(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2011-171983
(P2011-171983A)
(43)公開日　平成23年9月1日(2011.9.1)

(51)Int.Cl.		F I			テーマコード（参考）
H04L 29/08	(2006.01)	H04L	13/00	307Z	5B285
H04L 29/06	(2006.01)	H04L	13/00	305D	5K034
G06F 21/20	(2006.01)	G06F	15/00	330B	

審査請求 未請求　請求項の数 13　ＯＬ　（全 22 頁）

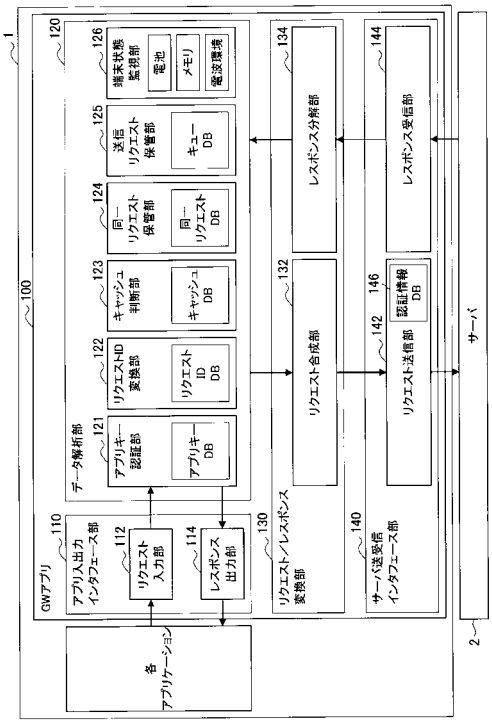
(21)出願番号　特願2010-33550(P2010-33550)
(22)出願日　平成22年2月18日(2010.2.18)

(71)出願人　000002185
ソニー株式会社
東京都港区港南１丁目７番１号
(74)代理人　100095957
弁理士　亀谷　美明
(74)代理人　100096389
弁理士　金本　哲男
(74)代理人　100101557
弁理士　萩原　康司
(74)代理人　100128587
弁理士　松本　一騎
(72)発明者　園田　修平
東京都港区港南１丁目７番１号　ソニー株式会社内

最終頁に続く

(54)【発明の名称】情報処理装置、情報処理方法およびコンピュータ読み取り可能な記録媒体

(57)【要約】
【課題】クライアント‐サーバ間での情報の送受信を制御し、クライアント‐サーバ間の通信量を低減することが可能な情報処理装置を提供する。
【解決手段】本発明の情報処理装置は、ネットワーク１０を介してデータを保持するサーバ２と接続されたクライアント端末１に設けられ、クライアント端末１内の１または複数のアプリケーションからのリクエストを受けて、サーバ２との情報の送受信を制御する。かかる情報処理装置は、サーバ２にアクセスするためのユーザの認証情報を保持する認証情報記憶部と、クライアント端末１内のアプリケーションからのリクエストに基づいて、リクエストに当該クライアント端末１のユーザの認証情報を付加して、サーバへ送信するリクエスト送信部１４２と、を備える。
【選択図】図３



【特許請求の範囲】
【請求項１】
ネットワークを介してデータを保持するサーバと接続されたクライアント端末に設けられ、
前記クライアント端末内の１または複数のアプリケーションからのリクエストを受けて、前記サーバとの情報の送受信を制御する、情報処理装置。
【請求項２】
前記サーバにアクセスするためのユーザの認証情報を保持する認証情報記憶部と、
前記クライアント端末内のアプリケーションからのリクエストに基づいて、前記リクエストに当該クライアント端末のユーザの認証情報を付加して、前記サーバへ送信するリクエスト送信部と、
を備える、請求項１に記載の情報処理装置。
【請求項３】
前記アプリケーションから受け取る前記リクエストには、前記サーバへ送信する優先順位を示す優先情報が含まれており、
前記リクエスト送信部は、前記優先情報に基づいて、前記リクエストを前記サーバへ送信する、請求項２に記載の情報処理装置。
【請求項４】
複数の前記アプリケーションから前記サーバに対する複数のリクエストを受け取ったとき、
複数の前記リクエストを合成して集約リクエストを生成するリクエスト合成部と、
前記集約リクエストに対する前記サーバからの集約レスポンスを、各前記リクエストに対応するレスポンスに分解するレスポンス分解部と、
を備える、請求項１～３のいずれか１項に記載の情報処理装置。
【請求項５】
前記リクエスト送信部は、前記優先情報より優先順位の低い前記リクエストは、前記リクエスト合成部により集約して前記サーバへ送信する、請求項４に記載の情報処理装置。
【請求項６】
前記アプリケーションからのリクエストに対する前記サーバからのレスポンスを保持するレスポンス蓄積部と、
前記レスポンス蓄積部に保持されたレスポンスを管理するキャッシュ判断部と、
を備え、
前記キャッシュ判断部は、前記アプリケーションから、前記レスポンス蓄積部に蓄積されたレスポンスを前記サーバから取得するリクエストを受けたとき、前記レスポンス蓄積部から前記レスポンスを取得して前記アプリケーションに送信する、請求項１～５のいずれか１項に記載の情報処理装置。
【請求項７】
前記レスポンス蓄積部は、蓄積された前記各レスポンスに対して当該レスポンスを保持する有効期限情報を記憶しており、
前記キャッシュ判断部は、前記有効期限情報に基づいて、前記アプリケーションからのリクエストに対するレスポンスを前記レスポンス蓄積部から取得するか否かを判断する、請求項６に記載の情報処理装置。
【請求項８】
少なくとも前記リクエスト端末の通信環境の良否、電池残量またはメモリ残量を監視する端末状態監視部をさらに備え、
前記リクエスト送信部は、前記端末状態監視部による前記リクエスト端末の監視結果に基づいて、前記リクエストを送信する、請求項１～７のいずれか１項に記載の情報処理装置。
【請求項９】
前記リクエスト送信部は、複数の前記アプリケーションから前記サーバへのリクエスト

に所定の文字列が含まれているとき、前記所定の文字列を当該文字列に対応する情報に変換して、前記サーバへ送信する、請求項１～８のいずれか１項に記載の情報処理装置。
【請求項１０】
前記認証情報記憶部には、複数ユーザの認証情報が記憶され、
前記リクエスト送信部は、複数の前記アプリケーションから前記リクエストを受け取ったとき、前記クライアント端末にログインしている前記ユーザの前記サーバ上の領域に対して、前記リクエストを送信する、請求項１～９のいずれか１項に記載の情報処理装置。
【請求項１１】
前記アプリケーションから前記リクエストを受け取った際に、前記アプリケーションおよび前記リクエストの内容に基づいて、当該リクエストの前記サーバへの送信の実行可否を判断する、実行判断部をさらに備える、請求項１～１０のいずれか１項に記載の情報処理装置。
【請求項１２】
ネットワークを介してデータを保持するサーバと接続され、クライアント端末内の１または複数のアプリケーションからのリクエストを受けて、前記サーバとの情報の送受信を制御する、情報処理方法。
【請求項１３】
コンピュータに、
ネットワークを介してデータを保持するサーバと接続されたクライアント端末に設けられる、前記クライアント端末内の１または複数のアプリケーションからのリクエストを受けて、前記サーバとの情報の送受信を制御する、情報処理装置として機能させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、情報処理装置、情報処理方法およびコンピュータ読み取り可能な記録媒体に関する。
【背景技術】
【０００２】
近年、モバイル情報端末において、パーソナルコンピュータのように、アプリケーションを追加してユーザ毎にカスタマイズすることの可能なシステムが注目されている。ここでいうアプリケーションには、第三ベンダが開発するアプリケーションも含まれる。また、個々のアプリケーションとサーバ側との親密性はより高まってきており、同一クライアントに、サーバを利用して処理を実行するアプリケーションが複数種類存在することも多い。
【０００３】
ユーザ毎に認証情報を必要とするサーバとアプリケーションとが連携する場合、アプリケーション毎にユーザに対して認証情報の入力を要求し、シングルサインインのためには、その認証情報を保持しておく必要がある。これは、ユーザにとって面倒な操作に他ならない。また、サーバの認証情報を保持する領域が増加することや、第三ベンダにサーバの認証情報が与えられることからセキュリティ面の問題も存在する。さらに、各アプリケーションが個々に同一サーバに対して通信を行うことは非効率であり、サーバの通信コストや負荷が増加したり、クライアント端末の電池寿命に影響を及ぼしたりすることも考えられる。
【０００４】
サーバの通信コストや負荷を軽減する方法として、例えば、特許文献１には、ネットワーク上にゲートウェイ装置を設け、各クライアントからのセッション制御用メッセージを集約して新しいメッセージを生成し、サーバに送信することで、セッションリソースを低減させる通信制御方法が開示されている。

