(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4974487号
(P4974487)
(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20日(2012.4.20)

(51)Int.Cl.　　F I
G06K 17/00 (2006.01)　　G06K 17/00　F
G06F 13/10 (2006.01)　　G06F 13/10　320A

請求項の数 16　(全 41 頁)

(21)出願番号　特願2005-209986(P2005-209986)
(22)出願日　平成17年7月20日(2005.7.20)
(65)公開番号　特開2006-72974(P2006-72974A)
(43)公開日　平成18年3月16日(2006.3.16)
審査請求日　平成20年7月16日(2008.7.16)
(31)優先権主張番号　60/606,281
(32)優先日　平成16年9月1日(2004.9.1)
(33)優先権主張国　米国(US)
(31)優先権主張番号　60/606,577
(32)優先日　平成16年9月2日(2004.9.2)
(33)優先権主張国　米国(US)
(31)優先権主張番号　11/061,337
(32)優先日　平成17年2月18日(2005.2.18)
(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　100145388
弁理士　藤原　弘和

最終頁に続く

(54)【発明の名称】デバイスサービスプロバイダインターフェイス

(57)【特許請求の範囲】
【請求項１】
デバイスとの対話を円滑にするシステムであって、
前記システムは、プロセッサと、前記プロセッサと動作するように接続される１以上の物理的コンピュータ読み取り可読記録媒体とを備え、前記コンピュータ読み取り可読記録媒体は、プロセッサにより実行されたときに、以下のコンポーネントを実行するコンピュータ実行可能命令を含み、前記コンポーネントは、
前記プロセッサにより、前記システムの外部にある、関連づけられたプロバイダを有するデバイスに関するプロトコル変換を受信するコンポーネントと、
前記プロセッサにより、前記デバイスと一律に通信するとともに、前記システムに知られていないデバイスベンダおよび前記システムに知られているプロトコル標準に関わらず、前記プロトコル変換に基づき、前記デバイスの機能を公表するためのデバイスサービスプロバイダインターフェイス（ＤＳＰＩ）を定義するデバイスサービスプロバイダインターフェイス（ＤＳＰＩ）コンポーネントであって、前記関連づけられたプロバイダは、デバイス固有の詳細を抽出するとともにプロセッサ非依存のプラットフォームアセンブリであるデバイスコマンドを使用することにより、前記デバイスの機能を公表するための前記デバイスサービスプロバイダインターフェイス（ＤＳＰＩ）を実施する、デバイスサービスプロバイダインターフェイスコンポーネントと、
前記プロセッサにより、前記プロトコル変換に基づき、前記デバイスに関連づけられた、コマンドおよび応答オブジェクトであるメッセージ交換を定義する要求応答コンポーネ

ントと
を備えることを特徴とするシステム。
【請求項２】
前記デバイスベンダは、前記ＤＳＰＩコンポーネントを使用するためのサードパーティのベンダ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３】
前記コマンドおよび前記応答は、非同期であるとともにメッセージＩＤを使用してマッチングされ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】
前記メッセージ交換は、コマンド、プロパティ、および通知の少なくとも１つである、要求と応答のペアであることを特徴とする請求項３に記載のシステム。
【請求項５】
前記デバイスは、無線ＩＣタグ（ＲＦＩＤ）デバイス、リアルタイムのセンサ、センサ、ウェブサービスに拡張可能なデバイス、およびリアルタイムのイベント生成システムの少なくとも１つ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】
前記プロセッサにより、メッセージ層およびトランスポート層の一方である層を定義するデバイスインターフェイスコンポーネントをさらに備え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】
前記プロセッサにより、デバイス接続データをカプセル化する、マークアップ言語によって定義された文字列を使用して、デバイス発見機構を定義するデバイス発見インターフェイスコンポーネントをさらに備え、前記デバイス発見機構は、プロバイダごとに１つのＤＳＰＩコンポーネントをインスタンス化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８】
前記ＤＳＰＩコンポーネントは、前記プロセッサにより、前記プロバイダの構成及び登録を処理するために、前記プロバイダをＲＦＩＤサーバにロードするＳＰＩコンテナコンポーネントをさらに備え、前記ＳＰＩコンテナコンポーネントは、互換性のあるＳＰＩバージョン、ドライバのデジタル認証、およびドライバ登録の１つを提供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９】
前記ＤＳＰＩコンポーネントは、前記プロセッサにより、安全な接続、プロバイダの認証、および前記デバイスの検証の１つを提供するセキュリティコンポーネントを使用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１０】
プロセッサと、前記プロセッサにより実行されたときに、以下の方法を実行する命令を記憶するメモリとを含むコンピュータシステム内で実装される、デバイスとの対話を円滑にする方法であって、前記方法は、
前記プロセッサが、前記システムの外部にある、関連づけられたプロバイダを有するデバイスに関するプロトコル変換を受信するステップと、
前記プロセッサが、前記デバイスと一律に通信するとともに、前記システムに知られていないデバイスベンダおよび前記システムに知られているプロトコル標準に関わらず、前記プロトコル変換に基づき、前記デバイスの機能を公表するためのデバイスサービスプロバイダインターフェイス（ＤＳＰＩ）を定義するステップであって、前記関連づけられたプロバイダは、デバイス固有の詳細を抽出するとともにプロセッサ非依存のプラットフォームアセンブリであるデバイスコマンドを使用することにより、前記デバイスの機能を公表するための前記デバイスサービスプロバイダインターフェイス（ＤＳＰＩ）を実施する、定義するステップと
を含むことを特徴とする方法。

【請求項１１】
プロセッサにより実行されたときに、請求項１０に記載の方法を実行するコンピュータ実行可能命令を格納していることを特徴とするコンピュータ可読記録媒体。
【請求項１２】
前記プロセッサが、前記プロバイダの構成及び登録を処理するために、ＲＦＩＤサーバにプロバイダをロードするステップと、
前記プロセッサが、メッセージ層およびトランスポート層を定義するステップと、
前記プロセッサが、１つのプロバイダに対して１つのコンポーネントをインスタンス化し、かつ、デバイスを使用することにより、前記デバイスを発見し構成するステップと
をさらに含むことを特徴とする請求項１０に記載の方法。
【請求項１３】
前記プロセッサが、非同期であるメッセージ交換のペアを実施することをさらに含むことを特徴とする請求項１０に記載の方法。
【請求項１４】
前記メッセージ交換のペアは、要求および応答、要求および返信、要求および通知、ならびに要求およびプロパティの１つであることを特徴とする請求項１３に記載の方法。
【請求項１５】
前記デバイスは、無線ＩＣタグ（ＲＦＩＤ）デバイス、リアルタイムのセンサ、センサ、ウェブサービスに拡張可能なデバイス、およびリアルタイムのイベント生成システムの少なくとも１つであることを特徴とする請求項１０に記載の方法。
【請求項１６】
プロセッサにより実行されたときに、無線ＩＣタグ（ＲＦＩＤ）デバイスとの対話を円滑にする方法を実行するコンピュータ実行可能命令を記憶するコンピュータ可読記録媒体であって、前記方法は、
前記プロセッサが、無線ＩＣタグ（ＲＦＩＤ）デバイスに関するプロトコル変換を受信するステップと、
前記プロセッサが、前記ＲＦＩＤデバイスと一律に通信するとともに、前記システムに知られていないデバイスベンダおよび前記システムに知られているプロトコル標準に関わらず、前記プロトコル変換に基づき、前記ＲＦＩＤデバイスの機能を公表するためのデバイスサービスプロバイダインターフェイス（ＤＳＰＩ）を定義するステップと、
前記プロセッサが、前記プロバイダの構成及び登録を処理するために、ＲＦＩＤサーバにＲＦＩＤプロバイダをロードするステップと、
前記プロセッサが、前記ＲＦＩＤデバイスから前記ＤＳＰＩへの通信機能を提供する、メッセージ層およびトランスポート層を定義するステップと、
前記プロセッサが、１つのプロバイダに対して１つのコンポーネントをインスタンス化し、かつ、デバイスを使用することにより、前記デバイスを発見し構成するステップと
を備え、
前記ＲＦＩＤデバイスは、前記システムの外部にある前記ＲＦＩＤプロバイダに関連づけられ、前記関連づけられたＲＦＩＤプロバイダは、デバイス固有の詳細を抽出するとともにプロセッサ非依存のプラットフォームアセンブリであるデバイスコマンドを使用することにより、前記ＲＦＩＤデバイスの機能を公表するための前記ＤＳＰＩを実施することを特徴とするコンピュータ可読記録媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は概して、無線ＩＣタグ（ＲＦＩＤ）に関し、より詳細には、一律なＲＦＩＤ通信および管理の提供を円滑にするシステムおよび／または方法に関する。
【背景技術】
【０００２】
多くの小売、製造、および流通施設は、効率を上げるために様々な革新的運用方法を適

用している。こうした施設は、消費者に関連する需要と供給の最適化を促進するように、店舗の在庫を監視することができる。利益を最大限にするための一側面は、商品および／または製品の消費に伴って補充が行われるような、在庫の適正な仕入れによって決まる。たとえば、コンピュータおよび／またはＶＣＲを販売する小売業者は、コンピュータの消費者売上げに対応してコンピュータを仕入れ、ＶＣＲの消費者売上げに対応してＶＣＲを仕入れなければならない。したがって、コンピュータの需要がＶＣＲより高い（たとえば、より多くのユニットが売れている）場合、小売業者は、需要と供給、最終的には利益を最適化するために、コンピュータをより頻繁に仕入れることができる。在庫および関連する売上げの監視は複雑な作業である場合があり、製品関連活動は、内部の仕組みがわからないのでブラックボックスに相当するが、製品の監視は、在庫／製品の効率において不可欠な要素である。

【０００３】

製品に関連する監視システムの一タイプは、ポータブルな画像収集デバイス（たとえば、バーコードリーダ）であり、製造、サービス、および／または宅配産業において広く用いられている。このようなデバイスは、様々なオンサイトデータ収集活動を実施することができる。ポータブルなデータ収集デバイスはしばしば、在庫管理、追跡、生産管理および促進、品質保証、ならびに／あるいは他の目的のために、卸売り店、小売店、出荷ターミナルにある製品、製品の包装、および／またはコンテナに添付されたバーコードデータフォームを読み取るように適合された一体型のバーコードデータフォームリーダを含む。

【０００４】

製品に一意のバーコードをつけることができ、バーコードは、その製品に関する情報に関連づけることができる。製品のバーコードをスキャンするのにバーコードスキャナを使用することができ、バーコードから製品関連情報を取得することができる。しかし、このような識別用の情報は、製品の外見を損なうので、見た目が悪い。さらに、バーコードの破損、汚れ、注釈、または他の物理的損傷／改変により、このような従来のシステムおよび／または方法がほとんど役に立たなくなる場合がある。バーコードの一部分が、製品から破り取られている場合、バーコードスキャナは、バーコードを正しく読み取ることができなくなり得る。同様に、製品の汚れにより、このようなバーコードが読取り不可能になり得る。

【０００５】

バーコードリーダおよび統一商品コード（ＵＰＣ）を使用する監視システムおよび／または方法は、ユーザ（たとえば、小売業者、卸売業者、製造者など）をさらに面倒な問題に直面させる。バーコードリーダは、製品を正しく監視するために、障害物のない視界を必要とする。たとえば、一般的なバーコードシステムでは、正しい読取りを行うために、バーコードおよび／またはＵＰＣから４～８インチ（１０．１６～２０．３２ｃｍ）以内にスキャナがある必要がある。バーコードシステムにこうした視界が必要なだけでなく、個々の各製品に対する手動スキャンも、製品を識別するために必要である。さらに、単一のバーコードおよび／またはＵＰＣは、ある製品のすべての事例を表さなければならない（たとえば、Ｔｏｍａｔｏブランドの１本のケチャップ瓶が、その製品を表す単一のＵＰＣおよび／またはバーコードとされる）。さらに、単一のバーコードおよび／またはＵＰＣに関連づけられる情報の量は制限されている。したがって、Ｔｏｍａｔｏブランドのケチャップをスキャンすることにより、製品の識別および値段を得ることができる。この情報には実体がないだけでなく、この情報によって、リアルタイムの製品監視を行うことができない。

【０００６】

自動識別およびデータ収集（ＡＩＤＣ）技術、特に無線ＩＣタグ（ＲＦＩＤ）は、監視システムおよび／または方法（たとえば、バーコードリーダ、バーコード、および／またはＵＰＣ）の上記の欠陥を取り除く必要性に少なくとも基づいて開発された。ＲＦＩＤは、ＲＦＩＤタグを使用してデータをリモートに格納し取得する技術である。ＲＦＩＤシステムは、無線周波数および関連づけられた信号に基づくので、製品の監視において、多数

の利益および／または利点が従来技術より勝っている。ＲＦＩＤ技術は、製品を監視し、かつ／またはＲＦＩＤタグから信号を受信するために、障害物のない視界を必要としない。したがって、手動スキャンは必要なく、スキャナが、目標物（たとえば、製品）に接近している必要がある。そうであっても、ＲＦＩＤにおいて、範囲は、無線周波数、ＲＦＩＤタグサイズ、および関連づけられた電力源に基づいて限られる。さらに、ＲＦＩＤシステムは、数秒以内で多くの読取りを行わせ、高速スキャンおよび識別を可能にする。言い換えると、ＲＦＩＤシステムは、タグがＲＦＩＤリーダの範囲内にあるとき、複数のタグを読み取らせ、かつ／または識別させる。ＲＦＩＤシステムで多くの読取りを行う機能は、個々の各製品の一意の識別コードを含む情報タグを提供できるようにすると完全なものとなる。したがって、バーコードシステムとは対照的に、Ｔｏｍａｔｏブランド製のケチャップの各瓶が、関連づけられた１つの識別コードをもつことになる。たとえば、Ｔｏｍａｔｏブランド製の２本のケチャップの瓶は、ＲＦＩＤシステムにおいて、それぞれに関連づけられた別個の２つの識別コードをもつ。一方、バーコードシステムでは、Ｔｏｍａｔｏブランド製の２本のケチャップの瓶は、同じバーコードおよび／またはＵＰＣをもつ。別の例では、ＲＦＩＤシステムおよび／または方法は、水中パイプの追跡および／または監視などのように、水中で実装することができるが、バーコード監視システムは、このような条件の下では、多数の問題点を提示する。

【０００７】

さらに、ＲＦＩＤシステムおよび／または方法は、タグつき項目に関連づけられたリアルタイムのデータを提供する。リアルタイムのデータストリームにより、小売業者、卸売業者、および／または製造者は、在庫および／または製品を正確に監視することができる。ＲＦＩＤの使用により、フロントエンド（ｆｒｏｎｔ－ｅｎｄ）での流通（たとえば、小売業者から消費者）およびバックエンド（ｂａｃｋ－ｅｎｄ）での流通（たとえば、卸売業者／製造者から小売業者）での製品の供給がさらに円滑になり得る。卸売業者および／または製造者は、商品の出荷、品質、量、出荷時間などを監視することができる。さらに、小売業者は、受け取った在庫の量、このような在庫のある場所、品質、在庫期間などを追跡することができる。上述した利益は、たとえば、フロントエンドでの供給、バックエンドでの供給、流通チェーン、製造、小売、自動化など、多数の領域に渡って機能するＲＦＩＤ技術の柔軟性を示している。

