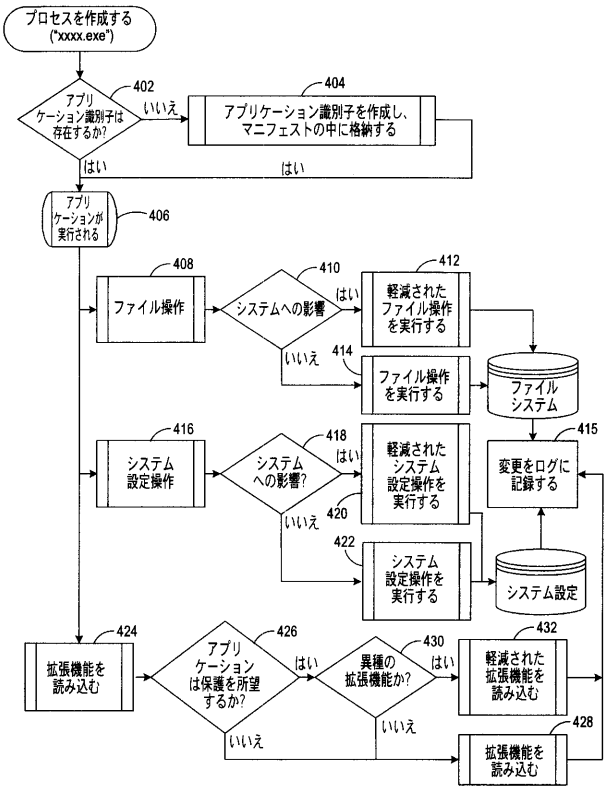
【図２】


リソースへのアクセスを求める要求を
アプリケーションプログラムから受け取る
202
アプリケーションプログラムのアプリケーション識別子を判定する
204
判定されたアプリケーション識別子および識別されたリソースの
関数としてマニフェストからの特権を識別する
206
識別された特権に応じて特定されたリソースへの
アクセスをアプリケーションプログラムに許可する
208

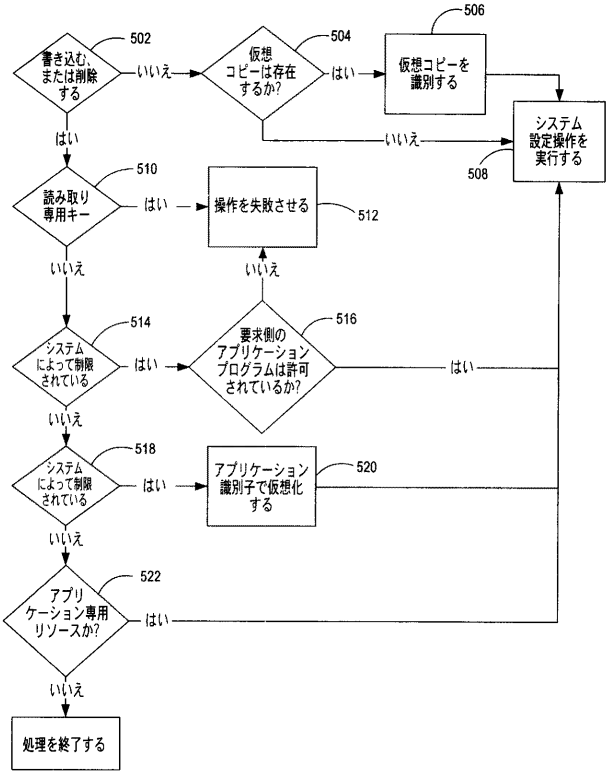
【図３】


アプリケーション
プログラムＩＤ１
302
システム
読み取り専用値の
変更か?
はい
304
失敗する
(暗黙に、または明示的に)
いいえ
306
システムが
保護する設定または
ファイルの
変更か?
はい
308
アプリケーション
プログラムＩＤ１
向けにビューを
仮想化する
いいえ
310
アプリ
ケーション
専用の設定または
ファイルの
変更か?
はい
312
変更を許す
いいえ
314
システム
拡張機能の
変更か?
はい
318
変更をログ記録する
いいえ
処理を終了する