【先行技術文献】
【特許文献】
【０００５】
【特許文献１】特開２００９－２１８６３１号公報
【発明の概要】
【発明が解決しようとする課題】
【０００６】
しかし、上記特許文献１のようにクライアントとサーバとの間にゲートウェイ装置を配置することは有効であるが、クライアントの各アプリケーションが個々にサーバと通信することには変わりなく、クライアントから送信される通信量は削減されないという問題があった。このようなクライアントからの通信量は、端末のメモリや電池、およびユーザの通信コスト等に影響することになる。
【０００７】
そこで、本発明は、上記問題に鑑みてなされたものであり、本発明の目的とするところは、クライアント‐サーバ間での情報の送受信を制御し、クライアント‐サーバ間の通信量を低減することが可能な、新規かつ改良された情報処理装置、情報処理方法およびコンピュータ読み取り可能な記録媒体を提供することにある。
【課題を解決するための手段】
【０００８】
上記課題を解決するために、本発明のある観点によれば、ネットワークを介してデータを保持するサーバと接続されたクライアント端末に設けられ、クライアント端末内の１または複数のアプリケーションからのリクエストを受けて、サーバとの情報の送受信を制御する、情報処理装置が提供される。
【０００９】
情報処理装置は、サーバにアクセスするためのユーザの認証情報を保持する認証情報記憶部と、クライアント端末内のアプリケーションからのリクエストに基づいて、リクエストに当該クライアント端末のユーザの認証情報を付加して、サーバへ送信するリクエスト送信部と、を備えることができる。
【００１０】
アプリケーションから受け取るリクエストには、サーバへ送信する優先順位を示す優先情報が含まれており、リクエスト送信部は、優先情報に基づいて、リクエストをサーバへ送信してもよい。
【００１１】
また、リクエスト送信部は、優先情報より優先順位の低いリクエストは、リクエスト合成部により集約してサーバへ送信してもよい。
【００１２】
情報処理装置は、複数のアプリケーションからサーバに対する複数のリクエストを受け取ったとき、複数のリクエストを合成して集約リクエストを生成するリクエスト合成部と、集約リクエストに対するサーバからの集約レスポンスを、各リクエストに対応するレスポンスに分解するレスポンス分解部と、を備えることもできる。
【００１３】
アプリケーションからのリクエストに対するサーバからのレスポンスを保持するレスポンス蓄積部と、レスポンス蓄積部に保持されたレスポンスを管理するキャッシュ判断部と、を備え、キャッシュ判断部は、アプリケーションから、レスポンス蓄積部に蓄積されたレスポンスをサーバから取得するリクエストを受けたとき、レスポンス蓄積部からレスポンスを取得してアプリケーションに送信することもできる。
【００１４】
レスポンス蓄積部は、蓄積された各レスポンスに対して当該レスポンスを保持する有効期限情報を記憶しており、キャッシュ判断部は、有効期限情報に基づいて、アプリケーションからのリクエストに対するレスポンスをレスポンス蓄積部から取得するか否かを判断

してもよい。
【００１５】
少なくともリクエスト端末の通信環境の良否、電池残量またはメモリ残量を監視する端末状態監視部をさらに備え、リクエスト送信部は、端末状態監視部によるリクエスト端末の監視結果に基づいて、リクエストを送信してもよい。
【００１６】
リクエスト送信部は、複数のアプリケーションからサーバへのリクエストに所定の文字列が含まれているとき、所定の文字列を当該文字列に対応する情報に変換して、サーバへ送信してもよい。
【００１７】
認証情報記憶部には、複数ユーザの認証情報が記憶され、リクエスト送信部は、複数のアプリケーションからリクエストを受け取ったとき、クライアント端末にログインしているユーザのサーバ上の領域に対して、リクエストを送信することもできる。
【００１８】
また、アプリケーションからリクエストを受け取った際に、アプリケーションおよびリクエストの内容に基づいて、当該リクエストのサーバへの送信の実行可否を判断する、実行判断部をさらに備えることもできる。
【００１９】
また、上記課題を解決するために、本発明の別の観点によれば、ネットワークを介してデータを保持するサーバと接続され、クライアント端末内の１または複数のアプリケーションからのリクエストを受けて、サーバとの情報の送受信を制御する、情報処理方法が提供される。
【００２０】
さらに、上記課題を解決するために、本発明の別の観点によれば、コンピュータに、ネットワークを介してデータを保持するサーバと接続されたクライアント端末に設けられる、クライアント端末内の１または複数のアプリケーションからのリクエストを受けて、サーバとの情報の送受信を制御する、情報処理装置として機能させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体が提供される。
【発明の効果】
【００２１】
以上説明したように本発明によれば、クライアント‐サーバ間での情報の送受信を制御し、クライアント‐サーバ間の通信量を低減することが可能な、情報処理装置、情報処理方法およびコンピュータ読み取り可能な記録媒体を提供することができる。
【図面の簡単な説明】
【００２２】
【図１】本発明の実施形態に係るシステムの概略構成を示す概念図である。
【図２】同実施形態にかかるクライアント‐サーバ構成の一例を示す説明図である。
【図３】同実施形態に係るＧＷアプリの機能構成を示すブロック図である。
【図４】リクエストＩＤ　ＤＢのテーブル構成例を示す説明図である。
【図５】キャッシュＤＢのテーブル構成例を示す説明図である。
【図６】同一リクエストＤＢのテーブル構成例を示す説明図である。
【図７】キューＤＢのテーブル構成例を示す説明図である。
【図８】認証情報ＤＢのテーブル構成例を示す説明図である。
【図９】アプリケーションからリクエストを受けたときの、当該アプリケーションとＧＷアプリとの間で行われる情報のやり取りを示すフローチャートである。
【図１０Ａ】アプリケーションからリクエストを受けたときの、ＧＷアプリとサーバとの間で行われる情報のやり取りを示すフローチャートである。
【図１０Ｂ】アプリケーションからリクエストを受けたときの、ＧＷアプリとサーバとの間で行われる情報のやり取りを示すフローチャートである。
【図１１】ＧＷアプリが複数のアプリケーションからリクエストを受けたときの、ＧＷア

プリの一動作例を示す説明図である。
【図１２】リクエストＸＭＬの一例を示す説明図である。
【図１３】リクエストＸＭＬの他の一例を示す説明図である。
【図１４】集約リクエストＸＭＬの一例を示す説明図である。
【図１５】レスポンスＸＭＬの一例を示す説明図である。
【図１６】レスポンスＸＭＬの他の一例を示す説明図である。
【図１７】集約レスポンスＸＭＬの一例を示す説明図である。
【図１８】キャッシュの許可情報が含まれるレスポンスＸＭＬの一例を示す説明図である。
【図１９】リクエストＩＤが含まれるレスポンスＸＭＬの一例を示す説明図である。
【図２０】多ユーザの認証情報を管理する場合の認証情報ＤＢのテーブル構成例を示す説明図である。
【図２１】複数ユーザがログインしている状態でリクエストＸＭＬが渡された時の動作例を示す説明図である。
【発明を実施するための形態】
【００２３】
以下に添付図面を参照しながら、本発明の好適な実施の形態について詳細に説明する。なお、本明細書及び図面において、実質的に同一の機能構成を有する構成要素については、同一の符号を付することにより重複説明を省略する。
【００２４】
なお、説明は以下の順序で行うものとする。
１．システム構成例
２．機能構成
３．クライアント‐サーバ間の通信処理
【００２５】
＜１．システム構成例＞
まず、図１および図２を参照して、本発明の実施形態に係るシステムの概略構成について説明する。なお、図１は、本実施形態に係るシステムの概略構成を示す概念図である。図２は、本実施形態に係るクライアント‐サーバ構成の一例を示す説明図である。
【００２６】
本実施形態に係るシステムは、１または複数のクライアント端末１と、各クライアント端末１へデータを提供するサーバ２とが、ネットワーク１０を介して接続されている。例えば、図１に示すように、４つのクライアント端末１Ａ～１Ｄとサーバ２とが接続されたデータ同期システムを構築することができる。クライアント端末１とサーバ２との通信は、無線通信であってもよく、有線通信であってもよい。また、クライアント端末１Ａ～１Ｄのユーザは、それぞれ異なるユーザであってもよく、同一ユーザであってもよい。
【００２７】
クライアント端末１は、サーバ２から所望のデータを取得可能な情報処理端末であって、例えば、携帯電話やＰＤＡ（Ｐｅｒｓｏｎａｌ Ｄｉｇｉｔａｌ Ａｓｓｉｓｔａｎｔｓ）等の通信端末や、コンピュータ、テレビジョン受像機等である。クライアント端末１は、サーバ２から取得したデータを用いて、アプリケーションの機能を実現する。サーバ２は、データを保持し、クライアント１からの要求によりデータを提供する。
【００２８】
本実施形態のクライアント端末１は、ユーザ毎に、アプリケーションを自由に追加して、カスタマイズ可能な端末を想定している。これらのアプリケーションには、サーバと通信することによりその機能を実現するものもある。このようなアプリケーションは、サーバが提供するＡＰＩ（Ａｐｐｌｉｃａｔｉｏｎ Ｐｒｏｇｒａｍ Ｉｎｔｅｒｆａｃｅ）を用いることによって、サーバ２を提供しているベンダのみならず第三ベンダによっても開発することができるものである。
【００２９】

本実施形態に係るクライアント端末１とサーバ２との間で行われる通信は、図２に示すように行われる。例えば、ユーザＢのクライアント端末１Ｂに、アプリＡ、アプリＢ、アプリＣ、アプリＤ、・・・のような複数のアプリケーションがインストールされているとする。各アプリケーションは、サーバ２の機能をネットワーク１０経由で利用することで、クライアント端末１Ｂにその機能を提供することができる。このため、アプリケーションの機能を実現させるためには、サーバ２と通信して、必要な情報の提供を受ける必要がある。