【０００８】

ＲＦＩＤシステムは、少なくともＲＦＩＤタグおよびＲＦＩＤトランシーバからなる。ＲＦＩＤタグは、ＲＦＩＤトランシーバからの無線周波数クエリとの間で受信および／または送信を行うアンテナを含み得る。ＲＦＩＤタグは、たとえば、粘着ステッカー、弾力性のあるラベル、および集積チップなど、小型の物体でよい。一般に、ＲＦＩＤタグが使用する４通りの周波数、すなわち低周波数タグ（１２５から１３４キロヘルツの間）、高周波数タグ（１３．５６メガヘルツ）、ＵＨＦタグ（８６８から９５６メガヘルツ）、およびマイクロ波タグ（２．４５ギガヘルツ）がある。

【０００９】

様々な周波数範囲内において、ＲＦＩＤタグは、受動でも、能動でもよい。受動ＲＦＩＤタグは、電源を含まない。ＲＦＩＤトランシーバからの受信無線周波数によって、アンテナ中に電流が誘導されると、タグが応答するのに十分な電力が与えられる。多くの例において、受動ＲＦＩＤタグの応答は短く、ＩＤ番号（たとえば、グローバル一意識別子（ＧＵＩＤ））からなる。ＧＵＩＤとは、一意であり、標準汎用一意識別子（ＵＵＩＤ）（たとえば、１６進形式で書かれた１６バイトの番号）によって与えることができる擬似乱数である。しかし、ＲＦＩＤシステムおよび／または方法は、たとえば、電子製品コード（ＥＰＣ）と呼ばれるマルチビット形式（たとえば、６４ビットまたは９６ビット）での情報格納にのみ注意を向けていた。受動ＲＦＩＤタグ内に電源がないおかげで、デバイスは小型であり、コスト効率が優れている。一部の受動ＲＦＩＤタグは、０．４ｍｍ×０．４ｍｍのサイズであり、厚さは１枚の紙より薄い。それでも、電源がないせいで、受動ＲＦＩＤタグの実際の読取り範囲は、１０ｍｍから約５メートルに限定される。

【００１０】
能動ＲＦＩＤタグは、より長い読取り範囲をもたらす電力源を含む。一般的な能動ＲＦＩＤタグは、米国の硬貨くらいの大きさであり、約数十メートルの読取り範囲をもたらし、最大数年間のバッテリ寿命を維持する。さらに、能動ＲＦＩＤタグには、読取りおよび／または書込みを行うことができる。たとえば、ＲＦＩＤタグは、能動ＲＦＩＤタグに書込みを行うことによって、盗難を防ぐためのセキュリティ層をさらに提供し得る。セキュリティビットは、少なくともＲＦＩＤトランシーバに基づいて、セキュリティ状況を判定することができる。あるセキュリティシステムでは、たとえば、能動ＲＦＩＤタグは、１にセットされ／書かれたセキュリティビットをもつ場合があり、このビットは、製品が、アラーム／警告をトリガせずに安全な区域を離れることを許可されていないことを示し得る。適切な条件が存在すると、ＲＦＩＤシステムおよび／または方法は、タグのビットをゼロと書く場合があり、このビットは、タグつき製品が、安全な区域を離れることを許可されることを示し得る。
【００１１】
概して、ＲＦＩＤシステムは、多数のコンポーネント、すなわちタグ、タグリーダ（たとえば、タグトランシーバ）、タグプログラミングステーション、循環リーダ、ソート機器、タグ在庫ワンドなどを含み得る。さらに、様々なメイク（ｍａｋｅ）、モデル、タイプ、および／またはアプリケーションを、それぞれのコンポーネント（たとえば、タグ、タグリーダ、タグプログラミングステーション、循環リーダ、ソート機器、タグ在庫ワンドなど）に関連づけることができ、こうした関連づけが、対象となっているＲＦＩＤシステムおよび他のＲＦＩＤシステムにおける発見、構成、セットアップ、通信、保守、セキュリティ、および／または互換性を複雑にさせ得る。
【発明の開示】
【発明が解決しようとする課題】
【００１２】
上記のことを鑑みて、メーカ（ｍａｋｅｒ）および関連づけられた仕様に関して、ＲＦＩＤデバイスを発見し、構成し、セットアップし、通信するための一律な方法を提供する必要がある。
【課題を解決するための手段】
【００１３】
以下では、本発明のいくつかの態様の基本的な理解をもたらすために、本発明の簡略な要約を提示する。この要約は、本発明の包括的な概要ではない。本発明の主要な／重大な要素を明らかにすることも、本発明の範囲を詳述することも意図していない。後で提示するより詳細な説明の前置きとして、本発明のいくつかの概念を簡略な形で提示することだけを目的としている。
【００１４】
本発明は、無線ＩＣタグ（ＲＦＩＤ）デバイスおよび／または他のリアルタイムのイベント生成システム（たとえば、センサ、ウェブサービスなど）との対話を円滑にするシステムおよび／または方法に関する。デバイスサービスプロバイダインターフェイス（ＤＳＰＩ）コンポーネントは、デバイスのタイプおよび／またはブランドに関わらず、デバイス（１つまたは複数）と一律に通信し管理するための抽象化層を提供することができる。具体的には、ＤＳＰＩコンポーネントは、デバイスのベンダ（たとえば、リーダのベンダ）がＲＦＩＤサーバに一律にサービスを提供するための少なくとも１つのインターフェイスを定義する。したがって、ＤＳＰＩコンポーネントは、サーバと少なくとも１つのデバイスとの間の層である。さらに、ＤＳＰＩコンポーネントは、発見、構成、通信、および接続管理を処理するためのインターフェイスを定義することができる。
【００１５】
本発明の一態様によると、ＤＳＰＩコンポーネントは、デバイスに関するプロトコル変換、ＲＦＩＤサーバデータ、ＲＦＩＤデバイスデータなどの１つまたは複数を受信する受信機コンポーネントを含み得る。ＤＳＰＩコンポーネントは、デバイスとの一律な通信、

ならびにデバイスベンダおよびプロトコル標準に関わらない、デバイスの機能の公開を円滑にする。さらに、受信機コンポーネントは、ＤＳＰＩコンポーネントに対して、外部および／または内部のいずれかでよい。デバイスは、ＲＦＩＤリーダ、ＲＦＩＤライタ、ＲＦＩＤ送信機、ＲＦＩＤデバイス、センサ、リアルタイムの生成システム、リアルタイムのセンサ、ウェブサービスに拡張可能なデバイス、およびリアルタイムのイベント生成システムなどでよいが、それに限定されないことを理解されたい。

【００１６】

本発明の別の態様によると、ＤＳＰＩコンポーネントは、デバイス層でのメッセージ交換を定義する要求応答コンポーネントを含み得る。メッセージ交換は非同期でよいことを理解されたい。さらに、メッセージ交換は、要求と応答のペア、通知、コマンド、および／またはプロパティでよい。メッセージ交換は、要求応答コンポーネントによって使用され、メッセージ交換は、ＲＦＩＤサーバと少なくとも１つのＲＦＩＤデバイスとの間のマークアップ言語シンタクスである。マークアップ言語は、ＸＭＬ（拡張マークアップ言語）、ＨＴＭＬ（ハイパーテキストマークアップ言語）、ＳＧＭＬ（汎用マークアップ言語）、およびＸＨＴＭＬ（拡張可能ハイパーテキストマークアップ言語）でよいが、それに限定されない。

【００１７】

さらに、ＤＳＰＩコンポーネントは、メッセージ層および／またはトランスポート層を定義するデバイスインターフェイスコンポーネントを含み得る。デバイスインターフェイスコンポーネントは、ＲＦＩＤサーバと少なくとも１つのＲＦＩＤデバイスとの間のメッセージ通信および接続の管理を可能にする。デバイスインターフェイスコンポーネントは、メッセージ（たとえば、通知、応答、要求など）を送信し、かつ／または受信するために、マークアップ言語シンタクスを使用することができる。

【００１８】

本発明のさらに別の態様において、ＤＳＰＩコンポーネントは、ＲＦＩＤデバイスを発見するデバイス発見インターフェイスコンポーネントを含み得る。したがって、デバイス発見インターフェイスコンポーネントは、新規ＲＦＩＤデバイス（１つまたは複数）について通知し、かつ／または報告することができるインターフェイスを定義することができる。効率的であるために、デバイス発見インターフェイスコンポーネントは、プロバイダごとに１つのコンポーネントをインスタンス化することができる。さらに、ＤＳＰＩコンポーネントは、ＲＦＩＤサーバへのＤＳＰＩプロバイダのロードを円滑にするＳＰＩコンテナコンポーネントを含み得る。ＳＰＩコンテナコンポーネントは、ＲＦＩＤサーバとＳＰＩの間のバージョン管理機能を維持する。さらに、ＳＰＩコンテナコンポーネントは、ＲＦＩＤサーバに少なくとも１つのドライバを登録することができる。

【００１９】

以下の説明および添付の図面では、本発明の特定の例示的な態様を詳細に説明する。ただし、こうした態様は、本発明の原理を利用することができる様々な方法のごくわずかを示すに過ぎず、本発明は、このような態様およびその等価物すべてを含むことを意図している。本発明の他の目的、利点および新規な特徴は、本発明の以下の詳細な説明を図面と併せ読むことにより、明らかになるであろう。

【発明を実施するための最良の形態】

【００２０】

本出願において使用する「コンポーネント」、「システム」などの用語は、コンピュータ関連のエンティティ、すなわちハードウェア、（たとえば実行中の）ソフトウェア、および／またはファームウェアのいずれかを指すことを意図している。たとえば、コンポーネントは、プロセッサ上で実行中のプロセス、プロセッサ、オブジェクト、実行ファイル、プログラム、および／またはコンピュータでよい。例として、サーバ上で実行されているアプリケーションおよびそのサーバが両方とも、コンポーネントとなり得る。１つのプロセス中に１つまたは複数のコンポーネントが存在してよく、コンポーネントは、１台のコンピュータに配置することも、かつ／または２台以上のコンピュータの間に分散するこ

ともできる。

【００２１】

図面を参照して本発明を説明するが、同じ参照番号は、全体を通して同じ要素を指すのに使用している。以下の記述では、説明の目的で、具体的な多くの詳細を、本発明の完全な理解をもたらすために述べる。ただし、こうした具体的な詳細なしでも本発明を実施できることが明らかであろう。他の例では、本発明を説明しやすくするために、公知の構造およびデバイスをブロック図の形で示す。

【００２２】

人工知能ベースのシステム（たとえば、明示的におよび／または暗黙的にトレーニングされた分類器）を、本明細書において述べる推論ならびに／あるいは確率論的決定および／または統計に基づく決定の実施に関連して利用することができる。本明細書で使用する「推論」という用語は、イベントおよび／またはデータを介して捕捉された１組の観察結果から、システム、環境、および／またはユーザの状態を類推しまたは推論するプロセスを指す。推論は、たとえば、特殊なコンテキストまたはアクションを識別するのに利用することもでき、複数の状態に渡る確率分布を生成することもできる。推論は、確率論的でよい。つまり、データおよびイベントの考慮に基づく、興味のある状態に渡る確率分布の計算でよい。推論は、１組のイベントおよび／またはデータから、より高いレベルのイベントを組み立てるのに利用される技術も指し得る。このような推論の結果、イベントが、時間的に近接して相関づけられるかどうか、イベントおよびデータが、１つのイベントおよびデータソースからのものであるか、それともいくつかのソースからのものであるかに関わらず、観察された１組のイベントおよび／または格納されているイベントデータから、新しいイベントまたはアクションが構成される。様々な分類方式および／またはシステム（たとえば、サポートベクトルマシン、ニューラルネットワーク、エキスパートシステム、ベイズの信念ネットワーク、ファジー理論、データ融合エンジンなど）は、本発明とともに、自動的および／または推論によるアクションの実施に関連して利用することができる。

【００２３】

ここで図面に移ると、図１は、一律な通信および／または管理をもたらすために、デバイスコンポーネントとの対話を円滑にするシステム１００を示す。システム１００は、デバイスインターフェイスコンポーネント１０２を利用して、デバイスコンポーネント１０４と一律に通信し管理する抽象化層を提供することができる。デバイスインターフェイスコンポーネント１０２は、サーバ１０６と少なくとも１つのデバイスコンポーネント１０４との間の「中間媒介」として作用し得る。サーバ１０６は、たとえばＲＦＩＤサーバでよく、少なくとも１つのサービス（たとえば、公表、加入、クエリ、ポーリング、管理、監視、アップデートなど）および／またはプログラミングされた（たとえば、製造および／または出荷などに関する）コンピュータプロセスを、デバイスコンポーネント１０４に提供することができる。さらに、デバイスインターフェイスコンポーネント１０２は、サーバデータ、デバイスに関するプロトコル変換、ＲＦＩＤサーバデータ、ＲＦＩＤデバイスデータなどの１つまたは複数を受信する受信機コンポーネント１０８を含み得る。デバイスインターフェイスコンポーネント１０２は、デバイスとの一律な通信、ならびにデバイスベンダおよびプロトコル標準に関わらない、デバイスの機能の公開を円滑にすることを理解されたい。さらに、受信機コンポーネント１０８は、デバイスインターフェイスコンポーネント１０２に対して、外部および／または内部のいずれかでよい。デバイスは、ＲＦＩＤリーダ、ＲＦＩＤライタ、ＲＦＩＤ送信機、ＲＦＩＤデバイス、デバイス、リアルタイムのセンサ、センサ、ウェブサービスに拡張可能なデバイス、およびリアルタイムのイベント生成システムなどでよいが、それに限定されないことをさらに理解されたい。

【００２４】

デバイスベンダ（たとえば、ある特定のタイプのハードウェアデバイスおよび関連づけられたソフトウェアドライバのハードウェア製造を専門にする独立系ハードウェアベンダ（ＩＨＶ））は、多量のデバイスおよび関連づけられたコマンドセットに少なくとも部分

的に基づいて、ミドルウェア製品にサービスを提供するために、デバイスインターフェイスコンポーネント１０２を使用することができる。言い換えると、あるベンダからの１つのデバイスが１組のコマンドを使用することができ、別のデバイスは、実質的に異なる１組のコマンドを使用することができる。様々なベンダおよびコマンドセットに関連する問題を緩和するために、デバイスインターフェイスコンポーネント１０２は、サーバ１０６に一律にサービスを提供する、デバイスベンダ用のインターフェイスを定義することができる。したがって、サーバ１０６は、デバイスインターフェイスコンポーネント１０２を使用して、デバイスに柔軟性を与える一律な技術を用いてデバイスコンポーネント１０４と対話することができる。さらに、デバイスインターフェイスコンポーネント１０２は、デバイスコンポーネント１０４の発見、構成、通信、および接続管理を委譲するための１つおよび／または複数のインターフェイスを提供することができる。デバイスインターフェイスコンポーネント１０２は、多数のデバイスコンポーネント１０４と対話することができ、プロバイダ（１つまたは複数）は、各デバイスコンポーネント１０４に関連づけられることを理解されたい。

【００２５】

たとえば、様々なレガシーシステム（たとえば、レガシーシステムが一般に、新しい標準という面において旧式である、通信用の固有プロトコルをもつリーダおよび／またはデバイスである）は、その名の通り、旧式のデバイスおよび／またはソフトウェアを使用する。こうしたレガシーシステムは、デバイスインターフェイスコンポーネント１０２を利用することができ、多数のレガシーデバイスおよびプロバイダにサーバ１０６との一律な対話を提供するための１つおよび／または複数のインターフェイスを可能にする。したがって、旧式のデバイスコンポーネント１０４および関連づけられたプロバイダ（図示せず）は、デバイスインターフェイスコンポーネント１０２およびサーバ１０６を介して様々なサービスおよび／またはプロセスを提供するのに使用することができる。