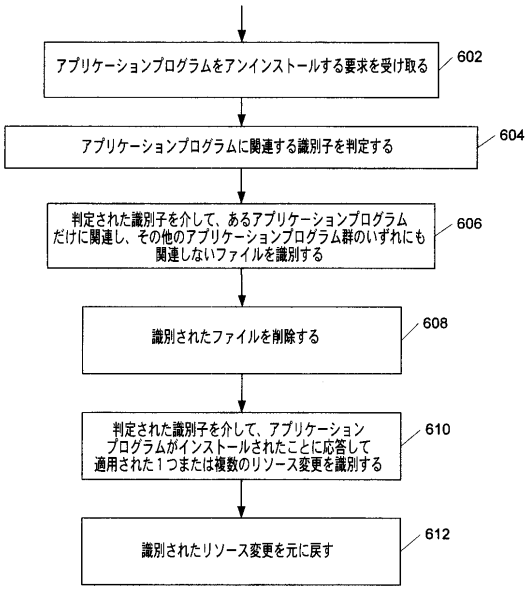
【図４】


プロセスを作成する
("xxxx.exe")
アプリ
ケーション識別子は
存在するか?
402
いいえ
404
アプリケーション識別子を作成し、
マニフェストの中に格納する
はい
はい
アプリ
ケーションが
実行される
406
ファイル操作
408
システムへの影響
410
はい
軽減された
ファイル操作
を実行する
412
いいえ
414
ファイル操作
を実行する
ファイル
システム
システム
設定操作
416
システム
への影響?
418
はい
軽減された
システム
設定操作を
実行する
420
415
変更をログに
記録する
いいえ
422
システム
設定操作
を実行する
システム設定
拡張機能を
読み込む
424
アプリ
ケーション
は保護を所望
するか?
426
はい
異種の
拡張機能か?
430
はい
軽減された
拡張機能を
読み込む
432
いいえ
いいえ
428
拡張機能を
読み込む

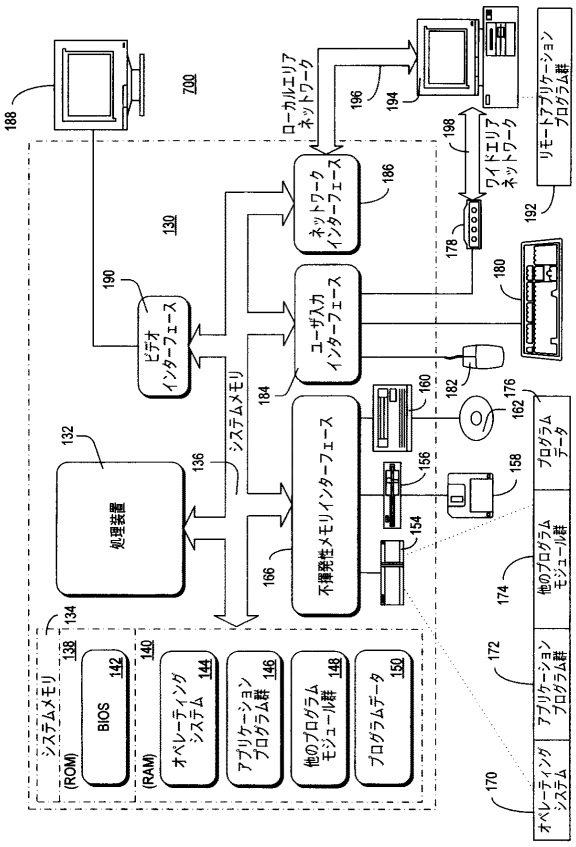
【図５】


書き込む、または削除する
502
いいえ
仮想コピーは存在するか?
504
はい
仮想コピーを識別する
506
いいえ
システム設定操作を実行する
508
はい
読み取り専用キー
510
はい
操作を失敗させる
512
いいえ
いいえ
システムによって制限されている
514
はい
要求側のアプリケーションプログラムは許可されているか?
516
はい
いいえ
システムによって制限されている
518
はい
アプリケーション識別子で仮想化する
520
いいえ
アプリケーション専用リソースか?
522
はい
いいえ
処理を終了する

【図６】


アプリケーションプログラムをアンインストールする要求を受け取る
602
アプリケーションプログラムに関連する識別子を判定する
604
判定された識別子を介して、あるアプリケーションプログラムだけに関連し、その他のアプリケーションプログラム群のいずれにも関連しないファイルを識別する
606
識別されたファイルを削除する
608
判定された識別子を介して、アプリケーションプログラムがインストールされたことに応答して適用された１つまたは複数のリソース変更を識別する
610
識別されたリソース変更を元に戻す
612

【図７】


700
188
ローカルエリアネットワーク
196
194
198
ワイドエリアネットワーク
リモートアプリケーションプログラム群
192
ネットワークインターフェース
186
130
178
180
ビデオインターフェース
190
ユーザ入力インターフェース
システムメモリ
184
160
182
176
162
132
136
処理装置
不揮発性メモリインターフェース
156
158
プログラムデータ
154
他のプログラムモジュール群
174
134
138
140
172
アプリケーションプログラム群
システムメモリ
(ROM)
BIOS
142
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群
146
他のプログラムモジュール群
148
プログラムデータ
150
(RAM)
170
オペレーティングシステム

フロントページの続き

(72)発明者 フレディ エル．アーロン
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 ジョナサン シー．ルー
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 ジュード ジェイコブ カバラム
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 ローズマリー フィッツシモンズ
アメリカ合衆国 ９８０５２ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション内
(72)発明者 ティモシー ディー．ヌーナン
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 バレリー ブイ．ツリク
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 デビッド ビー．プロバート
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 ドラゴス シー．サンボティン
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 ジュヌビエーブ フェルナンデス
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 エリック リ
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 オースティン レクター
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

Ｆターム(参考) 5B014 FA11 FB00 FB03 GD47
5B017 AA02 BA06 BA09 CA16
5B076 FB02