【００３０】

サーバ２は、登録ユーザ毎に異なるサービスを提供するサーバである。ユーザがサーバ２より当該サービスを受けるためには、ユーザアカウントによる認証が必要である。ここで、本実施形態に係るクライアント端末１Ｂは、アプリケーションによるサーバ２へのアクセスを代替して行うゲートウェイアプリケーション（以下、「ＧＷアプリ」と称する。）１００を備える。ＧＷアプリ１００は、アプリＡ～Ｄの代わりにサーバ２へアクセスするための認証情報を認証情報ＤＢ１４６に保持している。例えば、図２に示すユーザＢによって使用されるクライアント端末１Ｂは、ユーザＢの認証情報をＧＷアプリ１００の認証情報ＤＢ１４６に保持している。

【００３１】

アプリケーションよりサーバ２への代替アクセス要求を受けたＧＷアプリ１００は、ユーザＢの認証情報をサーバ２へ送信する。そして、サーバ２により認証されると、サーバ２がユーザＢに対して提供する情報を保持するユーザ領域Ｂへのアクセスが可能となる。このような状態となってアプリケーションはサーバ２から機能の実行に必要な情報の提供を受けることができる。

【００３２】

本実施形態では、クライアント‐サーバ間の通信の際に必要となる認証情報をクライアント端末１のＧＷアプリ１００にて保持し、各アプリケーションからサーバ２へのアクセス処理をＧＷアプリ１００が代替して実行する。また、ＧＷアプリ１００は、クライアント‐サーバ間の通信を効率よく行うため、サーバ２との情報の送受信を制御することも行うことができる。これにより、クライアント‐サーバ間の通信量を低減することができ、認証情報が外部へ漏れるのを防止することができる。以下、クライアント端末１に設けられるＧＷアプリ１００の構成とこれを用いたクライアント‐サーバ間の通信処理について説明していく。

【００３３】

＜２．機能構成＞

まず、図３に基づいて、本実施形態に係るＧＷアプリ１００の機能構成について説明する。なお、図３は、本実施形態に係るＧＷアプリ１００の機能構成を示すブロック図である。

【００３４】

ＧＷアプリ１００は、上述したように、クライアント端末１にインストールされたアプリケーションとサーバ２とのアクセス処理を代替する。ＧＷアプリ１００は、図３に示すように、アプリ入出力インタフェース部１１０と、データ解析部１２０と、リクエスト／レスポンス変換部１３０と、サーバ送受信インタフェース部１４０とを備える。

【００３５】

アプリ入出力インタフェース部１１０は、各アプリケーションとＧＷアプリ１００との間で情報を送受信するためのインタフェースであって、リクエスト入力部１１２と、レスポンス出力部１１４とからなる。リクエスト入力部１１２は、アプリケーションから受けたサーバ２へ対するアクセス要求を、データ解析部１２０へ出力する。レスポンス出力部１１４は、データ解析部１２０から入力されたサーバ２からの応答を、各アプリケーションへ通知する。

【００３６】

データ解析部１２０は、各アプリケーションとサーバ２との間で通信される情報を生成

し、処理する。データ解析部１２０は、アプリキー認証部１２１と、リクエストＩＤ変換部１２２と、キャッシュ判断部１２３と、同一リクエスト保管部１２４と、送信リクエスト保管部１２５と、端末状態監視部１２６とを備える。

【００３７】

アプリキー認証部１２１は、各アプリケーションからのリクエストの通信の可否を判断する。各アプリケーションは、サーバ２へのリクエストを送信する際、リクエストとともにアプリキーをＧＷアプリ１００へ送信する。アプリキーは、サーバ２の開発元から発行される情報であり、審査を経て、アプリキーおよび使用可能なＡＰＩレベルを示すデータが各アプリケーションに発行されている。このデータは、サーバ２の開発元によって認証子が付加されており、サーバ２の開発元以外から当該データの内容を取得することはできないものとなっている。

【００３８】

一方、ＧＷアプリ１００は、かかるデータの内容を取得することができ、各アプリケーションから送信されたアプリキーにサーバ２に対してリクエストする権限があるか否かを、アプリキーＤＢを参照して確認することができる。アプリキーＤＢは、サーバ２によって更新され、通信を許可もしくは禁止されるアプリケーションのアプリキーのリストを保持している。アプリキー認証部１２１は、アプリキーＤＢに記憶されている上方に基づき、データからセキュリティレベルを確認し、アプリケーションからのリクエストの内容が呼び出し元のアプリケーションのセキュリティレベルで送信可能か否かを判定する。そして、送信可能と判定した場合、アプリキー認証部１２１は、送信リクエスト保管部１２５にアプリケーションからのリクエストを送信する。一方、送信不可と判定した場合には、リクエストの送信を拒絶する内容を、アプリ入出力インタフェース部１１０を介して、呼び出し元のアプリケーションへ通知する。

【００３９】

リクエストＩＤ変換部１２２は、クライアント‐サーバ間で頻繁に送信されるリクエストを表すリクエストＸＭＬと、リクエストＸＭＬに固有のリクエストＩＤとの変換処理を行う。リクエストＩＤ変換部１２２は、このようなリクエストＸＭＬとリクエストＩＤとを、リクエストＩＤ　ＤＢに関連付けて記憶している。リクエストＩＤ　ＤＢは、例えば、図４に示すように、リクエストＸＭＬ（またはＩＤ）１２２１と、リクエストＩＤ１２２２とを保持するテーブル構成とすることができる。

【００４０】

リクエストＩＤ変換部１２２は、アプリケーションから、クライアント‐サーバ間で頻繁に送信されるリクエストの送信を受けた場合、リクエストＩＤに変換してサーバ２へ送信する。これにより、リクエストＸＭＬと比べてデータ量の小さいデータに変換してサーバ２へ送信するため、通信負荷を低減することができる。また、サーバ２からのレスポンスにリクエストＩＤが含まれている場合、リクエストＩＤ変換部１２２は、リクエストＩＤ　ＤＢを参照して、リクエストＩＤを、当該リクエストＩＤに対応するリクエストＸＭＬに変換する。

【００４１】

キャッシュ判断部１２３は、サーバ２からのレスポンスにキャッシュの許可情報が含まれているか否かを判断する。キャッシュ判断部１２３は、レスポンスにキャッシュの許可情報が含まれていると判断すると、許可されたリクエストＸＭＬ、レスポンスＸＭＬおよびそのキャッシュの有効期限をキャッシュＤＢに記録する。キャッシュＤＢは、例えば図５に示すように、許可されたリクエストＸＭＬ（またはＩＤ）１２３１、レスポンスＸＭＬ１２３２およびそのキャッシュの有効期限１２３３を保持するテーブル構成とすることができる。このように、サーバ２からのレスポンス情報をキャッシュＤＢに蓄積することにより、アプリケーションから同じリクエストを受けたときに、キャッシュＤＢを参照してレスポンスすることが可能となる。これにより、サーバ２との通信を省略することができ、通信負荷を軽減することができる。

【００４２】

同一リクエスト保管部１２４は、ＧＷアプリ１００のキューＤＢに既に格納されているリクエストをアプリケーションから受けた場合に、当該リクエストを要求したアプリケーションを特定するための情報を保持して管理する。同一リクエスト保管部１２４は、リクエストの呼び出し元であるアプリケーションのアプリＩＤと、当該リクエストと同一のリクエストの送信について現在キューＤＢに記憶されている情報のキューインデックスとを関連付けて同一リクエストＤＢに記憶する。同一リクエストＤＢは、例えば図６に示すように、キューインデックス１２４１と、アプリＩＤ１２４２とを保持するテーブル構成とすることができる。このように、同一のリクエストをサーバ２に対して行うアプリケーションを管理することで、同一のリクエストを重複してサーバ２に送信することを防止でき、通信負荷を軽減することができる。

【００４３】

送信リクエスト保管部１２５は、サーバ２へ送信するリクエストを保持して管理する。送信リクエスト保管部１２５は、アプリキー認証部１２１にて送信可能と判定されたリクエストを、キューＤＢに記憶する。キューＤＢは、例えば図７に示すように、キューインデックス１２５１と、アプリＩＤ１２５２と、優先度１２５３と、リクエストＸＭＬ１２５４とを保持するテーブル構成とすることができる。キューインデックス１２５１は、キューＤＢに記録されたリクエストの順序を示す番号であり、アプリＩＤ１２５２は、リクエストの呼び出し元のアプリケーションを特定する、アプリケーションに固有のＩＤである。

【００４４】

優先度１２５３は、リクエストを実行させるべき優先順位を示す値であり、値が大きいほど早急に実行することが要求されるリクエストであることを示している。また、優先度が０以下である場合は、リアルタイム性が要求されていないことを示している。例えば、図７のキューインデックス１２５１が「１」、「２」、「３」のリクエストは、リアルタイム性を要求するものではない。優先度は、各アプリケーションがリクエストを要求するときに合わせて設定するようにしてもよく、あるいはアプリケーションによって予め設定されていてもよく、実行したいＡＰＩに応じて決定されるものであってもよい。キューＤＢに記憶されたリクエストは、端末状態監視部１２６の判断に基づき、サーバ２へ順次送信される。