【００２６】

一例では、デバイスインターフェイスコンポーネント１０２は、複数のデバイスコンポーネント１０４とサーバ１０６の間の対話向けの一律な技術を提供するために使用することができる。デバイスコンポーネント１０４は、たとえば、レガシーデバイス、自動識別デバイス、ＥＰＣ－Ｇｌｏｂａｌ準拠デバイスなどでよい。さらに、各デバイスコンポーネント１０４は、関連づけられたプロバイダ（たとえば、ＥＰＣ－Ｇ準拠プロバイダ、固有プロバイダ、レガシープロバイダなど）を有し得ることを理解されたい。したがって、デバイスインターフェイスコンポーネント１０２は、デバイスコンポーネント１０４およびサーバ１０６用の様々なコマンドセットを使用して、多数のプロバイダの間に一律な対話をもたらすことができる。

【００２７】

デバイスインターフェイスコンポーネント１０２は、多数のデバイスコンポーネント１０４、ウェブサービス、および／またはリアルタイムイベント生成システム（図示せず）を代表する多数の標準に渡る正規化を可能にし得る。したがって、デバイスコンポーネント１０４は、ＲＦＩＤデバイスおよび／またはセンサデバイスでよい。さらに、デバイスインターフェイスコンポーネント１０２を使用することにより、ハードウェアの革新をより高いレベルに表面化させることが可能になる。デバイスインターフェイスコンポーネント１０２は、たとえば、プロセッサ非依存のプラットフォームアセンブリでよいプロバイダ（図示せず）によって実装することができる。プロバイダは、デバイスインターフェイスコンポーネント１０２によって定義された１つおよび／または複数のインターフェイスを実装することができる。プロバイダ用のインターフェイスを定義することによって、デバイスインターフェイスコンポーネント１０２は、デバイス固有コマンドを利用することによってデバイスコンポーネント１０４と対話する。したがって、デバイスインターフェイスコンポーネント１０２は、ＲＦＩＤサーバ１０６からのデバイス固有の詳細を最小限に評価する。

【００２８】

図２は、本発明の新規性に関する概観を提示するシステム２００を示す。たとえば、ＲＦＩＤデバイスベンダは、そのサービスを、オペレーティングシステムプラットフォーム上で実行されるホスト層に提供する一律な方法を必要とする。というのは、各デバイスが、異なるコマンドセット、プロトコル（１つまたは複数）、および／または振舞いをサポートするからである。デバイスサービスプロバイダインターフェイス（ＤＳＰＩ）は、ＲＦＩＤサービスプラットフォームが一律にＲＦＩＤデバイスと通信しデバイスを管理するための抽象化層である。この層は、多数の通信プロトコルに渡る正規化、レガシーリーダおよび他の自動ＩＤデバイスの一律なサポート、ならびに主要なハードウェア革新を上位層に表面化させる機能を提供する。

【００２９】

ＤＳＰＩは、デバイスベンダが一律な方法でオペレーティングシステム上のＲＦＩＤサービスプラットフォームにサービスを提供するように実装することができる、（発見、構成、通信、ならびにデバイスおよび接続管理の処理用の）抽象クラスを定義する。プロバイダは、ＲＦＩＤサービスホスト２０２の下で、マネージドエンティティとして実行されることができ、ホストマシン２０６と呼ばれる、サポートされるトランスポートに基づいて、たとえば、マネージド（ｍａｎａｇｅｄ）ＡＰＩ、アンマネージド（ｕｎｍａｎａｇｅｄ）コード、ＣＯＭ実装、またはＷｉｎ３２ＡＰＩを介して、１台のデバイス２１６および／または複数のデバイス２１６自体と通信することができる。

【００３０】

具体的には、プロセスインスタンスａｐｐドメイン２０８は、プロセスインスタンス２１０を含むことができ、プロセスインスタンスａｐｐドメイン２０８は、少なくとも１つのデバイスプロバイダａｐｐドメイン２１２と対話することができる。デバイスプロバイダａｐｐドメイン２１２は、デバイス用のＤＳＰＩ実装２１４を含むことができ、ＤＳＰＩ実装２１４に関連づけられた複数のデバイス２１６が存在し得る。１からＮ個のデバイスが存在することができ、Ｎは、図に示すように整数であることを理解されたい。

【００３１】

デバイスサービスプロバイダは、以下のカテゴリの少なくとも１つに該当する。すなわち、１）標準プロトコル［ＨＴＴＰ、ＴＣＰ、ＳＥＲＩＡＬ（たとえば、Ｓａｍｓｙｓ）］の１つを用いた純粋なマネージドコードの実装、２）マネージドコードプロトコルの実装ではあるが、アンマネージドコードをコールする（たとえばＵＳＢリーダデバイス、異なるプロトコルハンドラプロセスにつながるプロキシ）、３）ウィンドウズ（登録商標）デバイスドライバ（たとえば、ＳＡＴＯなど、純粋なプリンタドライバ）として実装された固有プロトコル用のマネージドコードラッパーである。

【００３２】

図３は、通信および／または管理のための一律な技術を提供するように、デバイスとの対話を円滑にするシステム３００を示す。ＲＦＩＤサーバ３０６は、デバイスサービスプロバイダインターフェイス（ＤＳＰＩ）コンポーネント３０２を使用することによって、ＲＦＩＤデバイス３０４と通信することができる。ＲＦＩＤサーバ３０６は、センサ（図示せず）ならびにどのＲＦＩＤデバイス３０４とも通信できることを理解されたい。ＲＦＩＤデバイスプロバイダ（図示せず）は、ＤＳＰＩコンポーネント３０２を利用することができ、ＲＦＩＤサーバ３０６との対話を円滑にする少なくとも１つのＤＳＰＩコンポーネントを定義することができる。ＤＳＰＩコンポーネント３０２によって定義されたインターフェイスは、ＤＳＰＩコンポーネント３０２にデバイス機能を公表するためのプロトコル変換用に、ＩＨＶによって実装することができる。ＤＳＰＩコンポーネント３０２は、ＲＦＩＤサーバおよび複数のＲＦＩＤデバイス３０４との一律な対話を可能にすることを理解されたい。ＲＦＩＤデバイス３０４は、ＲＦＩＤリーダ、ＲＦＩＤライタ、ＲＦＩＤ送信機などでよいが、それに限定されないことを理解されたい。

【００３３】

ＤＳＰＩコンポーネント３０２は、ＲＦＩＤデバイス３０４とのメッセージの処理を円滑にする要求応答コンポーネント３０８をさらに含み得る。要求応答コンポーネント３０

８は、ＲＦＩＤサーバ３０６とＲＦＩＤデバイス（１つまたは複数）３０４との間のメッセージ交換を定義する。メッセージ交換は、たとえば、メッセージのペア（たとえば、ベンダ固有コマンドを使用して、第１のメッセージが第２のメッセージをトリガし得る）、要求、返信、通知、イベント、プロパティ、クエリ、肯定応答、フラグなどでよい。要求応答コンポーネント３０８は、非同期および／または同期である少なくとも１つのメッセージ交換を定義することを理解されたい。したがって、要求応答コンポーネント３０８からの非同期であるメッセージ交換は、要求からの即時応答をトリガしない。

【００３４】

たとえば、レガシーデバイスおよび／または固有プロバイダは、一律な通信技術を使用するために、ＤＳＰＩコンポーネント３０２を実装することができる。ＤＳＰＩコンポーネントを実装するプロバイダは、たとえば、プロセッサ非依存プラットフォームアセンブリでよいことを理解されたい。ＤＳＰＩコンポーネント３０２は、アセンブリがＲＦＩＤデバイス３０４に関連して実装を開始するためのインターフェイスを定義する。言い換えると、ＤＳＰＩコンポーネント３０２は、ＲＦＩＤサーバ３０６にＩＨＶサービスを提供するために複数のデバイスおよび／または複数のプロバイダが使用することができる一律なコマンドセットを利用する。

【００３５】

さらに、ＤＳＰＩコンポーネント３０２は、デバイスインターフェイスコンポーネント３１０を含み得る。デバイスインターフェイスコンポーネント３１０は、メッセージ層および／またはトランスポート層を定義する。メッセージ層および／またはトランスポート層は、ＸＭＬ（拡張マークアップ言語）、ＨＴＭＬ（ハイパーテキストマークアップ言語）、ＳＧＭＬ（汎用マークアップ言語）、およびＸＨＴＭＬ（拡張可能ハイパーテキストマークアップ言語）などであるがそれに限定されないマークアップ言語によって実装することができる。トランスポート層は、メッセージ層から独立し得ることを理解されたい。さらに、デバイスインターフェイスコンポーネント３１０は、メッセージおよび接続管理を委譲する。ＲＦＩＤサーバ３０６との通信および／または対話が持続するために、デバイスインターフェイスコンポーネント３１０は、メッセージおよびトランスポート層を定義することができる。要求応答コンポーネント３０８とともに、デバイスインターフェイスコンポーネント３１０は、（たとえば、要求応答コンポーネント３０８によって定義された）メッセージのペアおよび（たとえば、デバイスインターフェイスコンポーネント３１０によって定義された）メッセージ／トランスポート層を使用して、ＲＦＩＤデバイス３０４およびＲＦＩＤサーバ３０６とのメッセージ通信を管理する。さらに、各デバイスの接続は、デバイスインターフェイスコンポーネント３１０によって、メッセージ／トランスポート層を介して管理される（たとえば、制御され、確立され、決定され、放棄され、監視される、など）。

【００３６】

ＲＦＩＤデバイス３０４の発見を円滑にするために、ＤＳＰＩコンポーネント３０２は、デバイス発見インターフェイスコンポーネント３１２を含み得る。言い換えると、デバイス発見インターフェイスコンポーネント３１２は、デバイス発見機構（１つまたは複数）を定義する。このような発見機構は、発見開始、発見中止、接続要件データ（たとえば、デバイスｉｄ、プロバイダ名など）などでよいが、それに限定されない。デバイス発見インターフェイスコンポーネント３１２は、プロバイダごとに１つのＤＳＰＩコンポーネントを効率的にインスタンス化し、このようなベンダに関連づけられたデバイスは、処理される（たとえば、制御され、管理され、監視される、など）ことを理解されたい。さらに、プロバイダは、デバイス発見インターフェイスコンポーネント３１２を実装し得ることを理解されたい。

【００３７】

ＤＳＰＩコンポーネント３０２は、ＲＦＩＤサーバ３０６にプロバイダ（図示せず）をロードするＳＰＩコンテナコンポーネント３１４をさらに含み得る。ＲＦＩＤサーバ３０６にプロバイダをロードすることによって、プロバイダの構成および登録が、ＳＰＩコン

テナコンポーネント３１４によって処理される。ＳＰＩコンテナコンポーネント３１４は、プロバイダに関するバージョンおよび識別情報を提供する。さらに、ＳＰＩコンテナコンポーネント３１４は、プロバイダの実装に対するスーパーゲートウェイである。ＳＰＩプロバイダ（１つまたは複数）は、たとえば、プロセッサ非依存プラットフォームアセンブリでよい。プロバイダ（１つまたは複数）は、複数のドライバを交換して使用できることを理解されたい。

【００３８】

たとえば、プロバイダは、調整されたソリューションを提供するために、設計、またはプラットフォーム、またはスマートデバイスによって、情報および機能を互いに共有し、かつ／または組み合わせることができるサービスとして、マークアップ言語アプリケーション、プロセス、およびウェブサイトの作成および／または使用を可能にするプロセッサ非依存ソフトウェアアセンブリでよい。このようなアセンブリを使用することによって、様々な利益および／または利点がもたらされる。ドライバの異なるバージョンがサーバ上に同時に存在し得るので、ドライバのバージョン管理の問題は、このようなアセンブリのバージョン管理を使用して解決することができる。したがって、あるバージョンから別のバージョンに変わるとき、アセンブリは、多数のバージョンを使用できるようにし、正しいバージョンが入手可能である。このようなアセンブリ形式で書かれるドライバは、バッファオーバーラン、エラーなどの影響を受けにくい。ドライバからの例外は、サーバの安定性が影響を受けないように分離させることができる。さらに、ＩＨＶは、ドライバにデジタル署名して、認証および／または正確さを保証することができる。

【００３９】

図４は、デバイスとの対話を円滑にして、通信および管理のための一律な技術を提供するシステム４００を示す。要求応答コンポーネント３０８は、デバイス（図示せず）およびサーバ（図示せず）が通信することを可能にするメッセージ交換のペアを提供するペアコンポーネント４０２を含み得る。たとえば、メッセージ交換は、非同期である要求および応答のペアを定義することができる。さらに、メッセージ交換は、識別を使用してマッチングされる（たとえば、対にされる）。識別は、たとえば、対応する具体的なペアに一意のメッセージ識別でよい。ペアコンポーネント４０２は、ベンダ固有コマンド（たとえば、メッセージ交換のペア、要求と応答のペアなど）をサポートできることを理解されたい。さらに、コマンド／ルーチン（たとえば、後で論じる「SendMessage()」）は、デバイスに要求を送信するのに使用することができる。

【００４０】

ペアコンポーネント４０２は、メッセージ交換のペアを格納することができるデータストア４０４を使用することができる。データストア４０４は、ＤＳＰＩ（図示せず）内、かつ／またはリモートサーバ上のメモリ内データベースでよいことを理解されたい。データストア４０４は、ＤＳＰＩコンポーネント（図示せず）によってサポートされる要求と応答のペアを保持するのに利用することができる。さらに、データストア４０４は、たとえば、揮発性メモリでも不揮発性メモリでもよく、揮発性および不揮発性メモリを両方とも含むこともできる。限定ではなく例として、不揮発性メモリは、ＲＯＭ（読出し専用メモリ）、ＰＲＯＭ（プログラム可能ＲＯＭ）、ＥＰＲＯＭ（電気的プログラム可能ＲＯＭ）、ＥＥＰＲＯＭ（電気的消去可能ＲＯＭ）、またはフラッシュメモリを含み得る。揮発性メモリは、外部キャッシュメモリとして作用するＲＡＭ（ランダムアクセスメモリ）を含む。限定ではなく例として、ＲＡＭは、ＳＲＡＭ（同期ＲＡＭ）、ＤＲＡＭ（ダイナミックＲＡＭ）、ＳＤＲＡＭ（シンクロナスＤＲＡＭ）、ＤＤＲ　ＳＤＲＡＭ（ダブルデータレート方式ＳＤＲＡＭ）、ＥＳＤＲＡＭ（拡張ＳＤＲＡＭ）、ＳＬＤＲＡＭ（シンクリンクＤＲＡＭ）、ＲＤＲＡＭ（ランバスダイレクトＲＡＭ）、ＤＲＤＲＡＭ（ダイレクトランバスダイナミックＲＡＭ）、およびＲＤＲＡＭ（ランバスダイナミックＲＡＭ）など、多くの形で市販されている。本システムおよび方法のデータストア４０４は、こうしたおよび他のどの適切なタイプのメモリも備えるが、それに限定されないことを意図している。