【００４５】

端末状態監視部１２６は、クライアント端末１の通信環境を確認する。端末状態監視部１２６は、例えば、クライアント端末１の電池の残量や、メモリ使用量、電波状態を確認し、サーバ２へ送信可能なデータ量を判断する。そして、端末状態監視部１２６は、クライアント端末１の通信環境に応じて、サーバ２へ送信するデータ量を決定する。端末状態監視部１２６により決定されたデータ量内のリクエストがキューＤＢから選択され、サーバ２へ送信される。

【００４６】

リクエスト／レスポンス変換部１３０は、データ解析部１２０にて作成されたリクエストおよびサーバ２から受信したレスポンスの形式を変換する機能部であって、リクエスト合成部１３２と、レスポンス分解部１３４とを備える。リクエスト合成部１３２とは、データ解析部１２０から入力されたリクエストをマージし、サーバ送受信部インタフェース部１４０へ出力する。レスポンス分解部１３４は、サーバ送受信部インタフェース部１４０を介して受信したサーバ２のレスポンスを分解し、データ解析部１２０へ出力する。

【００４７】

サーバ送受信インタフェース部１４０は、サーバ２とＧＷアプリ１００との間で情報を送受信するためのインタフェースであって、リクエスト送信部１４２と、レスポンス受信部１４４とからなる。リクエスト送信部１４２は、リクエスト／レスポンス変換部１３０から入力されたリクエストをサーバ２へ送信する。また、リクエスト送信部１４２は、ユーザの認証情報を記憶する認証情報ＤＢ１４６を保持している。レスポンス受信部１４４は、サーバ２から受信したレスポンスを、リクエスト／レスポンス変換部１３０へ出力す

る。
【００４８】
サーバ送受信インタフェース部１４０に設けられた認証情報ＤＢ１４６は、例えば図８に示すように、ユーザＩＤ１４６１と、クライアントＩＤ１４６２と、パスワード１４６３と、サーバＵＲＬ１４６４とを保持するテーブル構成とすることができる。ユーザＩＤ１４６１は、ユーザを特定する、ユーザに固有のＩＤであり、クライアントＩＤ１４６２は、サーバ２によって決定される、クライアント端末１を特定するＩＤである。また、パスワード１４６３は、サーバ２によって決定され、クライアントＩＤと関連付けられた情報であり、サーバＵＲＬ１４６４は、当該クライアント端末１からサーバ２上の自身の領域に対してＡＰＩを実行するためのＵＲＬである。認証情報ＤＢ１４６に格納された情報は、ＧＷアプリ１００によって一元管理され、ＧＷアプリ以外のクライアント端末１の各アプリケーションからは参照できない形式で保持されている。
【００４９】
このように、各アプリケーションの認証情報を、アプリケーション自身ではなくＧＷアプリ１００に保持させることにより、各アプリケーションは、ＧＷアプリ１００を介さなければサーバ２へアクセスすることができないようになっている。
【００５０】
以上、本実施形態に係るＧＷアプリ１００の機能構成について説明した。
【００５１】
＜３．クライアント‐サーバ間の通信処理＞
次に、図９～図１１に基づいて、本実施絵形態に係るＧＷアプリ１００を用いたクライアント‐サーバ間の通信処理について説明する。なお、図９は、アプリケーションからリクエストを受けたときの、当該アプリケーションとＧＷアプリ１００との間で行われる情報のやり取りを示すフローチャートである。図１０Ａおよび図１０Ｂは、アプリケーションからリクエストを受けたときの、ＧＷアプリ１００とサーバ２との間で行われる情報のやり取りを示すフローチャートである。図１１は、ＧＷアプリ１００が複数のアプリケーションからリクエストを受けたときの、ＧＷアプリ１００の一動作例を示す説明図である。
【００５２】
［アプリケーションとＧＷアプリとの間で行われる情報処理］
これらによって示されるクライアント‐サーバ間の通信処理は、クライアント端末１にインストールされたアプリケーションが、サーバ２と連携して機能を提供するために実行される処理である。当該処理は、まず、図９に示すように、サーバ２のＡＰＩを実行するために、各アプリケーションがサーバ２へのリクエストをＧＷアプリ１００へ送信することから開始される。ＧＷアプリ１００は、各アプリケーションからリクエストを受け取る（ステップＳ１００）。サーバ２のＡＰＩは、クライアントＵＲＬに、ヘッダとしてクライアントＩＤおよびパスワードをセットして、利用したいサーバ２のＡＰＩ名とその引数とで構成される、ＸＭＬ形式のテキスト（リクエストＸＭＬ）を受信することで実行される。かかるリクエストＸＭＬは、各アプリケーションで作成され、例えば図１２、図１３に示すように記述することができる。
【００５３】
例えば、図１２に示すリクエストＸＭＬでは、実行させるＡＰＩ名「アプリ１」と引数ｘとを通知している。また、図１３に示すリクエストＸＭＬでは、実行させるＡＰＩ名「アプリ２」と引数ｙ、ｚとを通知している。なお、リクエストは、このようなＸＭＬ形式のテキストのみでなく、その他のデータ交換フォーマットを利用することもできる。
【００５４】
ステップＳ１００の処理は、例えば図１１に示す例では、クライアント端末（例えば、ユーザＢのクライアント端末１Ｂ）のアプリケーション群のアプリケーションから、リクエストＸＭＬがＧＷアプリ１００へ送信される処理に対応する。リクエストＸＭＬを受信したＧＷアプリ１００のリクエスト入力部１１２は、当該リクエストＸＭＬをアプリキー

認証部１２１へ送信する。

【００５５】

次いで、アプリキー認証部１２１は、受信したリクエストＸＭＬをサーバ２へ送信してもよいか否かを判定する（ステップＳ１０２）。アプリキー認証部１２１は、呼び出し元にアプリケーションから受信したアプリキーおよびリクエストの内容に基づいて、リクエストの通信の可否を判定する。アプリキー認証部１２１は、送信要求を受けたリクエストが、必要とするセキュリティレベルに満たないアプリケーションから呼び出された場合には、サーバ２への送信は不可と判定し、呼び出し元のアプリケーションへセキュリティエラーを通知する（ステップＳ１０４）。

【００５６】

一方、アプリケーションから受信したデータのセキュリティレベルを確認し、リクエストの内容より呼び出し元のアプリケーションのセキュリティレベルで送信可能と判定した場合には、アプリキー認証部１２１は、当該リクエストの内容（リクエストＸＭＬ）がキャッシュＤＢに記憶されているか否かを、キャッシュ判断部１２３に判定させる。すなわち、キャッシュ判断部１２３は、アプリキー認証部１２１から通知されたリクエストＸＭＬでキャッシュＤＢをクエリし（ステップＳ１０６）、当該リクエストＸＭＬのレスポンス情報が記憶されているか否かを確認する（ステップＳ１０８）。

【００５７】

ステップＳ１０８にて、キャッシュＤＢに当該リクエストＸＭＬのレスポンス情報が記憶されていると判断した場合、キャッシュ判断部１２３は、当該リクエストＸＭＬに対応するキャッシュの有効期限を確認し、有効期限が過ぎていないか否かを判定する（ステップＳ１１０）。ステップＳ１１０において、キャッシュの有効期限が過ぎていないと判定された場合には、キャッシュ判断部１２３は、呼び出し元のアプリケーションに対して当該リクエストＸＭＬに対するレスポンス情報（レスポンスＸＭＬ）を、レスポンス出力部１１４を介して送信する（ステップＳ１１２）。

【００５８】

一方、ステップＳ１１０にて、キャッシュの有効期限が過ぎている場合には、サーバ２へ再度アクセスし、情報を取得する必要がある。この場合、まず、キャッシュ判断部１２３は、キャッシュＤＢから当該リクエストＸＭＬに対応するキャッシュ情報を削除する（ステップＳ１１４）。そして、ステップＳ１０６において当該リクエストＸＭＬでリクエストＩＤ変換部１２２のリクエストＩＤ　ＤＢをクエリした結果に基づいて、リクエストＩＤ変換部１２２は、当該リクエストＸＭＬと合致するリクエストＸＭＬが記憶されているか否かを判定する（ステップＳ１１６）。

【００５９】

リクエストＩＤ　ＤＢに合致するリクエストＸＭＬが記憶されていると判定した場合には、リクエストＩＤ変換部１２２は、当該リクエストＸＭＬをリクエストＩＤに変換する（ステップＳ１１８）一方、リクエストＩＤ　ＤＢに合致するリクエストＸＭＬが記憶されていないと判定した場合には、リクエストＩＤへの変換は行われず、ステップＳ１２０の処理へ進む。ステップＳ１２０では、送信リクエスト保管部１２５にて、当該リクエストＸＭＬでキューＤＢをクエリする処理が行われる。これにより、同一のリクエストが既にキューＤＢに記憶されているかいないを判定する。

【００６０】

ステップＳ１２０にて当該リクエストＸＭＬと同一の当該リクエストＸＭＬがキューＤＢに記憶されていると判定した場合、送信リクエスト保管部１２５は、同一リクエスト保管部１２４に対して、同一リクエストＤＢに、当該リクエストＸＭＬの送信処理を示すキューインデックスと、当該リクエストＸＭＬの読み出し元のアプリＩＤを、同一リクエストＤＢに記録させる（ステップＳ１２２）。一方、ステップＳ１２０にて当該リクエストＸＭＬと同一の当該リクエストＸＭＬがキューＤＢに記憶されていないと判定した場合、送信リクエスト保管部１２５は、キューＤＢに、当該リクエスト、優先度およびアプリＩＤを記録する（ステップＳ１２４）。

【００６１】
以上、アプリケーションからリクエストを受けたときの、当該アプリケーションとＧＷアプリ１００との間で行われる情報のやり取りについて説明した。
【００６２】
［サーバとＧＷアプリとの間で行われる情報処理］
次に、図１０Ａおよび図１０Ｂに基づいて、ＧＷアプリ１００とサーバ２との間で行われる情報のやり取りについて説明する。まず、ＧＷアプリ１００は、サーバ２に送信するリクエストを保持するキューＤＢに対して、優先度が０より大きいリクエストをクエリして特定する（ステップＳ２００）。ステップＳ２００は、リアルタイム性の要求されるリクエストを抽出し、他のリクエストに優先して送信するために行われる。そして、優先度が０より大きいリクエストが存在するか否かを判定し（ステップＳ２０２）、存在する場合には、端末状態監視部１２６によりクライアント端末１の通信環境の状態が良好であるか否かを確認する（ステップＳ２０４）。
【００６３】
ステップＳ２０４では、通信環境の状態として、電波環境について確認する。電波環境が良好である場合には、送信データ量が多くても通信エラーが発生する確率は低い。したがって、電波環境が良好である場合には、キューＤＢから最大ｙ個のリクエストを取得して（ステップＳ２１０）、サーバ２への送信対象とする。一方、電波環境が良好ではない場合には、優先度の高いリクエストのみをサーバ２への送信対象とする。
【００６４】
ステップＳ２０２に戻り、優先度の高いリクエストが抽出されなかった場合にも、端末状態監視部１２６によりクライアント端末１の通信環境の状態が良好であるか否かを確認する（ステップＳ２０６）。この場合、端末状態監視部１２６は、通信環境の状態として、メモリ使用量や、電池残量、電波環境を確認する。メモリの空き容量が少ない場合に通信を行うとユーザがクライアント端末１を操作する際の操作感が悪化してしまう。また、電池残量が十分でない場合は、優先順位の低い通信を避け、電池を節約することが望ましい。そこで、優先度の低いリクエストについては、クライアント端末１の操作感を維持することや電池を節約することを優先させて、サーバ２へは送信せず、処理を終了する。一方、通信環境が良好である場合には、キューＤＢに記憶されているリクエストの数と、リクエストの送信を開始する蓄積数（閾値ｘ）とを比較して、リクエストの送信を開始するか否かを判定する（ステップＳ２０８）。
【００６５】
リクエストは、通常、複数まとめて送信される。このため、キューＤＢに蓄積されたリクエストの数が所定数（閾値ｘ）を超えたときに、サーバ２へのリクエストの送信処理が開始される。すなわち、ステップＳ２０８では、リクエストの送信を開始する数のリクエストがキューＤＢに蓄積されているか否かを確認している。ステップＳ２０８において、キューＤＢに蓄積されたリクエストの数が閾値ｘ以下である場合には、サーバ２へのリクエストの送信は行わず、処理を終了する。一方、キューＤＢに蓄積されたリクエストの数が閾値ｘを超えた場合には、キューＤＢから最大ｙ個のリクエストを取得して送信対象とする（ステップＳ２１０）。
【００６６】
ステップＳ２１０において、キューＤＢから取得する最大のリクエスト数ｙは、後述するステップＳ２１２において、マージする最大のリクエスト数となる。送信リクエスト保管部１２５は、キューＤＢに格納されているすべてのリクエストからｙ個のリクエストを取得して、リクエスト合成部１３２へ出力する。
【００６７】
なお、ステップＳ２０８の閾値ｘおよびステップＳ２１０のリクエスト数ｙは、ユーザの設定、もしくは端末の状態に応じて、適宜変更可能である。これにより、通信環境やクライアント端末１、サーバ２の状態に応じて通信状態を最適化することができる。一方、電波状態が悪い場合や、優先度の高いリクエストのデータ量が多い場合には、優先度の高

いリクエストの通信エラーの確率を低くするために、優先度の低いリクエストを通信対象から除外するようにしてもよい。

【００６８】

リクエスト合成部１３２は、送信リクエスト保管部１２４からキューＤＢのリクエストを受けると、受け取ったリクエストが複数であるか否かを確認する。そして、リクエストが複数である場合には、これらをまとめて新しいリクエストを生成する（ステップＳ２１２）。サーバ２へ送信されるリクエストが１つである場合には、例えば、図１２、図１３に示したリクエストＸＭＬがそのまま、あるいはリクエストＩＤに変換されて、サーバ２へ送信される。一方、サーバ２へ送信されるリクエストが複数ある場合には、ＸＭＬの階層構造を利用して、複数のＡＰＩを１つのリクエストＸＭＬに記述して、一回の通信でサーバに通信することができる。

【００６９】

例えば、図１２および図１３に示す２つのリクエストＸＭＬをサーバ２へ送信するとする。このとき、２つのリクエストＸＭＬは、例えば図１４に示すように、図１２のリクエストＸＭＬを１番目に実行されるＡＰＩとして、図１３のリクエストＸＭＬを２番目に実行されるＡＰＩとして、記述することが可能である。これにより、サーバ２との通信回数を低減することが可能となる。リクエスト合成部１３２は、サーバ２へ複数のリクエストを送信する場合にはこれらをまとめたリクエストＸＭＬ（集約リクエストＸＭＬ）を作成して、リクエスト送信部１４２へ出力する。そして、リクエスト送信部１４２は、リクエストをサーバ２へ送信（ＰＯＳＴ）する（ステップＳ２１４）。

【００７０】

その後、サーバ２はリクエストを受け取ると、リクエストＸＭＬより実行すべきコマンドを解釈し、クライアントＵＲＬが指し示すユーザ領域に対してコマンドを実行する。そして、サーバ２は、この実行結果を示すＸＭＬ形式のテキストをレスポンスＸＭＬとして、クライアント端末１のＧＷアプリ１００のレスポンス受信部１４４に送信する。レスポンス受信部１４４は、レスポンスＸＭＬをレスポンス分解部１３４へ出力する。そして、レスポンスＸＭＬに複数のリクエストＸＭＬに対する応答が含まれている場合、レスポンス分解部１３４は、レスポンスＸＭＬを複数のレスポンスに分解する（ステップＳ２１６）。

【００７１】

ここで、サーバ２から送信されるレスポンスＸＭＬは、例えば図１５、図１６に示すように記述されている。図１５に示すレスポンスＸＭＬは、図１２のリクエストＸＭＬに対する応答であり、図１６に示すレスポンスＸＭＬは、図１３のリクエストＸＭＬに対する応答である。レスポンスＸＭＬには、アプリケーションから要求されたＡＰＩ実行の成否と、実行結果が含まれる。また、図１４に示すように、複数のリクエストＸＭＬを合成した集約リクエストＸＭＬがサーバ２へ送信された場合には、例えば図１７に示すように、各リクエストＸＭＬに対する応答を含むレスポンスＸＭＬを受け取る。図１７のレスポンスＸＭＬには、１番目のＡＰＩ実行の成否とその実行結果、および２番目のＡＰＩ実行の成否とその実行結果が含まれる。このように、レスポンスＸＭＬが複数のリクエストＸＭＬに対する応答を含む場合には、レスポンス分解部１３４により、各リクエストＸＭＬに対する個々のレスポンスＸＭＬに分解される。分解されたレスポンスＸＭＬは、データ解析部１２０へ出力される。

【００７２】

サーバ２から受け取ったレスポンスＸＭＬには、ＡＰＩの実行結果の他に、例えば、当該リクエストに対するレスポンスのキャッシュ可否や、リクエストＸＭＬに対するリクエストＩＤの付与の要否等の情報が含まれる。データ解析部１２０は、これらの情報を解析して、次にアプリケーションからリクエストを受けたときのサーバ２の通信負荷を低減するための処理を実行する。

【００７３】

まず、データ解析部１２０のキャッシュ判断部１２３は、リクエストに対するレスポン

スのキャッシュの可否を確認する（ステップＳ２１８）。キャッシュ判断部１２３は、レスポンスＸＭＬにキャッシュを許可する許可情報が含まれているかを確認し、許可情報が含まれていれば当該レスポンスＸＭＬをキャッシュし、許可情報が含まれていなければ当該レスポンスＸＭＬはキャッシュしない。キャッシュの許可情報が含まれているレスポンスＸＭＬの一例を図１８に示す。図１８に示すように、キャッシュの許可情報は、例えばキャッシュの有効期限であり、この有効期限がレスポンスＸＭＬに含まれている場合には、キャッシュが許可されていることを意味する。