【００４１】
ペアコンポーネント４０２は、複数のメッセージ交換のペアをサポートする。たとえば、以下のテーブル、すなわちテーブル１は、デバイスとサーバの間で通信を行うのに使用することができる、適切な要求と応答のペア（および記述）のサンプルを示す。
【００４２】
【表１】

コマンド	応答	記述
WriteId (string id, stringpassCode)	状況コードおよび／またはエラー	タグにIDを書き込む。 Id:16進エンコードした文字列として書くタグId。 passCode:任意選択 16進エンコードした文字列としてのパスワード。 Idは、ヌルの場合、writePassCodeコマンドに等しい
GetTagData (string id)	プロバイダ固有文字列としてのユーザデータ	タグIdによって与えられる特定のタグの(ユーザ) データを読み取る。 タグにある全ユーザデータは､細粒度のアクセスなしで返される。
WriteTagData (string id, string data)	状況コードおよび/またはエラー	タグIdによって与えられる特定のタグにユーザデータを書き込む。 データ列は､プロバイダが指定したようにエンコードされている。
GetTagIds()	文字列のリスト	コール時に入手可能なタグおよびデータをすべて読み取り、コレクションとして返す。 1つまたは複数のアンテナから、デバイスによって入手可能なタグすべてをコールの瞬間に返すのが、リアルタイムのコールである。

表１－１

【００４３】

【表２】

GetTagList()	タグリスト	タグリスト中のID/ソース読取りエントリのリスト。
ClearTagList()	状況コードおよび/またはエラー	タグリストをクリアする。
GetTagMetaData (string id)	タグの詳細	タグIdによって与えられるタグの具体的な詳細（タイプ、製造番号、ブロックなど）を読み取る。
LockTag (string id, string lockCode)	状況コードおよび/またはエラー	特定のタグをロックする。 lockCodeは、先に設定されている場合、必要となる。
Kill (string id, string killCode)	状況コードおよび/またはエラー	タグを抹消する。 killCodeは、先に設定されている場合、必要となる。
AddReadFilter (string bitMask, bool incExc)	状況コードおよび/またはエラー	デバイスに簡易ビットマスクフィルタを加える。
RemoveReadFilter (string bitMask, bool incExc)	状況コードおよび/またはエラー	指定されたフィルタをデバイスから取り除く
ClearReadFilters ()	状況コードおよび/またはエラー	デバイス内のフィルタをすべてクリアする。
Reboot()	状況コードおよび/またはエラー	デバイスをリブートする。 デバイスとのどの接続も失われ、確立し直されない。
WriteId (string id, string passCode)	状況コードおよび/またはエラー	タグにＩＤを書き込む。 Id:16進エンコードした文字列として書くタグId。 passCode:16進エンコードした文字列としての任意選択のパスワード。 Idは、ヌルの場合、writePassCodeコマンドに等しい。

表１－２

【００４４】

さらに、ペアコンポーネント４０２は、返信および／または通知（たとえば、それぞれ「CmdResponseEvent」および「NotificationEvent」を介して受信される）を提供し得る。たとえば、通知イベント「ReadTagEvent」は、イベントモードで使用し、タグの読取り／検出イベントを報告するために、デバイスによって送ることができる。

【００４５】

要求応答コンポーネント３０８は、ＤＳＰＩコンポーネント（図示せず）によって定義されたプロパティペアを提供し得るプロパティコンポーネント４０６をさらに含み得る。プロパティペアは、たとえば、メッセージ交換のペアとほぼ同様の振舞いをもつことができる。具体的には、プロパティは、例外を投げない「get」でも「set」でもよい。プロパティは、共通機構（たとえば、後で論じるSendMessage()メソッドを介して送られる要求）によって扱うことができることを理解されたい。標準プロパティは、ＤＳＰＩコンポーネント（図示せず）によって定義することができ、特定のプロバイダが、他のプロパティを定義できることを理解されたい。プロバイダ（１つまたは複数）は、タイムアウト要求のために、（たとえば、ミリ秒で表される）ＲＥＱＵＥＳＴ－ＴＩＭＥＯＵＴと呼ばれる

プロパティをサポートすることができる。プロパティコンポーネント４０６は、データストア４０４を使用してプロパティを格納できることを理解されたい。

【００４６】

プロパティコンポーネント４０６は、複数のプロパティペアをサポートする。たとえば、以下のテーブル、すなわちテーブル２は、デバイスとサーバの間で通信を行うのに使用することができる様々なプロパティペア（および記述）を示す。

【００４７】

【表３】

コマンド	応答	記述
GetProperty(string propertyGroupName, string propertyName)	プロパティオブジェクト。このようなプロパティがない場合はNULL。	指定されたプロパティを入手する。
SetProperty(Property property)	状況コードおよび/またはエラー	プロパティを設定する。
ApplyPropertyProfile(Property[]propertyProfile)	状況コードおよび/またはエラー	このメソッドを用いて、複数のプロパティを同時に適用することができる。このような機能が有用となる次のようなシナリオが存在する。すなわち、デバイスの最適な性能のために、1組のプロパティが設定される必要がある。こうしたプロパティはすべて、デバイス構成においてプロファイルとして指定することができる。こうしたプロパティは、直ちにサーバによって適用することができる。
GetProperty(string propertyGroupName, string propertyName)	プロパティオブジェクト。このようなプロパティがない場合はNULL。	指定されたプロパティを入手する。
SetProperty(Property property)	状況コードおよび/またはエラー	プロパティを設定する。

表２

【００４８】

さらに、プロパティコンポーネント４０６は、複数の標準プロパティをサポートする。たとえば、以下のテーブル、すなわちテーブル３は、標準プロパティ、具体的には、プロパティが読取りおよび／または書込みであるか、ならびに記述を示す。

【００４９】

【表４】

プロパティ	読取り/書込み	必須	記述
REQUEST_TIMEOUT	R/W	Yes	ミリ秒で表した、要求メッセージに対するタイムアウト期間。プロバイダは、この期間が満了した後、タイムアウト応答を送信する。

表３

【００５０】

要求応答コンポーネント３０８は、デバイスとサーバの間のメッセージ交換を使用するために、プログラムコードを使用することができる。プログラムコードは、ＸＭＬ（拡張マークアップ言語）、ＨＴＭＬ（ハイパーテキストマークアップ言語）、ＳＧＭＬ（汎用マークアップ言語）、およびＸＨＴＭＬ（拡張可能ハイパーテキストマークアップ言語）などであるが、それに限定されないマークアップ言語でよい。たとえば、マークアップ言語は、要求、応答、および通知用のシンタクスを提供するのに使用することができる。要求、通知などは、強い型付けをもたらすように、ＤＳＰＩコンポーネント（図示せず）においてプログラミングされたオブジェクトとして表すことができることを理解されたい。

【００５１】

図５は、デバイスとの対話を円滑にして、通信および管理のための一律な技術を提供するシステム５００を示す。デバイスインターフェイスコンポーネント３１０は、メッセージ層およびトランスポート層の定義を円滑にする通信コンポーネント５０２を含み得る。さらに、デバイスインターフェイスコンポーネント３１０は、メッセージ通信および接続管理のための技術をさらに含む。通信コンポーネント５０２は、メッセージ層およびトランスポート層を実装するために、少なくとも１つの送信チャネルおよび少なくとも１つの受信チャネルを含み得ることを理解されたい。

【００５２】

たとえば、通信コンポーネント５０２は、送信チャネル５０４を含み得る。送信チャネルは、情報を送信するために「SendMessage()」を使用することができる。さらに、通信コンポーネント５０２は、第１の受信チャネル５０６および第２の受信チャネル５０８を有し得る。第１の受信チャネル５０６は、応答用（たとえば、要求用）に「CmdResponseEvent」を使用することができ、第２の受信チャネル５０８は、通知用に「NotificationEvent」を使用することができる。応答イベントは、同期した要求－応答コマンド用でよく、通知は、非同期用でよく、通知は、タグリストイベント、リーダ管理イベントなどでよい。デバイスインターフェイスコンポーネント３１０は、「ProviderException」も使用できることを理解されたい。「ProviderException」は、プロバイダ関連の例外すべてに対する最上位レベルの例外である。どの内部例外も、この例外の範囲内で渡すことができる。たとえば、以下のコードが、ProviderExceptionクラスを定義することができる。

【００５３】

【表５】

```
public class ProviderException : ApplicationException {
    public ProviderException(string message):base(message) {
    }
    public ProviderException(string message, Exception e):base(message, e) {
    }
}
```

【００５４】

通信コンポーネント５０２は、デバイスにメッセージを送信することができる。以下の

コードは、メッセージを送信するための「SendMessage()」を実装する例である。

【００５５】

【表６】

```
void SendMessage (ICommand command);
```

【００５６】

通信コンポーネント５０２は、たとえば、ConnectionDownException、SendFailedException、System.ArgumentException（たとえば、メッセージパラメータが無効の場合）などだが、それに限定されない例外も投げ得る。さらに、通信コンポーネント５０２は、以下のコードを使用することができる。

【００５７】

【表７】

```
event ResponseEventHandler CmdResponseEvent;
```

【００５８】

上記のコードは、要求に対する応答が受信されたときに生成されるイベントである。上記のコードは、コマンドへの応答が受信されたとき、デバイスによって与えることができることを理解されたい。さらに、通信コンポーネント５０２は、以下のコードを使用することができる。

【００５９】

【表８】

```
event ResponseEventHandler NotificationEvent;
```

【００６０】

上記のイベントは、通知イベントがデバイスによって受信されたとき、生成される。さらに、上記のコードは、非同期イベントが受信されたとき、デバイスによって与えることができる。デバイスインターフェイスコンポーネント３１０は、様々なコードおよび／またはイベントを使用できることを理解されたい。

【００６１】

以下のコードは、通信用のメッセージ層をオープンするために、デバイスインターフェイスコンポーネント３１０によって実装することができる。標準ＭＬ（ＳＭＬ）メッセージ要求は、以下のコードを使用して送られることを理解されたい。

【００６２】

【表９】

```
public abstract class DeviceInterface
{
        public abstract DeviceInformation DeviceInformation {get;}
        public abstract event ResponseEventHandler CmdResponseEvent;
        public abstract event NotificationEventHandler NotificationEvent;
        public abstract void SendMessage(Command command);
        public abstract bool SetupConnection();
        public abstract void Close();
        public abstract bool IsConnectionAlive();
        public abstract HashTable GetSources();
}

public delegate void ResponseEvent(object source, ResponseEventArgs args);
public delegate void NotificationEvent(object source, NotificationEventArgs
args);
public class ResponseEventArgs:EventArgs
{
private Command command;
public ResponseEventArgs(Command command) {...}
}
public class NotificationEventArgs:EventArgs
{
private Notification notification;
public NotificationEventArgs(string id, Notification.Notification notification)
{...}
}
public class RfidProviderException : ApplicationException {...}
public class ConnectionDownException : RfidProviderException {...}
public class SendFailedException : RfidProviderException {...}
// connection management
public class ConnectionFailedException : RfidProviderException {...}
```

【００６３】

「DeviceInformation DeviceInformation{get;}」は、このデバイスインスタンスに関連するデバイス情報を提供する。デバイス情報は、このデバイスについてのすべての情報を含む。デバイス情報クラスは、発見イベントにおける発見時に、デバイスインターフェイスコンポーネント３１０から提供される。たとえば、デバイス情報クラスは、以下のように定義することができる。

【００６４】

【表１０】

```
public class DeviceInformation
{
private string deviceId;
private ConnectionInformation connectionInfo;
private object providerData;
}
```

【００６５】

「void SendMessage(ICommand command);」は、デバイス層におけるすべてのメッセージを送信するのに使用されるコマンドである。ConnectionDownException（たとえば、デバイスが接続されていないか、または接続状態が低下している）、SendFailedException（たとえば、デバイスへの接続が存在するが、デバイスへのメッセージの送信が失敗した）、System.ArgumentException（たとえば、メッセージパラメータが無効の場合）などの例外を投げることができる。関数「event ResponseEventHandler CmdResponseEvent;」は、要求に対する応答が届く度に生成されるイベントである。「event NotificationEventHandler NotificationEvent;」は、非同期通知イベントがデバイスから受信される度に生成されるイベントである。さらに、通信に関連する例外を処理する際に、以下のコードを使用することができる。

【００６６】

【表１１】

```
public delegate void ResponseEvent(object source, ResponseEventArgs args);
public class ResponseEventArgs:EventArgs {
        public readonly string id;  // same as message id
        public readonly string data; // an xml string
        public ResponseEventArgs(string id, string data) {...}
}
public class ReaderTimeoutException :Exception{
        public readonly string messageId;
        public ReaderTimeoutException(string messageId) {...}
}
```

【００６７】

上記のコードは、プロパティおよび／または管理のための技術を含み得ることを理解されたい。さらに、以下のコードが、以下のような例外関連クラスを作成することができる。

【００６８】

【表１２】

```
public class ProviderException : ApplicationException {...}
public class ConnectionDownException : ProviderException {...}
public class SendFailedException : ProviderException {...}
public class ConnectionFailedException : ProviderException{...}
void SendMessage(string message);
```

【００６９】

デバイスインターフェイスコンポーネント３１０は、デバイスとの維持される接続を管理することもできる。このような接続の管理は、プロバイダに関連づけられた複数のデバイス用でよいことを理解されたい。デバイスへの接続は、様々な関数（後で論じる）を使用して維持される。プロバイダは、このような接続が、関数「Close()」を用いてクローズされるまで、デバイスへの接続を維持することを理解されたい。接続管理は、以下のコードを使用することによって利用することができる。

【００７０】
【表１３】

```
bool SetupConnection();
void Close();
bool IsConnectionAlive();
HashTable GetSources();
```

【００７１】
「bool SetupConnection();」は、（たとえば、このDeviceInterfaceインスタンスによって抽象化された）デバイスとの接続をセットアップする。このメソッドは、例外ConnectionFailedException（たとえば、デバイスへの接続が失敗した）を使用することができる。「void Close();」は、ドライバから、デバイスに関する状態を削除し、かつ／または接続をクローズすることができる。「bool IsConnectionAlive();」は、（たとえば、このDeviceInterfaceインスタンスによって抽象化された）デバイスとの接続が存在する場合、真を返す。「HashTable GetSources();」は、名称と状況（たとえば、ソースが接続されていてアクティブである場合は真）として、デバイスにおけるソース（たとえば、アンテナ）すべてを返す。
【００７２】
図６は、デバイスとの対話を円滑にして、通信および管理のための一律な技術を提供するシステム６００を示す。デバイス発見インターフェイスコンポーネント３１２は、検出された新しいデバイスをＤＳＰＩコンポーネント（図示せず）に通知する通知コンポーネント６０２を含み得る。通知コンポーネント６０２は、プロバイダごとに１つのコンポーネントがインスタンス化されるように極めて効率がよいので、デバイスを処理することが可能になる。サーバは、リソースを制御することができないので、プロバイダが、デバイス発見インターフェイスコンポーネント３１２を実装することを理解されたい。ただし、プロバイダは、このようなデバイス発見インターフェイスコンポーネント３１２を実装することができるが、そのように限定されるわけではない。たとえば、作成されるスレッドの数は、ある特定のタイプのデバイスの数に依存し得る。さらに、発見スレッド（１つまたは複数）が、実行中にエラーに遭遇した場合、プロバイダは、スレッドを実行させるために、このようなエラー（１つまたは複数）を処理することができる。さらに、デバイス発見インターフェイスコンポーネント３１２は、DeviceListenerInterfaceなどだが、それに限定されないインターフェイスを使用することによって、デバイス発見機構を定義することができる。
【００７３】
通知コンポーネント６０２は、以下のコードを使用することによって、新しく検出されたデバイスの発見を可能にし得る。
【００７４】

【表１４】

```
public abstract class DeviceListenerInterface
{
        public abstract event DiscoveryEventHandler DiscoveryEvent;
        public abstract void StartDiscovery();
        public abstract void StopDiscovery();
}

public delegate void DiscoveryEventHandler(object source, DiscoveryEventArgs
args);
public class DiscoveryEventArgs:EventArgs {
private DeviceInformation deviceInfo;
```

【００７５】

発見の開始および中止は、例外「public class DiscoveryException:ApplicationException{...}」を投げることができる。具体的には、「DiscoveryEventHandler」は、すべての発見イベント用のコールバックハンドラ（たとえば、プロバイダが、このイベントを使用してホストにデバイス発見イベントを送信する）であり、「StartDiscovery」は、デバイスの発見を開始するための関数であり、「StopDiscovery」は、新しいデバイスの発見を中止するための関数である。さらに、上記のコードは、発見されたＲＦＩＤデバイスに関連する情報を含むデバイス情報オブジェクトを使用する。

【００７６】

デバイス発見インターフェイスコンポーネント３１２は、デバイスとの接続を円滑にする接続コンポーネント６０４をさらに含む。たとえば、接続コンポーネント６０４は、デバイスに接続するのに必要とされる情報および／またはデータをカプセル化する、「deviceIdXml」などのＸＭＬ文字列を使用することができる。スキーマは、デバイスの一意のＩＤ、プロバイダ名、トランスポート名、標準トランスポートなどを含む接続コンポーネント６０４によって利用することができる。たとえば、接続コンポーネント６０４によって、以下のスキーマを使用することができる。

【００７７】

【表１５】

```
<xsd:element name="deviceInformation" type="DeviceInformation" />
<xsd:complexType name="DeviceInformation">
        <xsd:sequence>
                <xsd:element name="id" type="xsd:string" />
                <xsd:element name="provider" type="xsd:string" />
                <xsd:element name="transport" type="xsd:string" />
                <xsd:element name="providerSpecific" type="xsd:anyType" />
        </xsd:sequence>
</xsd:complexType>
        <xsd:simpleType name="Transport">
                        <xsd:restriction base="xsd:string">
                                <xsd:enumeration value="TCPIP"/>
                                <xsd:enumeration value="SERIAL"/>
                                <xsd:enumeration value="HTTP"/>
                                <xsd:enumeration value="WIRELESS"/>
                        </xsd:restriction>
        </simpleType>
```

【００７８】

デバイス情報クラスは、ＲＦＩＤデバイスについての情報を提供する。このクラスは、発見時に、発見イベント中で、図３のデバイスインターフェイスコンポーネント３１０からプロバイダによって提供される。

【００７９】

さらに、接続情報クラスは、ＲＦＩＤデバイスとの接続のために用いることができる。発見をサポートするデバイス用に、このオブジェクトは、DiscoveryEventのDeviceInformationオブジェクトから入手することができる。発見をサポートしないデバイス用には、このオブジェクトは、ServiceProviderInterfaceオブジェクトにおいてGetDevice()をコールする前に、手作業で埋めることができる。接続情報クラスは、以下のように定義することができる。

【００８０】

【表１６】

```
public class ConnectionInformation
{
private string providerId;
private Transport transport;
private ITransportSettings transportSettings;
}
public enum Transport
{
TcpIP,
Serial,
Http,
Wireless
}
```

【００８１】

図７は、デバイスとの対話を円滑にして、通信および管理のための一律な技術を提供するシステム７００を示す。ＳＰＩプロバイダは、たとえば、多数の利点をもたらすアセン

ブリ（たとえば、．ＮＥＴアセンブリ）でよい。ＳＰＩコンテナコンポーネント３１４は、ドライバ状態を最低限に保持することを理解されたい。言い換えると、発見され、かつ／または接続されているデバイスの量は、ＳＰＩコンポーネント３１４にとって重要でない。

【００８２】

ＳＰＩコンテナコンポーネント３１４は、サービスプロバイダインターフェイスを実装することができるサービスプロバイダコンポーネント７０２を含み得る。サービスプロバイダインターフェイスは、ＤＳＰＩコンポーネント（図示せず）に対するエントリポイントである。ＤＳＰＩの実装者は、エントリポイントを使用するためのインターフェイスを必要とする。インターフェイス中のメソッドは、ProviderExceptionを使用できることを理解されたい。たとえば、以下のコードを利用することができる。

【００８３】

【表１７】

```
public abstract class ServiceProviderInterface : DeviceListenerInterface
{
        public abstract ProviderInformation ProviderInformation {get;};
        public abstract DataEncoding UserDataEncoding {get;}
        public abstract void Init(string hostId, string initXml);
        public abstract void ShutDown();
        public abstract DeviceInterface GetDevice(ConnectionInformation
deviceConnectionInfo);
        public abstract bool IsValidDevice(DeviceInformation deviceInfo);
        public abstract string[] GetPropertyGroupNames();
        public abstract Hashtable GetPropertyMetadata(string
propertyGroupName, string
        propertyName);
        public abstract PropertyProfile GetIdealReaderPropertyProfile();
        public abstract PropertyProfile GetIdealWriterPropertyProfile();
}
```

【００８４】

ドライバクラスは、上記の作成されたインターフェイスをサポートすることを理解されたい。さらに、後で定義される「ProviderException」をコールすることができる。「ProviderInformation ProviderInformation{get;};」は、ＤＳＰＩプロバイダについての情報を提供し、この情報は、ＤＳＰＩホスト（１つまたは複数）によって情報目的に使用することができる。プロバイダ情報クラスは、このような情報を提供するのに使用することができる。たとえば、プロバイダ情報クラスを定義するのに、以下を使用することができる。

【００８５】

【表１８】

```
public class ProviderInformation
{
  private string id;
  private string description;
        private string version;
}
```

【００８６】

「Encoding UserDataEncoding{get;};」は、プロバイダによって期待される符号化を、ユーザデータ（たとえば、アスキー、１６進など）用に提供する。関数「void Init(string hostId, string initXml);」は、プロバイダのコンストラクタである。ホストは、このメソッドを使用してＤＳＰＩプロバイダを初期化する。「hostId」は、プロバイダによって情報目的に使うことができ、「initXml」は、プロバイダ固有のＸＭＬ（たとえば、そのスキーマがプロバイダに固有である）であり、プロバイダによって、このＸＭＬ自体を初期化するのに使用することができる。関数「void Shutdown()」は、デバイスが依然として接続されているときに使うことができ、サーバは、このような接続をクローズする責任がある。この関数は、発見スレッドを中止し、状態条件をクリーンアップし、かつ／またはシステムをリセットする。したがって、この関数は、プロバイダのデストラクタであり、プロバイダをシャットダウンする。「DeviceListenerInterface GetDeviceListener()」は、コード中で引用しているように、発見に使用されるDeviceInterface Listenerインターフェイス（たとえば、デバイスインターフェイスコンポーネント３１０）をサポートするオブジェクトへの参照を返す。さらに、「DeviceInterface GetDevice(ConnectionInformation deviceConnectionInfo」は、DeviceInterfaceインターフェイス用のファクトリとして作用するServiceProviderInterfaceインターフェイスを実装する。DeviceInterfaceインターフェイスは、ホストと通信することができるＲＦＩＤデバイスを抽象する。コード「bool IsValidDevice(DeviceInformation deviceInfo);」は、デバイス情報が、このデバイスによってサポートされる有効なデバイスを表すとともに、接続可能な状態にあるか、または既に接続されている場合、真の戻り値を与えることにも留意されたい。「string[] GetPropertyGroupNames();」は、このプロバイダによってサポートされるプロパティグループの名前を返す。「String.Empty」という名前をもつグローバルプロパティグループは、このメソッドによって返されない。関数「Hashtable GetpropertyMetadata(string propertyGroupName, string propertyName);」は、グループ名と、それに対するプロパティメタデータのマップとして、プロパティについてのメタデータを返す。戻り値は、以下のテーブルによって定義することができる。

【００８７】

【表１９】

	プロパティ名がNULL	プロパティ名がNULLでない
プロパティグループ名がNULL	すべてのグローバルプロパティ用メタデータ：空の文字列とプロパティメタデータ。	名前つきグローバルプロパティ用メタデータ：空の文字列とプロパティメタデータ。このメタデータがグローバルプロパティでない場合、NULLを返す
プロパティグループ名がNULLでない	このグループ中のすべてのプロパティ用メタデータ、グループ名とプロパティメタデータ。	名前つきグループ中の名前つきプロパティ用メタデータ：グループ名とプロパティメタデータ。このメタデータが、このグループに属すプロパティでない場合、NULLを返す

【００８８】

関数「PropertyProfile GetIdealReaderPropertyProfile();」は、このプロバイダ用のリーダ役割にある（たとえば、タグ通知モードにある）ＲＦＩＤデバイス用の、最適なプロパティプロファイルを返す。このプロファイルは、適用されると、リーダを最適に動作

させるプロパティの組を含む。「PropertyProfile GetIdealWriterPropertyProfile();」は、このプロバイダ用のライタ役割（たとえば、同期コマンドモード）にあるＲＦＩＤデバイス用の、最適なプロパティプロファイルを返す。このプロファイルは、適用されると、ライタを最適に動作させるプロパティの組を含む。

【００８９】

ＳＰＩコンテナコンポーネント３１４は、サーバへの、少なくとも１つのドライバの登録を円滑にするドライバコンポーネント７０４を含み得る。ドライバコンポーネント７０４は、検証済みかつ認証済みドライバを認めるために、サーバへのドライバ登録を可能にする。たとえば、ドライバ登録は、互換性および機能を保証する、保護され、かつ／または認証済みのドライバを使用可能にするデジタル署名を使用することができる。たとえば、IDriverManagerインターフェイスを、以下のコードを用いて実装することができる。

【００９０】

【表２０】

```
interface IDriverManager {
        string LoadDriver(string assemblyPath, string className);
        void UnloadDriver (string identifier);
        void StopReadersOfDriver(string identifier);
        string ListLoadedDrivers();
 }
```

【００９１】

IDriverManagerインターフェイスは、たとえば、ＲＦＩＤアクセラレータによってサポートすることができる。さらに、クライアントは、ドライバをリモートに接続し管理するのに、．ＮＥＴプラットフォームを使用できることを理解されたい。後で述べるように、．ＮＥＴプラットフォームは、ドライバおよび汎用性の使用に関連する様々な利点をもたらす。コード「string LoadDriver(AssemblyQualifiedNamespace);」は、ドライバコンポーネント７０４によって実装することができる。パスは、ローカルマシン上に置かれるアセンブリでよく、以下の例外、すなわち、アセンブリの未発見と、アセンブリ中のドライババージョンの、サーバによる未サポートと、アセンブリの識別子の（たとえば、ロードされた複数のドライバに渡る）非一意性とを投げることができ、識別子は、ロードが行われるとドライバを識別する。ドライバが正しくロードされると、「providerInstanceId」が返されることを理解されたい。さらに、ドライバコンポーネント７０４は、サーバ構成からドライバを削除させる関数「void UnloadDriver(string providerInstanceId);」を使用することができる。識別子に対応するドライバの未発見は、関数に関連して投げることができる例外である。

【００９２】

ドライバコンポーネント７０４は、ドライバの管理を円滑にするコード、「string ListLoadedDrivers();」を実装することができる。ドライバコンポーネント７０４は、管理を可能にすることができ、関数「string ListLoadedDrivers();」から戻される文字列は、以下の形式を有するＸＭＬ文字列でよいことを理解されたい。

【００９３】

【表２１】

```
<xsd:element name="drivers" type="Drivers" />
<xsd:complexType name="Drivers">
    <xsd:sequence>
        <xsd:element name="driver" type="Driver"
maxOccurs="unbounded" />
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Driver">
    <xsd:sequence>
        <xsd:element name="name" type="xsd:string" />
        <xsd:element name="assembly" type="xsd:string" />
        <xsd:element name="version" type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>
```

【００９４】

上記のコードは、「Drivers」という参照名をもつ複合タイプを作成することができる。さらに、このコードは、ドライバ名（たとえば、providerInstanceId）、ドライバアセンブリ名、およびドライババージョンを含む。

【００９５】

図８は、デバイスサービスプロバイダインターフェイス（ＤＳＰＩ）コンポーネント８０２を使用して、複数のデバイスおよび関連づけられたプロバイダのための一律な通信および管理の提供を円滑にするシステム８００を示す。プロバイダ８０４は、ＥＰＣ－Ｇ準拠プロバイダ８０８、第１の固有プロバイダ８１０、および第２の固有プロバイダ８１２など、複数のプロバイダを含み得る。プロバイダ８０４は、関連づけられたデバイスへのサービスを可能にすることを理解されたい。たとえば、ＥＰＣ－Ｇ準拠プロバイダ８０８は、ＥＰＣ－Ｇｌｏｂａｌデバイス８１４に関連づけることができ、第１の固有プロバイダ８１０は、第１のレガシーデバイス８１６に関連づけることができ、第２の固有プロバイダ８１２は、第２のレガシーデバイス８１８に関連づけることができる。ＤＳＰＩコンポーネント８０２は、関連づけられたデバイス（たとえば、ＥＰＣ－Ｇｌｏｂａｌデバイス８１４、第１のレガシーデバイス８１６、および第２のレガシーデバイス８１８）用の既存のプロバイダ８０４の１つを介して対話を行うことを理解されたい。ＤＳＰＩコンポーネント８０２は、各デバイスが異なる１組のコマンドをサポートするとき、プロバイダ８０４がミドルウェア製品にサービスを提供するための一律な方法を提供する。言い換えると、ＤＳＰＩコンポーネント８０２は、ＲＦＩＤサーバ８０６に一律にサービスを提供するために、デバイスベンダ用の（たとえば、関連づけられたデバイス用にも）インターフェイスを定義する。

【００９６】

ＤＳＰＩコンポーネント８０２は、プロバイダ８０４の少なくとも１つに固有のメッセージ層コマンドをサポートできることを理解されたい。プロバイダ固有コマンドは、以下のコマンドオブジェクトを使用して送ることができる。

【００９７】

【表２２】

```
public class VendorDefined : Command
  {
    private string vendor;
    private string command;
    private ParameterCollection parameters;
    private VendorDefinedResponse response;
  }
  public class VendorDefinedResponse : Response
  {
    private string reply;
    private vendorData data;
  }
```

【００９８】

上記のオブジェクトは、プロバイダ固有コマンドを文字列として含む。追加パラメータは、ParameterCollectionに入れて渡すことができる。コマンドに対する返信は、返信文字列として送られ、どの追加データも、データオブジェクトに入れられる。さらに、プロバイダ固有の通知は、以下のイベントオブジェクトを使用して送ることができることを理解されたい。

【００９９】

【表２３】

```
public struct VendorDefinedEvent : IEvent
{
    private string id;
    private string vendor;
    private string name;
    private Data data;
}
```