【００７４】

このように、キャッシュの許可情報を含むレスポンスＸＭＬを受け取ったキャッシュ判断部１２３は、キャッシュすることを許可されたリクエストＸＭＬ、レスポンスＸＭＬ、そのキャッシュの有効期限をキャッシュＤＢ（図５参照）に記録する（ステップＳ２２０）。このとき、レスポンスＸＭＬを要求したリクエストが、ＸＭＬ形式で記述されたものではなく、リクエストＩＤによって与えられたものであった場合には、リクエストＩＤ　ＤＢ（図４）を参照して、リクエストＩＤをリクエストＸＭＬに変換してからキャッシュＤＢに記録する。なお、ステップＳ２１８において、レスポンスＸＭＬにキャッシュの許可情報が含まれていない場合には、キャッシュすることは許可されていないため、このリクエストＸＭＬに対するレスポンスＸＭＬはキャッシュＤＢに記録されない。

【００７５】

次いで、ステップＳ２１８と同様に、リクエストＩＤ変換部１２２により、サーバ２からのレスポンスＸＭＬにリクエストＩＤが含まれているか否かを確認する（ステップＳ２２２）。リクエストＩＤは、サーバ２との通信負荷を低減するために、アプリケーションとサーバ２との間で頻繁にやり取りされるリクエストＸＭＬの代わりに用いられる情報である。リクエストＩＤ変換部１２２は、サーバ２からのレスポンスＸＭＬにリクエストＩＤが含まれていることを確認すると、当該リクエストＩＤと、そのリクエストＩＤに対応するリクエストＸＭＬをリクエストＩＤ　ＤＢ（図４参照）に記録する（ステップＳ２２４）。

【００７６】

リクエストＩＤを含むレスポンスＸＭＬの一例を図１９に示す。図１９に示すように、リクエストＩＤは、サーバ２から指定される情報として、レスポンスＸＭＬ内に記述されている。リクエストＩＤ変換部１２２は、このようにレスポンスＸＭＬに含まれるリクエストＩＤをリクエストＩＤ　ＤＢに記録することにより、その後アプリケーションから同一のリクエストを受けた場合には、リクエストＩＤに変換してサーバ２へ送信することができる。なお、ステップＳ２２２において、レスポンスＸＭＬにリクエストＩＤが含まれていない場合には、このリクエストＸＭＬに対するリクエストＩＤはなく、リクエストＩＤ　ＤＢにも記録されない。

【００７７】

さらに、同一リクエスト保管部１２４は、当該リクエストＸＭＬと同一のリクエストの送信を要求した他のクライアント端末１が存在するか否かを確認する（ステップＳ２２６）。同一リクエスト保管部１２４は、同一リクエストＤＢを参照して、当該リクエストＸＭＬの送信処理と関連付けられたキューインデックスが同一リクエストＤＢに存在するか否かを確認する。そして、同一のキューインデックスが存在する場合には、当該キューインデックスを関連付けられたアプリＩＤを、レスポンスＸＭＬの出力先に含める（ステップＳ２２８）。そして、同一リクエスト保管部１２４は、同一リクエストＤＢにより、実行されたインデックスを削除する（ステップＳ２３０）。

【００７８】

なお、ステップＳ２２６にて、同一リクエスト保管部１２４により、当該リクエストＸＭＬと同一のリクエストの送信を要求した他のクライアント端末１が存在しないと判定された場合には、ステップＳ２２８、Ｓ２３０の処理は行われず、ステップＳ２３２の処理が実行される。

【００７９】

ステップＳ２１８～Ｓ２３０までの、レスポンスＸＭＬの解析が終了すると、データ解析部１２０は、レスポンス出力部１１４を介して呼び出し元のアプリケーションにレスポンスＸＭＬを送信する（ステップＳ２３２）。その後、送信リクエスト保管部１２５は、実行されたキューインデックスとこれに関連付けられた情報をキューＤＢから削除する（ステップＳ２３４）。

【００８０】

以上、図９から図１１に示したように、クライアント‐サーバ間の通信が行われる。このように、ユーザ毎に認証情報を必要とするサーバのＡＰＩを利用して機能を提供する、クライアントにインストールされたアプリケーションは、従来ではアプリケーション毎に、ユーザに対し認証情報の入力を要求し、その認証情報を各々のアプリケーションが保持しておく必要があった。これは、ユーザにとって面倒な操作を与えるだけでなく、サーバの認証情報を保持する箇所が多くなることや、第三ベンダにユーザはサーバの認証情報をあたえることから、セキュリティ面の問題が存在した。また、各アプリケーションが個々に同一のサーバに対して通信を行うため非効率であり、サーバの通信コストおよび負荷の増加、クライアントの電池寿命に影響していた。

【００８１】

そこで、本実施形態のＧＷアプリを利用することにより、サーバは第三ベンダにユーザの認証情報を漏らすとこなく、ＡＰＩを提供することができる。また、クライアント端末内でリクエストの内容をまとめることで、セッションリソースを節約することができ、サーバの通信量を減らすことができる。ユーザにとっては認証情報を守れるほか、対応アプリ毎にログインする手間が省け、また端末の電池やメモリに応じた動作をするため快適に各アプリを扱うことが可能になる。クライアントからの通信量が減ることにより、ユーザは通信料の負担も軽減される。アプリを作成する第三ベンダにとってはサーバとの通信や認証情報を保持する部分の検討が不要になり、開発を容易に行うことができるようになる。

【００８２】

なお、リクエストをキューＤＢから取得してサーバ２との通信を行う処理は、通常、マルチスレッドによって並列に処理される。ＧＷアプリ１００が大量のリクエストを複数まとめてサーバ２へ送信している際に優先度の高いリクエストがキューに入った場合に、シングルスレッドでは対応できないためである。なお、スレッド数が多いほどパフォーマンスは向上するが、メモリや電池を消費するため、ＧＷアプリ１００は、スレッドの数をクライアント端末１の空きメモリ残量、電池の使用量に応じて動的に変化させることができる。

【００８３】

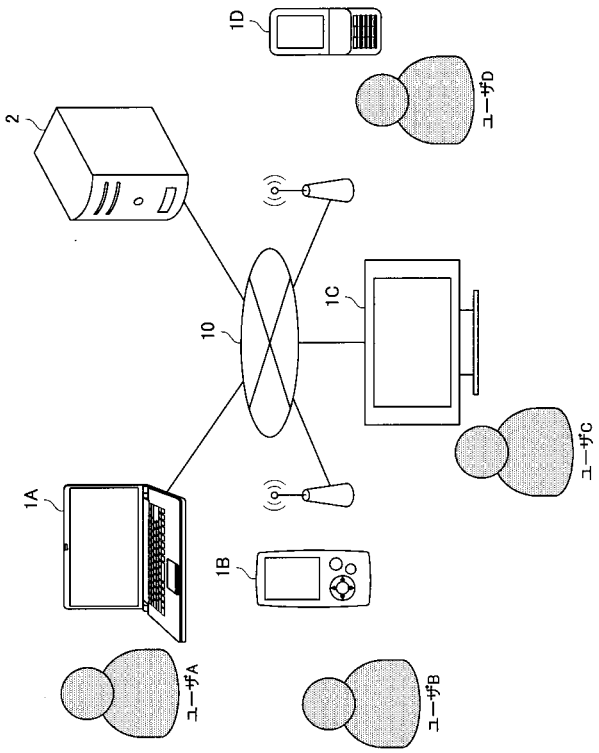
ＧＷアプリ１００の認証情報ＤＢに格納されたデータは、ＧＷアプリ１００以外からの参照はできないが、各アプリケーションは、そのデータをリクエストＸＭＬの中に入力したい場合がある。このため、ＧＷアプリ１００は各アプリケーションに、ＧＷアプリ１００内で認証情報ＤＢ１４６のデータと自動変換するための、ある決まった文字列を提供する。各アプリケーションは、リクエストＸＭＬに含めたい認証情報ＤＢ１４６内のデータがある場合に、その決まった文字列を該当箇所に記述しておく。ＧＷアプリ１００がそのリクエストを受け取ると、リクエスト送信部１４２にて、その文字列と認証情報ＤＢ１４６内の希望されているデータを入れ替える。これにより、認証情報ＤＢ１４６内のデータは、ＧＷアプリ１００以外に漏れることなく、各アプリケーションから使用することが可能となる。

【００８４】

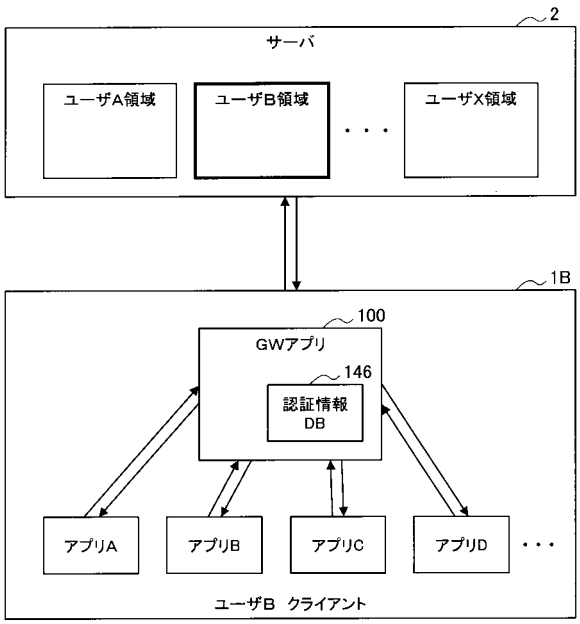
また、図２０に示すように、ＧＷアプリ１００は認証情報ＤＢ１４６に１ユーザのみでなく、多ユーザの認証情報を管理することができる。図２０は、多ユーザの認証情報を管理する場合の認証情報ＤＢ１４６のテーブル構成例である。また、各ユーザのログイン／ログオフといった状態を、ＧＷアプリ１００が提供するＡＰＩによって切り替えることが可能である。