【０１００】

このオブジェクトは、イベントｉｄ、ベンダｉｄ、プロバイダ固有イベント名、およびイベント用のどのプロバイダ固有データも含む。

【０１０１】

本発明の一態様によると、ＤＳＰＩプロバイダは、タグリスト機能を実装することができる。タグリストは、デバイスによって検出されたタグを（たとえば、バッファ制限まで）格納する。タグリストは、DUPLICATE_ELIMINATION_TIMEを使用して重複を排除し、タグリストにタグを格納する。タグリストは、デバイスで検出されたタグを格納するのに使用され、そうすることによって、ホストは、プロバイダに接続されていない場合であっても、タグを失くさなくなる。タグリストは、以下で定義される、タグエントリのコレクションを含む。

【０１０２】

【表２４】

```
public struct Tag
  {
    private string tagId;
    private string tagData;
    private string tagType;
    private DateTime time;
    private string source;
  ...
  }
```

【０１０３】

タグは、タグＩｄ、任意選択のタグデータ、任意選択のタグタイプ、任意選択の時間、および任意選択のソースを格納する。ソースは、単純なプロバイダ固有文字列（たとえば、Antenna1）として格納される。ソース名は、Antenna1、Antenna2などで良いが、それに限定されないことを理解されたい。

【０１０４】

本発明の別の態様によると、検出されたタグのタグｉｄに対して、ビットマスクフィルタを適用することができる。フィルタパターンは、たとえば、０、１、およびｘなどを使用して実装することができ、ｘは、値が決められていないことを表す。フィルタは、長さが一致するタグｉｄにのみ適用される。一例では、フィルタは、通知に対してのみ適用される。同期コマンドは、こうしたフィルタを適用しない。BitMaskReadFilterは、以下を使用して利用することができる。

【０１０５】

【表２５】

```
public class BitMaskReadFilter
  {
    private string bitMask;
    private bool inclusive;
  }
```

【０１０６】

本発明のさらに別の態様によると、プロバイダ８０４の少なくとも１つは、パスコードをサポートする１組のタグコマンドを提供することができる。ユーザが、「WriteID」コマンドを使用してパスコードを設定した場合、パスコードは、こうしたすべての安全な操作に適用可能である。ユーザが、パスコードを設定しなかった場合、こうしたすべての安全な操作は、通常コマンドになる。パスコードは、タグに固有となる。

【０１０７】

図９は、デバイスサービスプロバイダインターフェイス（ＤＳＰＩ）コンポーネント９０２を使用して、デバイスのための一律な通信および／または管理の提供を円滑にするシステム９００を示す。ＤＳＰＩコンポーネント９０２は、通信および管理のための一律な技術を提供することによって、ＲＦＩＤサーバ９０６とＲＦＩＤデバイス９０４の間の通信を円滑にする。たとえば、少なくとも１つのＲＦＩＤデバイス９０４に関連づけられたプロバイダ（図示せず）が、ＤＳＰＩコンポーネント９０２を実装することができる。プロバイダは、ＤＳＰＩコンポーネント９０２によって定義されたインターフェイス（１つまたは複数）を実装する、プロセッサ非依存のプラットフォームアセンブリでよい。

【０１０８】

ＤＳＰＩコンポーネント９０２は、少なくともＤＳＰＩコンポーネント９０２、ＲＦＩＤサーバ９０６、および／またはＲＦＩＤデバイス９０４に基づいて活動のログを提供す

る履歴コンポーネント９０８を使用することができる。履歴コンポーネントは、たとえば、プロバイダリスト、プロバイダに関連づけられたデバイス、接続履歴、接続データ、信号データ、認証データなどの情報を追跡し、かつ／またはログをとることができる。さらに、履歴コンポーネント９０８によって作成される活動ログは、リアルタイムのデータストリームおよび／または格納されているデータでよいことを理解されたい。履歴コンポーネント９０８は、データストア９１４を使用して、このようなログ活動および／または履歴データを格納することができる。このようなデータストア９１４は、後で詳しく述べる。

【０１０９】

ＤＳＰＩコンポーネント９０２は、システム９００に追加セキュリティを提供するために、セキュリティコンポーネント９１０と対話することができる。セキュリティコンポーネント９１０は、認証（たとえば、ログインおよび／またはパスワード）、検証（たとえば、このようなログインおよびパスワードの検証）、保護された接続（たとえば、少なくとも検証に基づく）、セキュリティレベル（たとえば、ユーザ名および／またはパスワードに基づく）、暗号化などを許可することができる。たとえば、セキュリティコンポーネント９１０は、プロバイダが、ＤＳＰＩコンポーネント９０２によって定義されたインターフェイスと対話し、かつ／またはインターフェイスを使用できるようにさせる情報を交換することができる、安全な接続を提供することができる。さらに、システム９００の認証および／または検証は、提供されるドライバに関するユーザの信頼性を高め得る。

【０１１０】

本発明の一態様によると、ＤＳＰＩコンポーネント９０２は、学習コンポーネント９１２を使用して、ＲＦＩＤサーバ９０６とＲＦＩＤデバイス９０４の間の通信の確立を円滑にすることができる。学習コンポーネント９１２は、たとえば、プロバイダ活動／振舞い、デバイス活動／振舞い、セキュリティ傾向などを判定するために、機械学習（たとえば、人工知能）技術を使用することができる。学習コンポーネント９１２は、プロバイダが、ＤＳＰＩコンポーネント９０２によって定義されたインターフェイスを実装することができるかどうか判定するために、たとえば、履歴データおよび／または活動ログなどの履歴コンポーネント９０８に関連づけられた情報を分析することができる。さらに、ある特定のデバイスが接続されるか、かつ／またはサービスを提供されるか判定する際、学習コンポーネント９１２は、少なくとも部分的に人工知能技術に基づいて決定された特性および／または振舞いパターンを分析することができる。

【０１１１】

履歴コンポーネント９０８、セキュリティコンポーネント９１０、および／または学習コンポーネント９１２は、データを格納するためにデータストア９１４を使用できることを理解されたい。データストア９１４は、ＤＳＰＩコンポーネント９０２内、かつ／またはリモートサーバ上に置くことができる。さらに、データストア９１４は、たとえば、揮発性メモリでも不揮発性メモリでもよく、揮発性および不揮発性メモリを両方とも含むこともできる。限定ではなく例として、不揮発性メモリは、ＲＯＭ（読出し専用メモリ）、ＰＲＯＭ（プログラム可能ＲＯＭ）、ＥＰＲＯＭ（電気的プログラム可能ＲＯＭ）、ＥＥＰＲＯＭ（電気的消去可能ＲＯＭ）、またはフラッシュメモリを含み得る。揮発性メモリは、外部キャッシュメモリとして作用するＲＡＭ（ランダムアクセスメモリ）を含む。限定ではなく例として、ＲＡＭは、ＳＲＡＭ（同期ＲＡＭ）、ＤＲＡＭ（ダイナミックＲＡＭ）、ＳＤＲＡＭ（シンクロナスＤＲＡＭ）、ＤＤＲ　ＳＤＲＡＭ（ダブルデータレート方式ＳＤＲＡＭ）、ＥＳＤＲＡＭ（拡張ＳＤＲＡＭ）、ＳＬＤＲＡＭ（シンクリンクＤＲＡＭ）、ＲＤＲＡＭ（ランバスダイレクトＲＡＭ）、ＤＲＤＲＡＭ（ダイレクトランバスダイナミックＲＡＭ）、およびＲＤＲＡＭ（ランバスダイナミックＲＡＭ）など、多くの形で市販されている。本システムおよび方法のデータストア９１４は、こうしたおよび他のどの適切なタイプのメモリも備えるが、それに限定されないことを意図している。

【０１１２】

図１０～１２は、本発明による方法を示す。説明を簡単にするために、この方法は一連

の作用として示してある。本発明は例示した作用および／または作用の順序に限定されないことを理解されたい。たとえば、作用は、様々な順序で、かつ／または並行して起こることができ、本明細書に提示も記述もしていない他の作用とともに起こり得る。さらに、例示するすべての作用が、本発明による方法の実装に必要となるわけではない。さらに、本方法は、別法として、状態図またはイベントによる一連の相関状態として表すことができることが、当業者には理解されよう。

【０１１３】

図１０は、対話のための一律な技術を使用することによって、デバイスとの通信を円滑にする方法１０００を示す。１００２で、プロバイダを、サーバにロードすることができる。具体的には、ＳＰＩコンテナが、ＲＦＩＤサーバにプロバイダ（１つまたは複数）をロードすることができる。サーバにプロバイダ（１つまたは複数）をロードすると、ＳＰＩのバージョンは、このようなサーバと互換性があることを検証することができる。さらに、ドライバ（１つまたは複数）は、互換ドライバ（１つまたは複数）を提供するために（たとえば、デジタル検証）、ＲＦＩＤサーバと統合することができる。

【０１１４】

１００４で、デバイスが発見され構成される。発見は、１つのプロバイダに対して１つのコンポーネントがインスタンス化され、デバイスを使用することができるように効率的である。さらに、プロバイダは、発見処理中に発見スレッドエラーを処理する。デバイスを発見した後、文字列は、デバイスを接続し、かつ／または構成するのに必要とされる情報を提供することができる。たとえば、文字列は、マークアップ言語形式（たとえば、ＸＭＬ（拡張マークアップ言語）、ＨＴＭＬ（ハイパーテキストマークアップ言語）、ＳＧＭＬ（汎用マークアップ言語）、およびＸＨＴＭＬ（拡張可能ハイパーテキストマークアップ言語）など）でよく、たとえば、デバイスの一意のｉｄ、プロバイダ名、トランスポート名などだが、それに限定されない情報を有する。

【０１１５】

次に１００６で、デバイスに一律なサービスを提供するために、インターフェイスを定義し使用することができる。プロバイダがＲＦＩＤサーバにロードされ、デバイスが発見され、かつ／または構成されると、インターフェイス（１つまたは複数）は、一律な技術を用いて通信を行うために、プロバイダによって使用することができる。定義されたインターフェイスは、新しいデバイスの発見および構成を提供することができ、さらに、通信は、すべての（たとえば、新しいおよび確立された）デバイスの接続管理を提供することができる。一律な技術（１つまたは複数）を使用することによって、多数の標準に渡る正規化がもたらされる。

【０１１６】

図１１は、デバイスとの通信を円滑にする方法１１００を示す。１１０２で、プロバイダが、ＲＦＩＤサーバにロードされる。認証されると、プロバイダはロードされ、少なくとも１つのインターフェイスを使用することによって、複数のサービスを一律に提供することができる。１１０４で、メッセージ層および／またはトランスポート層が定義される。メッセージ通信および／または接続管理は、メッセージ層および／またはトランスポート層内で提供される。たとえば、送信チャネルは、メッセージ交換を送信するのに、「SendMessage()」を使用することができる。さらに、応答用の第１の受信チャネルは、「CmdResponseEvent」を実装することができ、通知用の第２の受信チャネルは、「NotificationEvent」を使用することができる。接続管理は、ＸＭＬ文字列を使用して提供することができる。プロバイダは、関数「Close()」を用いて接続がクローズされるまで、デバイスとの接続を維持することができることを理解されたい。

【０１１７】

１１０６で、デバイス（１つまたは複数）は、メッセージ層および／またはトランスポート層を使用することによって発見し構成することができる。デバイス（１つまたは複数）の発見および構成は、メッセージ交換のペアを用いて行われる。メッセージ交換のペアは、サーバとデバイスの間で交換される。メッセージ交換のペアは、非同期であり、メッ

セージ識別を使用してマッチングすることができることを理解されたい。後で述べるように、「SendMessage()」は、デバイスに要求を送信するのに使うことができ、返信および通知は、それぞれ「CmdResponseEvent」および「NotificationEvent」を用いて受信される。さらに、メッセージ交換のペアは、デバイス（１つまたは複数）を構成するのにプロパティを使用することができる。標準プロパティは、構成および／または発見処理によっても定義し実装することができることを理解されたい。

【０１１８】

次に１１０８で、発見され構成されたデバイスに、ＲＦ１Ｄサーバを介して一律なサービスを提供するために、インターフェイスが定義され使用される。メッセージ交換のペアは、既存の、および／または新たに検出されたデバイスと通信するための一律な技術を提供することができる。発見、構成、通信、および接続管理のための多数のインターフェイスを定義することができることを理解されたい。概して、方法１１００は、デバイスと一律に通信しデバイスを管理するための抽象化層を提供することができる。

【０１１９】

図１２は、デバイスと通信し、かつ／またはデバイスを管理するための一律な技術を提供する方法１２００を示す。１２０２で、サービスを提供するために、ＲＦＩＤプロバイダがＲＦＩＤサーバにロードされる。認証および／または検証が行われると、それに従って新しいプロバイダが検出されロードされることを理解されたい。１２０４で、メッセージ層およびトランスポート層が定義され、デバイスサービスプロバイダインターフェイス（ＤＳＰＩ）コンポーネントおよびＲＦＩＤデバイスから、通信機能を提供する。１２０６で、新しいデバイスが検出されたかどうか、判定が行われる。新しいデバイスが検出された場合、プロセスは１２０８に進み、ここで、デバイスが適宜接続され、認証され、構成される。新しいデバイスが検出されない場合、プロセスは、１２１０へ続き、ここで、検出された（たとえば、発見され、検証され、かつ／または認証された）ＲＦＩＤデバイスに一律なサービスを提供するために、インターフェイスを定義し使用することができる。

【０１２０】

プロセス１２００は、新しいプロバイダおよび／または関連づけられた新しいデバイスがＲＦＩＤサーバ内に存在するかどうかに関して、絶え間ないポーリングおよび／または期間判定を行うことを理解されたい。言い換えると、検出されたデバイスに一律なサービスを提供する間、プロセスは、新しいプロバイダを検出すると、１２０２に進むことができ、ここで、プロバイダをロードすることができ、関連づけられたデバイスを発見し、接続し、認証することができる。さらに、予め存在するプロバイダ上に新しいデバイスを確立することができることを理解されたい。したがって、プロセスは、このような新しいデバイスを適宜発見し構成することができる。ほぼ絶え間のないポーリングおよび／または定期的な検査は、新しいデバイスおよび／またはプロバイダの判定を利用することができる。

【０１２１】

本発明の様々な態様を実装する状況をさらに提供するために、図１３～１４および以下の説明は、本発明の様々な態様を実装することができる適切な計算機環境の、簡潔で一般的な説明を提供することを意図している。これまでは、ローカルコンピュータおよび／またはリモートコンピュータ上で実行されるコンピュータプログラム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本発明を説明したが、本発明は他のプログラムモジュールとの組合せでも実装できることが当業者には理解されよう。概して、プログラムモジュールは、特定のタスクを実施し、かつ／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、コンポーネント、データ構造などを含む。

【０１２２】

さらに、本発明の方法は、他のコンピュータシステム構成とともに実施できることが当業者には理解されよう。他のコンピュータシステム構成は、シングルプロセッサコンピュータシステムまたはマルチプロセッサコンピュータシステム、ミニコンピュータ、メイン

フレームコンピュータ、ならびにパーソナルコンピュータ、ハンドヘルドコンピューティングデバイス、マイクロプロセッサベースの家電製品および／またはプログラム可能な家電製品などを含み、こうしたシステムはそれぞれ、１つまたは複数の関連づけられたデバイスと動作可能に通信することができる。例示した本発明の態様は、通信ネットワークを介してリンクされるリモート処理デバイスによって特定のタスクが実施される分散型計算機環境でも実施することができる。ただし、すべてではなくともいくつかの本発明の態様は、スタンドアロンコンピュータにおいても実施することができる。分散型計算機環境では、プログラムモジュールは、ローカルメモリ記憶デバイスおよびリモートメモリ記憶デバイス内両方に配置することができる。