【００８５】
図２１に複数ユーザがログインしている状態でリクエストＸＭＬが渡された時の動作例を示す。図２１に示すように、ＧＷアプリ１００は、各アプリケーションからリクエストＸＭＬを受け取ると、ログインしているユーザの認証情報を用いてサーバに対しリクエストを行う。ＧＷアプリ１００において複数のユーザがログイン状態に設定されている場合、ＧＷアプリ１００は各アプリケーションからリクエストＸＭＬを受け取ると、そのリクエストＸＭＬをそれぞれの認証情報を利用して、サーバ２上の各ユーザの領域に対してＡＰＩを実行する。ＧＷアプリ１００は各レスポンスをまとめて一つのレスポンスＸＭＬとして呼び出し元のアプリケーションに返す。
【００８６】
以上、添付図面を参照しながら本発明の好適な実施形態について詳細に説明したが、本発明はかかる例に限定されない。本発明の属する技術の分野における通常の知識を有する者であれば、特許請求の範囲に記載された技術的思想の範疇内において、各種の変更例または修正例に想到し得ることは明らかであり、これらについても、当然に本発明の技術的範囲に属するものと了解される。
【００８７】
例えば、上記実施形態では、ＧＷアプリ１００は、アプリケーションからのリクエストに認証情報を付加してサーバ２へ送信したが、本発明はかかる例に限定されない。クライアント端末１内のアプリケーションがリクエストするサーバ２のＡＰＩには、ユーザの認証情報を必要としないものもある。例えば、ＧＷアプリ１００は、複数のユーザによる複数のアプリケーションのリクエストをまとめて、各ユーザの個人領域ではなくサーバ２の共通領域に対してＡＰＩを実行することも可能である。このように、本発明のＧＷアプリ１００は、複数のアプリケーションからのリクエストを束ねて効率よくサーバ２と通信できるよう制御する。
【符号の説明】
【００８８】
１　　クライアント端末
２　　サーバ
１００　　ＧＷアプリ
１１０　　アプリ入出力インタフェース部
１２０　　データ解析部
１２１　　アプリキー認証部
１２２　　リクエストＩＤ変換部
１２３　　キャッシュ判断部
１２４　　同一リクエスト保管部
１２５　　送信リクエスト保管部
１２６　　端末状態監視部
１３０　　リクエスト／レスポンス変換部
１４０　　サーバ送受信インタフェース部１４０
１４６　　認証情報ＤＢ

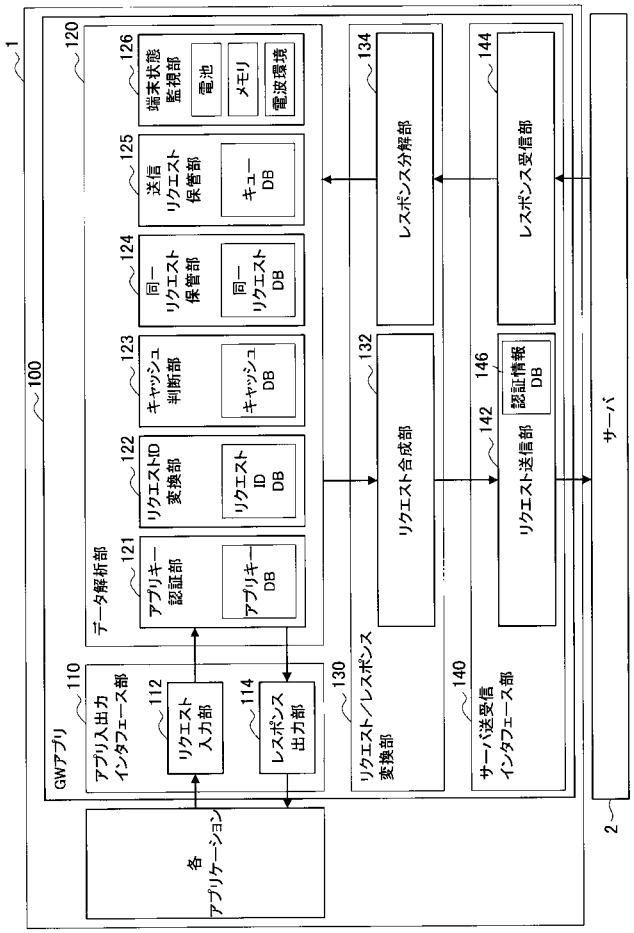
【図１】



【図２】



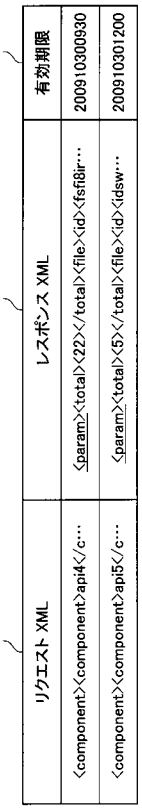
【図３】



【図４】

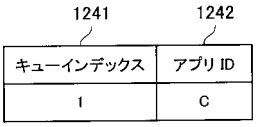
1221 リクエスト XML(or ID)	1222 リクエスト ID
<component><component>api6</component><param><arg1…	xxxxxxx3
<component><component>api7</component><param><arg1…	xxxxxxx7

【図５】



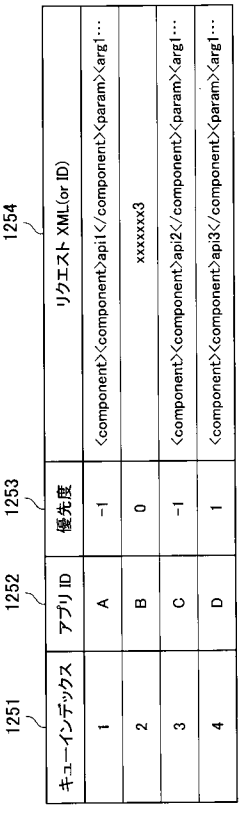
1231 リクエスト XML	1232 レスポンス XML	1233 有効期限
<component><component>api4</c…	<param><total>22</total><file><id>fsfi8ir…	2009103009300
<component><component>api5</c…	<param><total>5</total><file><id>idsw…	2009103012000

【図６】



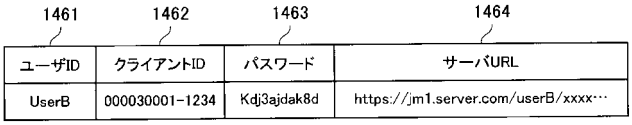
1241 キューインデックス	1242 アプリ ID
1	C

【図７】



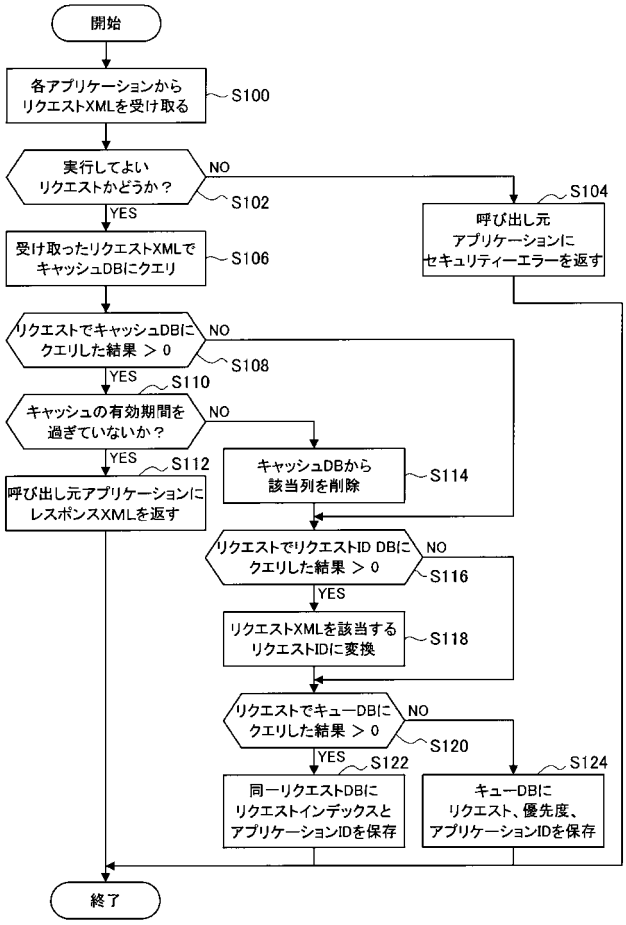
1251 キューインデックス	1252 アプリ ID	1253 優先度	1254 リクエスト XML(or ID)
1	A	-1	<component><component>api1</component><param><arg1…
2	B	0	xxxxxxx3
3	C	-1	<component><component>api2</component><param><arg1…
4	D	1	<component><component>api3</component><param><arg1…