【０１２３】

図１３は、本発明と相互作用し得る例としての計算機環境１３００の概略的なブロック図である。システム１３００は、１つまたは複数のクライアント（群）１３１０を含む。クライアント（群）１３１０は、ハードウェアおよび／またはソフトウェア（たとえば、スレッド、プロセス、コンピューティングデバイス）でよい。システム１３００は、１つまたは複数のサーバ（群）１３２０も含む。サーバ（群）１３２０は、ハードウェアおよび／またはソフトウェア（たとえば、スレッド、プロセス、コンピューティングデバイス）でよい。サーバ１３２０は、たとえば、本発明を利用して変換を実施するためのスレッドを収容することができる。

【０１２４】

クライアント１３１０とサーバ１３２０の間で起こり得る１つの通信は、２つ以上のコンピュータプロセスの間で伝送されるように適合されたデータパケットの形をとり得る。システム１３００は、クライアント（群）１３１０とサーバ（群）１３２０の間の通信を円滑にするのに利用することができる通信フレームワーク１３４０を含む。クライアント（群）１３１０は、クライアント（群）１３１０にローカルな情報を格納するのに利用することができる、１つまたは複数のクライアントデータストア（群）１３５０に動作可能に接続される。同様に、サーバ（群）１３２０は、サーバ１３４０にローカルな情報を格納するのに利用することができる、１つまたは複数のサーバデータストア（群）１３３０に動作可能に接続される。

【０１２５】

図１４を参照すると、本発明の様々な態様を実装する例示的な環境１４００は、コンピュータ１４１２を含む。コンピュータ１４１２は、処理ユニット１４１４、システムメモリ１４１６、およびシステムバス１４１８を含む。システムバス１４１８は、システムメモリ１４１６を含むがそれに限定されないシステムコンポーネントを処理ユニット１４１４に結合する。処理ユニット１４１４は、市販されている様々なプロセッサのいずれでもよい。デュアルマイクロプロセッサおよび他のマルチプロセッサアーキテクチャも、処理ユニット１４１４として利用することができる。

【０１２６】

システムバス１４１８は、市販されている様々などのバスアーキテクチャも用いるメモリバスもしくはメモリコントローラ、周辺バスもしくは外部バス、および／またはローカルバスを含むいくつかのタイプのバス構造（群）のどれでもよく、こうしたバス構造は、ＩＳＡ（業界標準アーキテクチャ）、ＭＳＡ（マイクロチャネルアーキテクチャ）、ＥＩＳＡ（拡張ＩＳＡ）、ＩＤＥ（インテリジェントドライブエレクトロニクス）、ＶＬＢ（ＶＥＳＡローカルバス）、ＰＣＩ（周辺装置相互接続）、カードバス、ＵＳＢ（ユニバーサルシリアルバス）、ＡＧＰ（拡張グラフィックスポート）、ＰＣＭＣＩＡ（ＰＣカードアダプタ）、ファイアワイア（ＩＥＥＥ１３９４）、およびＳＣＳＩ（小型コンピュータシステムインターフェイス）を含むが、それに限定されない。

【０１２７】

システムメモリ１４１６は、揮発性メモリ１４２０および不揮発性メモリ１４２２を含む。たとえば起動中に、コンピュータ１４１２内部の要素の間で情報を転送するための基本ルーチンを含む基本入出力システム（ＢＩＯＳ）が、不揮発性メモリ１４２２に格納さ

れる。限定ではなく例として、不揮発性メモリ１４２２は、ＲＯＭ（読出し専用メモリ）、ＰＲＯＭ（プログラム可能ＲＯＭ）、ＥＰＲＯＭ（電気的プログラム可能ＲＯＭ）、ＥＥＰＲＯＭ（電気的消去可能プログラム可能ＲＯＭ）、またはフラッシュメモリを含み得る。揮発性メモリ１４２０は、外部キャッシュメモリとして作用するＲＡＭ（ランダムアクセスメモリ）を含む。限定ではなく例として、ＲＡＭは、ＳＲＡＭ（静的ＲＡＭ）、ＤＲＡＭ（ダイナミックＲＡＭ）、ＳＤＲＡＭ（シンクロナスＤＲＡＭ）、ＤＤＲ　ＳＤＲＡＭ（ダブルデータレート方式ＳＤＲＡＭ）、ＥＳＤＲＡＭ（拡張ＳＤＲＡＭ）、ＳＬＤＲＡＭ（シンクリンクＤＲＡＭ）、およびＲＤＲＡＭ（ランバスダイレクトＲＡＭ）、ＤＲＤＲＡＭ（ダイレクトランバスダイナミックＲＡＭ）、ＲＤＲＡＭ（ランバスダイナミックＲＡＭ）など、多くの形で市販されている。

【０１２８】

コンピュータ１４１２は、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む。図１４は、たとえばディスク記憶装置１４２４を示す。ディスク記憶装置１４２４は、磁気ディスクドライブ、フロッピー（登録商標）ディスクドライブ、テープドライブ、Ｊａｚドライブ、Ｚｉｐドライブ、ＬＳ－１００ドライブ、フラッシュメモリカード、またはメモリスティックなどのデバイスを含むが、それに限定されない。さらに、ディスク記憶装置１４２４は、記憶媒体を、別個に含むことも、ＣＤ－ＲＯＭ（コンパクトディスクＲＯＭデバイス）、ＣＤ－Ｒドライブ（書込み可能ＣＤドライブ）、ＣＤ－ＲＷドライブ（書換え可能ＣＤドライブ）、またはＤＶＤ－ＲＯＭ（デジタルビデオディスクＲＯＭドライブ）などの光ディスクドライブを含むがそれに限定されない他の記憶媒体と組み合わせて含むこともできる。システムバス１４１８へのディスク記憶装置１４２４の接続を円滑にするために、取外し可能または固定式インターフェイスは通常、インターフェイス１４２６などとして使われる。

【０１２９】

図１４は、適切な動作環境１４００において述べられる、ユーザと基本的なコンピュータリソースとの間の媒介として作用するソフトウェアを示すことを理解されたい。このようなソフトウェアは、オペレーティングシステム１４２８を含む。オペレーティングシステム１４２８は、ディスク記憶装置１４２４に格納することができ、コンピュータシステム１４１２のリソースを制御し割り振るように作用する。システムアプリケーション１４３０は、システムメモリ１４１６またはディスク記憶装置１４２４のどちらかに格納された、プログラムモジュール１４３２およびプログラムデータ１４３４を介して、オペレーティングシステム１４２８によるリソース管理を活用する。本発明は、様々なオペレーティングシステム、またはオペレーティングシステムの組合せを用いて実装できることを理解されたい。

【０１３０】

ユーザは、入力デバイス（群）１４３６を介して、コマンドまたは情報をコンピュータ１４１２に入力する。入力デバイス１４３６は、ポインティングデバイス、たとえばマウス、トラックボール、スタイラス、タッチパッド、キーボード、マイクロホン、ジョイスティック、ゲームパッド、衛星パラボラアンテナ、スキャナ、ＴＶチューナカード、デジタルカメラ、デジタルビデオカメラ、ウェブカメラなどを含むが、それに限定されない。こうしたおよび他の入力デバイスは、インターフェイスポート（群）１４３８を介して、システムバス１４１８によって処理ユニット１４１４に接続される。インターフェイスポート（群）１４３８は、たとえば、シリアルポート、パラレルポート、ゲームポート、およびＵＳＢ（ユニバーサルシリアルバス）を含む。出力デバイス（群）１４４０は、入力デバイス（群）１４３６と同じタイプのポートの一部を使う。したがって、たとえば、ＵＳＢポートは、コンピュータ１４１２への入力を可能にし、コンピュータ１４１２から出力デバイス１４４０に情報を出力するのに用いることができる。出力アダプタ１４４２は、他の出力デバイス１４４０の中でも、専用アダプタを必要とする、モニタ、スピーカ、およびプリンタのようないくつかの出力デバイス１４４０があることを示すために図示してある。出力アダプタ１４４２は、限定ではなく例として、出力デバイス１４４０とシス

テムバス１４１８の間の接続手段を提供するビデオカードおよびサウンドカードを含む。他のデバイスおよび／またはデバイスからなるシステムは、リモートコンピュータ（群）１４４４のように、入力および出力機能両方を提供することに留意されたい。

【０１３１】

コンピュータ１４１２は、１つまたは複数のリモートコンピュータ、たとえばリモートコンピュータ（群）１４４４への論理接続を使用してネットワーク接続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（群）１４４４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、ワークステーション、マイクロプロセッサベースの機器、ピアデバイスまたは他の共通ネットワークノードなどでよく、通常、コンピュータ１４１２に関連して説明した要素の多くまたはすべてを含む。簡潔にするために、メモリ記憶装置１４４６のみをリモートコンピュータ（群）１４４４とともに示してある。リモートコンピュータ（群）１４４４は、ネットワークインターフェイス１４４８を介してコンピュータ１４１２に論理的に接続され、さらに通信接続１４５０を介して物理的に接続される。ネットワークインターフェイス１４４８は、ＬＡＮ（ローカルエリアネットワーク）およびＷＡＮ（ワイドエリアネットワーク）などの有線および／または無線通信ネットワークを包含する。ＬＡＮ技術は、ＦＤＤＩ（光ファイバ分散データインターフェイス）、ＣＤＤＩ（銅線配線データインターフェイス）、イーサネット（登録商標）、トークンリングなどを含む。ＷＡＮ技術は、二地点間リンク、ＩＳＤＮ（統合サービスデジタルネットワーク）のような回路交換ネットワークおよびその変形、パケット交換ネットワーク、ならびにＤＳＬ（デジタル加入者線）を含むが、それに限定されない。

【０１３２】

通信接続（群）１４５０は、ネットワークインターフェイス１４４８をバス１４１８に接続するのに利用されるハードウェア／ソフトウェアを指す。通信接続１４５０は、説明をわかりやすくするためにコンピュータ１４１２内部に示してあるが、コンピュータ１４１２の外部にあってもよい。ネットワークインターフェイス１４４８への接続に必要なハードウェア／ソフトウェアは、単なる例として、標準的な電話レベルのモデム、ケーブルモデム、およびＤＳＬモデムを含むモデム、ＩＳＤＮアダプタ、ならびにイーサネット（登録商標）カードなど、内部および外部技術を含む。

【０１３３】

上で説明した内容は、本発明のいくつかの例を含む。当然ながら、本発明を説明するためのコンポーネントまたは方法のあらゆる組合せを説明することはできないが、本発明のさらに多くの組合せおよび入替えが可能であることが当業者には理解されよう。したがって、本発明は、添付の特許請求の範囲の精神および範囲内であるこのようなすべての変更形態、修正形態、および変形形態を包含することを意図したものである。

【０１３４】

特に、上述したコンポーネント、デバイス、回路、システムなどによって実施される様々な機能に関して、このようなコンポーネントを説明するのに用いた用語（「手段」への言及も含む）は、特に示さない限り、説明した（たとえば、機能的に等価な）コンポーネントの指定された機能を実施するとともに、開示した構造と構造的に等価でないとしても、本明細書において示した本発明の例示的な態様においてそうした機能を実施する、どのコンポーネントにも対応することを意図している。この点に関して、本発明は、システム、ならびに本発明の様々な方法の作用および／またはイベントを実施するコンピュータ実行可能命令を有するコンピュータ可読媒体を含むことも理解されよう。

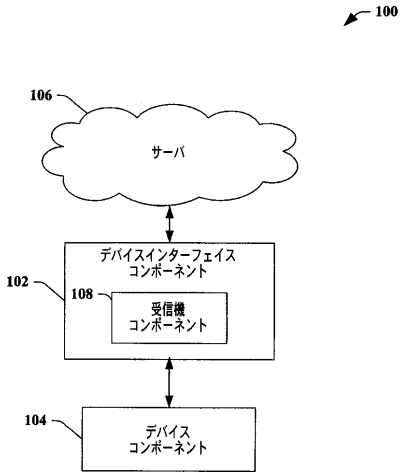
【０１３５】

さらに、本発明のある特定の特徴は、いくつかの実装形態のただ１つに関して開示したが、このような特徴は、所与のまたは特定のどのアプリケーションにとっても望ましく、有利であるように、他の実装形態の１つまたは複数の他の特徴と組み合わせることができる。さらに、詳細な説明または特許請求の範囲において「含む」という用語およびその変形が使われている限りでは、そうした用語は、「備える」という用語と同様に包括的であることを意図している。

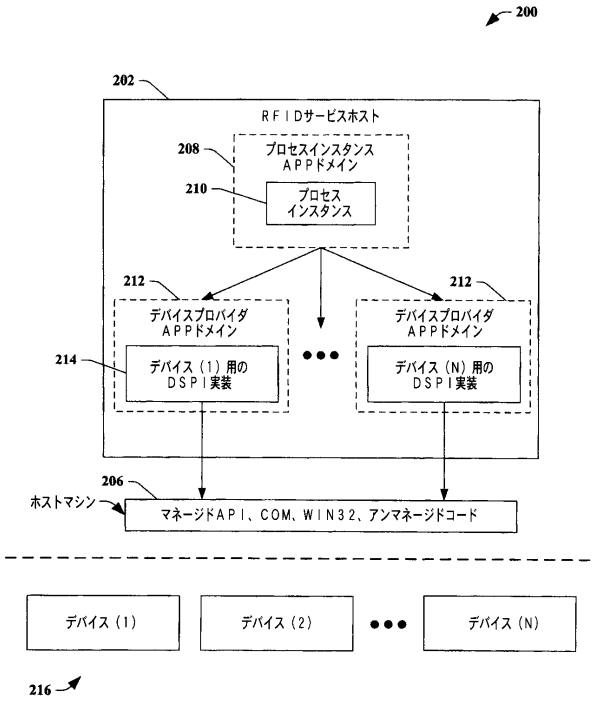
【図面の簡単な説明】
【０１３６】
【図１】デバイスコンポーネントとの対話を円滑にする例示的なシステムを示すブロック図である。
【図２】デバイスとの対話を円滑にする例示的なシステムを示すブロック図である。
【図３】無線ＩＣタグシステムにおいて、デバイスとの対話を円滑にする例示的なシステムを示すブロック図である。
【図４】デバイスとの通信を円滑にする例示的なシステムを示すブロック図である。
【図５】デバイスとの通信を円滑にする例示的なシステムを示すブロック図である。
【図６】デバイスとの通信を円滑にする例示的なシステムを示すブロック図である。
【図７】デバイスとの通信を円滑にする例示的なシステムを示すブロック図である。
【図８】複数のデバイスおよび関連づけられたプロバイダとの通信を円滑にする、例示的なシステムを示すブロック図である。
【図９】デバイスとの通信を円滑にする例示的なシステムを示すブロック図である。
【図１０】デバイスに対する一律なサービスの提供を円滑にする例示的な方法を示すフローチャートである。
【図１１】デバイスに対する一律なサービスの提供を円滑にする例示的な方法を示すフローチャートである。
【図１２】デバイスに対する一律なサービスの提供を円滑にする例示的な方法を示すフローチャートである。
【図１３】本発明の新規な態様を利用することができる、例示的なネットワーク接続環境を示す図である。
【図１４】本発明の新規な態様を利用することができる、例示的な動作環境を示す図である。
【符号の説明】
【０１３７】
１０２　デバイスインターフェイスコンポーネント
１０４　デバイスコンポーネント
１０６　サーバ
１０８　受信機コンポーネント
２０２　ＲＦＩＤサービスホスト
２０６　ホストマシン
２０８　プロセスインスタンスＡＰＰドメイン
２１０　プロセスインスタンス
２１２　デバイスプロバイダＡＰＰドメイン
２１４　デバイス（１）用のＤＳＰＩ実装、デバイス（Ｎ）用のＤＳＰＩ実装
２１６　デバイス（１）、デバイス（２）、デバイス（Ｎ）
３０２　ＤＳＰＩコンポーネント
３０４　ＲＦＩＤデバイス
３０６　ＲＦＩＤサーバ
３０８　要求応答コンポーネント
３１０　デバイスインターフェイスコンポーネント
３１２　デバイス発見インターフェイスコンポーネント
３１４　ＳＰＩコンテナコンポーネント
４０２　ペアコンポーネント
４０４　データストア
４０６　プロパティコンポーネント
５０２　通信コンポーネント
５０４　送信
５０６　受信１

５０８　受信２
６０２　通知コンポーネント
６０４　接続コンポーネント
７０２　サービスプロバイダコンポーネント
７０４　ドライバコンポーネント
８０２　ＤＳＰＩコンポーネント
８０４　プロバイダ
８０６　ＲＦＩＤサーバ
８０８　ＥＰＣ－Ｇ準拠プロバイダ
８１０　第１の固有プロバイダ
８１２　第２の固有プロバイダ
８１４　ＥＰＣ－ＧＬＯＢＡＬデバイス
８１６　第１のレガシーデバイス
８１８　第２のレガシーデバイス
９０２　ＤＳＰＩコンポーネント
９０４　ＲＦＩＤデバイス
９０６　ＲＦＩＤサーバ
９０８　履歴コンポーネント
９１０　セキュリティコンポーネント
９１２　学習コンポーネント
９１４　データストア
１３１０　クライアント（群）
１３２０　サーバ（群）
１３３０　サーバデータストア（群）
１３４０　通信フレームワーク
１３５０　クライアントデータストア（群）
１４１４　処理ユニット
１４１６　システムメモリ
１４１８　バス
１４２０　揮発性メモリ
１４２２　不揮発性メモリ
１４２４　ディスク記憶装置
１４２６　インターフェイス
１４２８　オペレーティングシステム
１４３０　アプリケーション
１４３２　モジュール
１４３４　データ
１４３６　入力デバイス（群）
１４３８　インターフェイスポート（群）
１４４０　出力デバイス（群）
１４４２　出力アダプタ（群）
１４４４　リモートコンピュータ（群）
１４４６　メモリ記憶装置
１４４８　ネットワークインターフェイス
１４５０　通信接続（群）

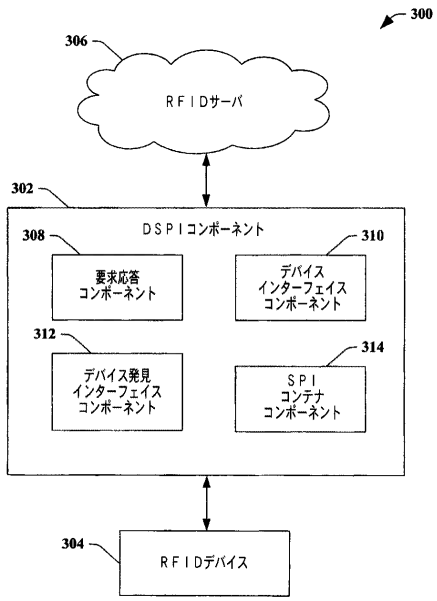
【図１】


100
106
サーバ
デバイスインターフェイス
コンポーネント
102
108
受信機
コンポーネント
104
デバイス
コンポーネント

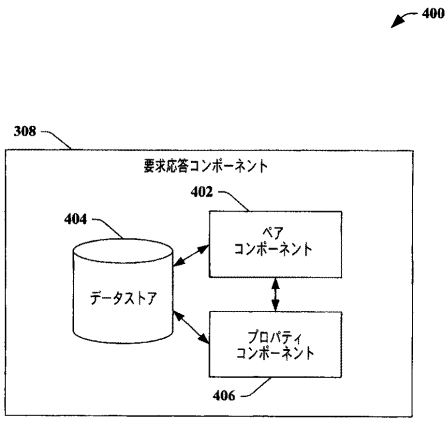
【図２】


200
202
RFIDサービスホスト
208
プロセスインスタンス
APPドメイン
210
プロセス
インスタンス
212
212
デバイスプロバイダ
APPドメイン
デバイスプロバイダ
APPドメイン
214
デバイス（1）用の
DSPI実装
デバイス（N）用の
DSPI実装
206
ホストマシン
マネージドAPI、COM、WIN32、アンマネージドコード
デバイス（1）
デバイス（2）
デバイス（N）
216

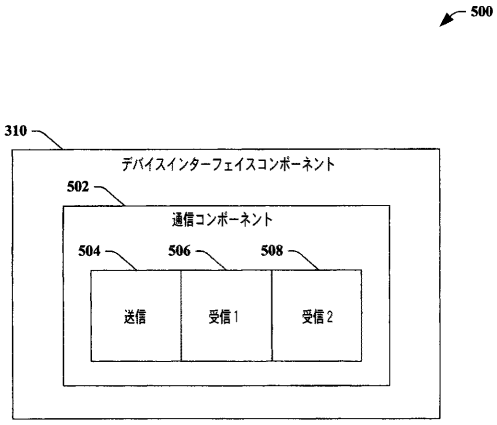
【図３】


300
306
RFIDサーバ
302
DSPIコンポーネント
308
要求応答
コンポーネント
310
デバイス
インターフェイス
コンポーネント
312
デバイス発見
インターフェイス
コンポーネント
314
SPI
コンテナ
コンポーネント
304
RFIDデバイス

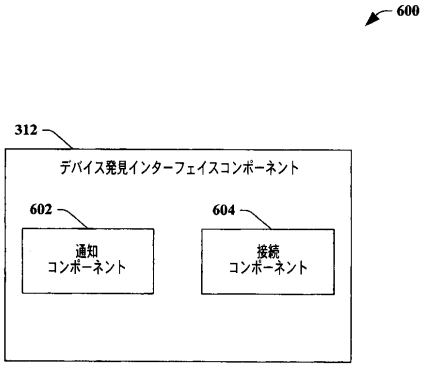
【図４】


400
308
要求応答コンポーネント
402
404
ペア
コンポーネント
データストア
プロパティ
コンポーネント
406

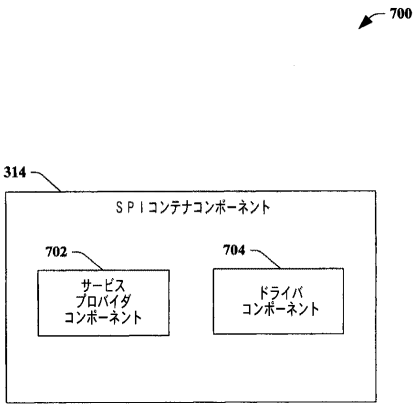
【図５】


500
310
デバイスインターフェイスコンポーネント
502
通信コンポーネント
504
506
508
送信
受信1
受信2

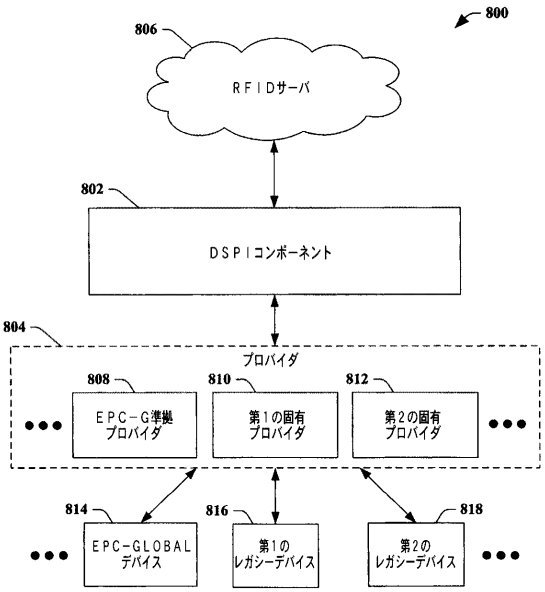
【図６】


600
312
デバイス発見インターフェイスコンポーネント
602
通知
コンポーネント
604
接続
コンポーネント

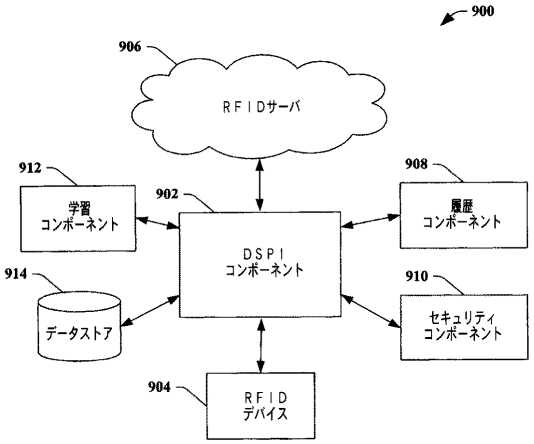
【図７】


700
314
SPIコンテナコンポーネント
702
サービス
プロバイダ
コンポーネント
704
ドライバ
コンポーネント

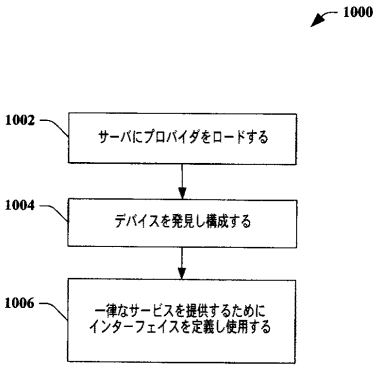
【図８】


800
806
RFIDサーバ
802
DSPIコンポーネント
804
プロバイダ
808
EPC-G準拠
プロバイダ
810
第1の固有
プロバイダ
812
第2の固有
プロバイダ
814
EPC-GLOBAL
デバイス
816
第1の
レガシーデバイス
818
第2の
レガシーデバイス

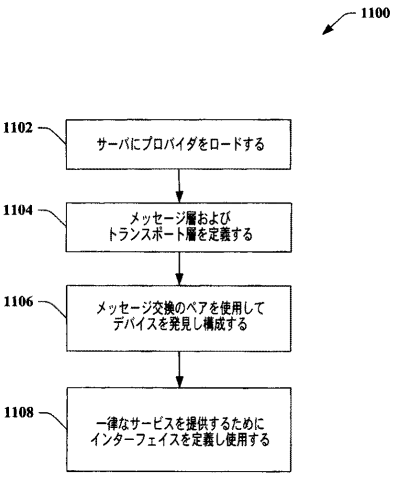
【図９】



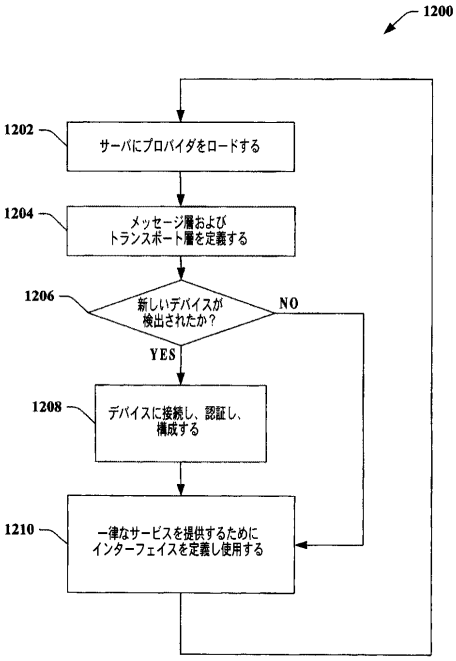
【図１０】



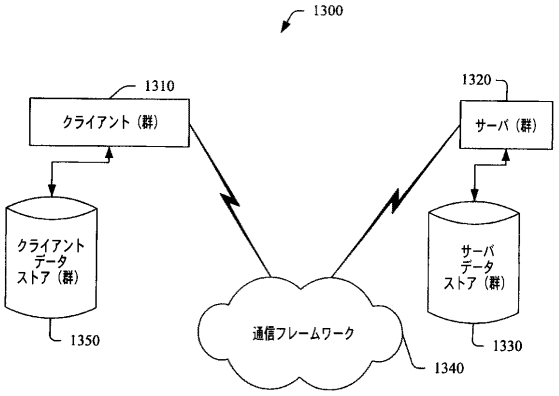
【図１１】



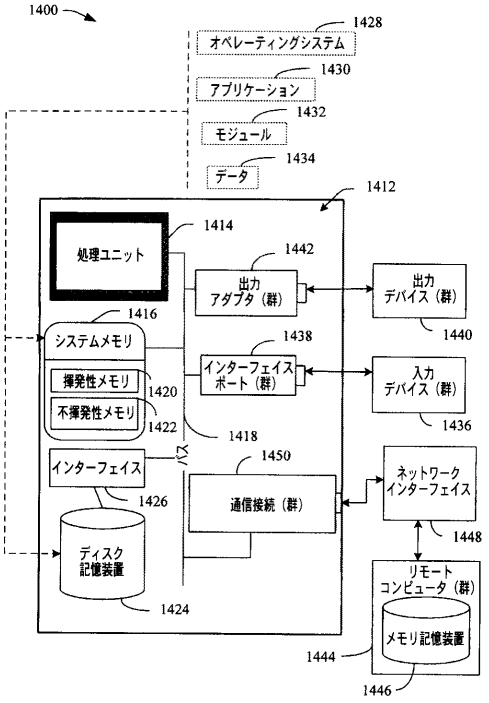
【図１２】



【図１３】


1300
1310
クライアント（群）
1320
サーバ（群）
クライアント
データ
ストア（群）
1350
通信フレームワーク
1340
サーバ
データ
ストア（群）
1330

【図１４】


1400
1428
オペレーティングシステム
1430
アプリケーション
1432
モジュール
1434
データ
1412
処理ユニット
1414
1442
出力
アダプタ（群）
出力
デバイス（群）
1440
1416
システムメモリ
1438
インターフェイス
ポート（群）
入力
デバイス（群）
1436
揮発性メモリ
1420
不揮発性メモリ
1422
1418
インターフェイス
1450
通信接続（群）
ネットワーク
インターフェイス
1448
1426
ディスク
記憶装置
1424
リモート
コンピュータ（群）
メモリ記憶装置
1444
1446

フロントページの続き

(72)発明者　アブヒシェック　アガーワル
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　アヌシュ　クマー
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　バラスブラマニアン　スリラム
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　モハメド　ファクルディーン　アリ　アハメド
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内
(72)発明者　ジャナキ　ラム　ゴテティ
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　バムシドハー　ジー．アール．レッディー
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　ビノド　アナンサラマン
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

審査官　宮下　浩次

(56)参考文献　特開平０７－２９６１２４（ＪＰ，Ａ）
米国特許出願公開第２００３／２２５９２８（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｋ　１７／００
Ｇ０６Ｆ　１３／１０