【図８】

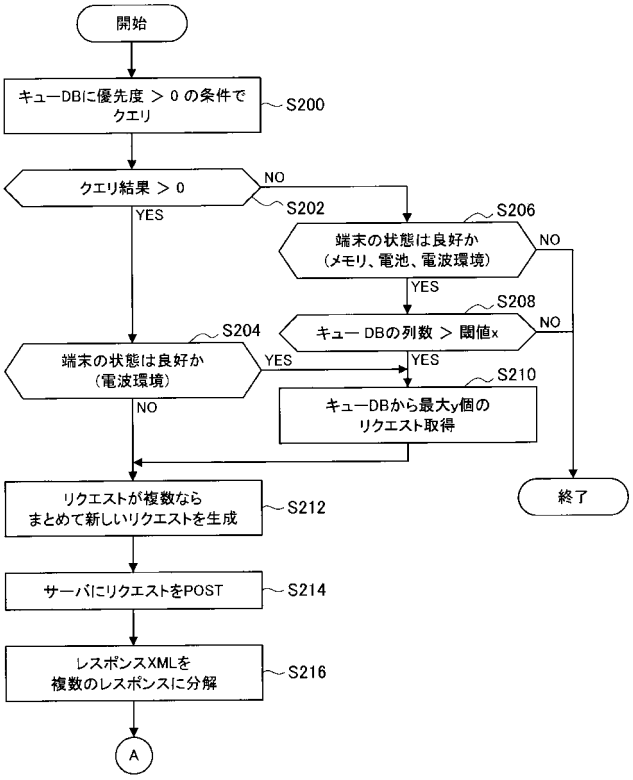


1461 ユーザID	1462 クライアントID	1463 パスワード	1464 サーバURL
UserB	000030001-1234	Kdj3ajdak8d	https://jm1.server.com/userB/xxxx…

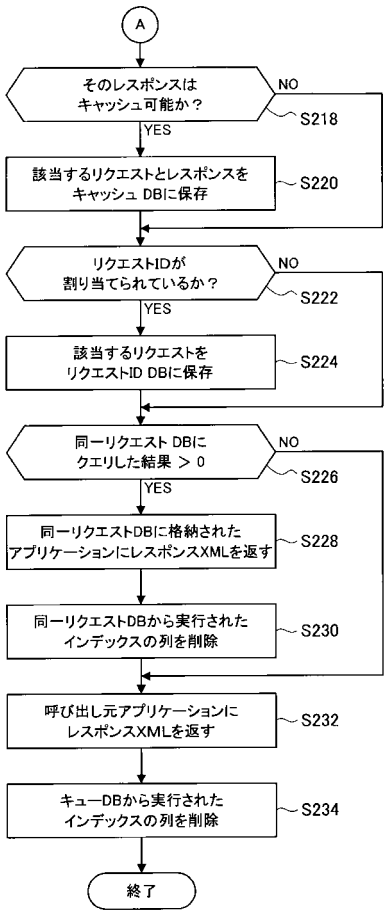
【図９】



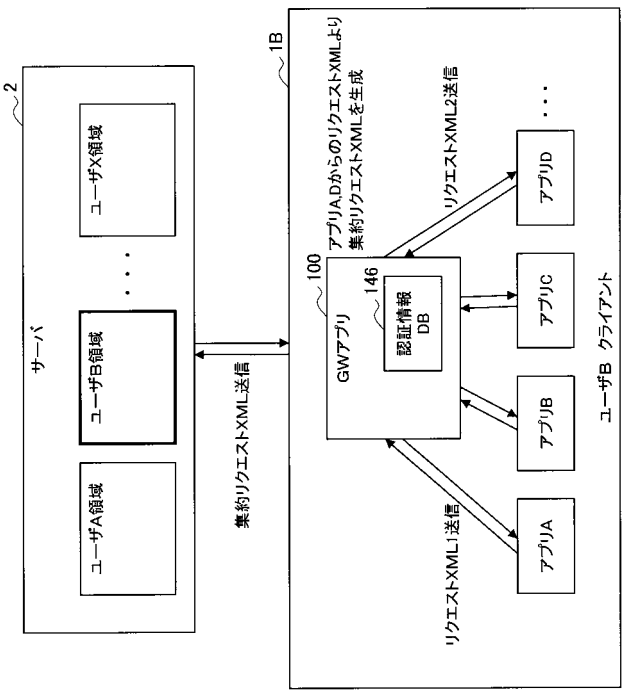
【図１０Ａ】



【図１０Ｂ】



【図１１】



【図１２】

```
<server>
  <component>
    <component>api1</component>  } 実行されるAPI名
    <param>
      <arg1>x</arg1>  } 実行されるAPIの引数
    </param>
  </component>
</server>
```

【図１３】

```
<server>
  <component>
    <component>api2</component>  } 実行されるAPI名
    <param>
      <arg1>y</arg1>
      <arg2>z</arg2>             } 実行されるAPIの引数
    </param>
  </component>
</server>U
```

【図１４】

```
<server>
  <component>                    ┐
    <component>api1</component>  │
    <param>                      │ 1番目に実行されるAPI
      <arg1>x</arg1>             │
    </param>                     │
  </component>                   ┘
  <component>                    ┐
    <component>api2</component>  │
    <param>                      │ 2番目に実行されるAPI
      <arg1>y</arg1>             │
      <arg2>z</arg2>             │
    </param>                     │
  </component>                   ┘
</server>
```

【図１５】

```
<server>
  <component>
    <code>0</code>  } 要求したAPI実行の成否
    <param>
      <result1>oooooooa</result1>  }
      <result2>ooooooob</result2>  } 実行結果
    </param>
  </component>
</server>
```

【図１６】

```
<server>
  <component>
    <code>0</code>  } 要求したAPI実行の成否
    <param>
      <result3>oooooooc</result3>  } 実行結果
    </param>
  </component>
</server>
```

【図１７】

```
<server>
  <component>
    <code>0</code>  } 1番目に要求したAPI実行の成否
    <param>
      <result1>oooooooa</result1>  }
      <result2>ooooooob</result2>  } 1番目のAPIの実行結果
    </param>
  </component>
  <component>
    <code>0</code>  } 2番目に要求したAPI実行の成否
    <param>
      <result3>oooooooc</result3>  } 2番目のAPIの実行結果
    </param>
  </component>
</server>
```

【図１８】

```
<server>
  <component>
    <code>0</code>
    <cache>200910300930</cache>  } キャッシュの有効期限
    <param>
      <result1>oooooood</result1>
      <result2>oooooooe</result2>
    </param>
  </component>
</server>
```

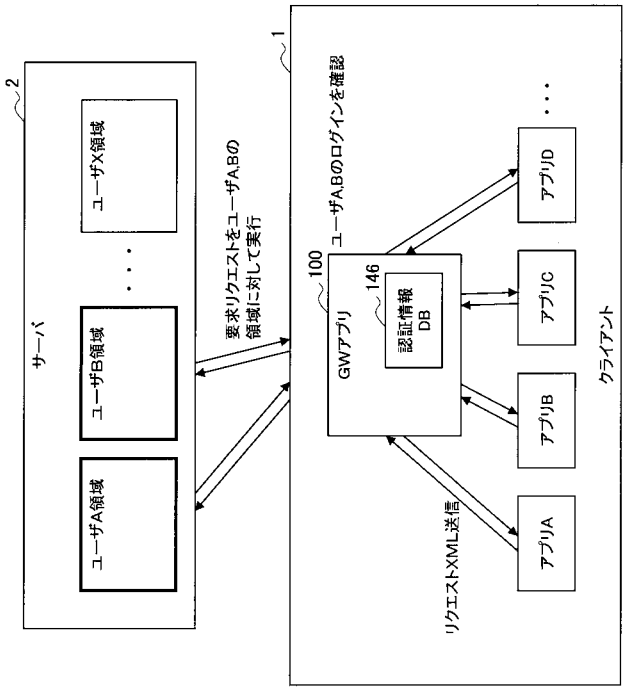
【図１９】

```
<server>
  <component>
    <code>0</code>
    <requestID>xxxxxxx3</requestID>  } リクエストIDの指定
    <param>
      <result1>oooooood</result1>
      <result2>oooooooe</result2>
    </param>
  </component>
</server>
```

【図２０】

1461 ユーザID	1462 クライアントID	1463 パスワード	1464 サーバURL	1465 ログイン
UserA	000030001-5678	Jfa3jasdi78	https://jm1.server.com/userA/xxxx…	Off
UserB	000030001-1234	Kdj3ajdak8d	https://jm1.server.com/userB/xxxx…	On

【図２１】


2
サーバ
ユーザA領域
ユーザB領域
ユーザX領域
要求リクエストをユーザA,Bの
領域に対して実行
1
ユーザA,Bのログインを確認
100
GWアプリ
146
認証情報
DB
リクエストXML送信
アプリA
アプリB
アプリC
アプリD
クライアント

フロントページの続き

(72)発明者　河内　勉
東京都品川区大崎１丁目１１番１号　ゲートシティ大崎ＷＥＳＴタワー９Ｆ　ヴィジョンアーツ株式会社内
(72)発明者　高田　昌幸
東京都港区港南１丁目７番１号　ソニー株式会社内

Ｆターム(参考) 5B285 AA01 BA03 CA17 CB02 CB63 CB72 CB85
5K034 AA02 AA09 HH06 MM08 MM22